Αλγόριθμοι τοπικής αναζήτησης με απαγόρευση κινήσεων και παραλληλοποίηση της αναζήτησης για την επίλυση του προβλήματος ροϊκής παραγωγής

Δημήτρης Παπαδάκης
Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Οκτώβριος 1995

Parallel Tabu Search Algorithms for the Flow-Shop Problem

Dimitris N. Papadakis

Master of Science Thesis

Department of Computer Science
University of Crete

ABSTRACT

During the last decades, the flowshop sequencing problem has held the attention of many researchers. The flowshop sequencing problem is a production scheduling problem in which each one of N jobs must be processed in the same sequence on each one of M machines.

Since Johnson proposed optimal two and three-stage production schedules, many heuristics have been suggested to solve this problem. Complete enumeration, branch and bound techniques or integer programming determine the optimal sequence for very small problems, but efficient heuristics are necessary in order to solve larger ones.

In this thesis two cases of the flowshop problem were dealt with. In the first case the objective was to minimize the maximum completion time of the jobs C_{max}, whereas in the second case each job was assigned a due date and the objective was to minimize primarily the maximum tardiness and secondary the maximum completion time of the jobs.

The computational complexity of the flowshop sequencing problem does not allow the development of computationally fast algorithms which could optimally solve it. In this thesis taboo search techniques were studied and heuristic algorithms were developed to suboptimally solve the flowshop sequencing problem.

Taboo search is a simple and flexible method for obtaining good solutions quickly and has been successfully applied to *NP*-complete problems such as the flowshop problem. The efficience of the method depends on the size of neighborhoods in the search space which are iteratively examined in order to obtain local optimums. In case that the optimum in a neighborhood is not unique the method can not determine the best path to continue the search.

To overcome the weaknesses of taboo search, a pruning path mechanism was developed which proved effective for both the problems of flowshop that were studied. In the case of

the flowshop problem with the unique criteria of minimizing the maximum completion time of the jobs, a tree search technique was developed which allows the evaluation of equivalent search paths. Finally, the process of taboo search was parallelized which drastically improved its running time.

Supervisor : Panos Constantopoulos
Associate Professor of Computer Science
University of Crete

Περίληψη

Τα προβλήματα χρονικού προγραμματισμού στόχο τους έχουν τον προγραμματισμό ενός συνόλου εργασιών κατά τέτοιον τρόπο ώστε να βελτιστοποιούνται ορισμένα κριτήρια και να ικανοποιούνται κάποιοι περιορισμοί που συνήθως τίθενται από ανώτερα στάδια του προγραμματισμού της παραγωγής. Τα προβλήματα αυτά παρουσιάζουν ιδιαίτερο ενδιαφέρον καθώς πηγάζουν από το χώρο της βιομηχανικής παραγωγής, ενώ παράλληλα αποτελούν στην πλειοψηφία τους δύσκολα συνδυαστικά προβλήματα, κατά κανόνα NP-δύσκολα (NP-hard).

Η ειδική κατηγορία προβλημάτων χρονικού προγραμματισμού που μελετήθηκαν στην εργασία αυτή αφορά την επεξεργασία εργασιών σε αλυσίδα μηχανών. Κάθε εργασία αποτελείται από μια αυστηρά διατεταγμένη ακολουθία κατεργασιών, οι οποίες εκτελούνται στις μηχανές. Επειδή η ροή επεξεργασίας των κατεργασιών είναι ενιαία για όλες τις εργασίες, το σύστημα χαρακτηρίζεται ως σύστημα Ροϊκής Παραγωγής (Flowshop). Η εκτέλεση των κατεργασιών στις μηχανές απαιτεί εξάρμωση των μηχανών. Οι χρονοι εξάρμωσης (setup time) των μηχανών είναι σταθεροί για κάθε κατεργασία και ανεξάρτητοι ακολουθίας.

Στην εργασία αυτή μελετήθηκαν δύο προβλήματα ροϊκής παραγωγής : στην πρώτη περίπτωση μοναδικό κριτήριο βελτίστου ήταν η ελαχιστοποίηση του χρόνου περάτωσης (C_{max}) του προγράμματος εργασιών, ενώ στη δεύτερη περίπτωση σε κάθε εργασία είχε ανατεθεί μια προθεσμία παράδοσης, οπότε πρωτεύον κριτήριο ήταν η ελαχιστοποίηση της μέγιστης καθυστέρησης (T_{max}) των εργασιών και η εύρεση εφικτού προγράμματος (T_{max} = 0), όποτε αυτό ήταν δυνατόν, και δευτερεύον κριτήριο ήταν η ελαχιστοποίηση του μέγιστου χρόνου περάτωσης του προγράμματος των εργασιών.

Η υπολογιστική πλοκή των προβλημάτων ροϊκής παραγωγής δεν επιτρέπει την ανάπτυξη ταχέων αλγορίθμων για τη βέλτιστη επίλυσή τους. Για το λόγο αυτό στην εργασία αυτή αναπτύχθηκαν ευρηματικοί αλγόριθμοι τοπικής αναζήτησης με απαγόρευση κινήσεων (Tabu Search) για την εύρεση υποβέλτιστων λύσεων. Η απόδοση των αλγορίθμων αυτών εξαρτάται κυρίως από το μέγεθος των υποπεριοχών του χώρου αναζήτησης που εξετάζονται σε κάθε βήμα για την εύρεση τοπικών βελτίστων και από το πλήθος των τοπικών βελτίστων που υπάρχουν στις υποπεριοχές αυτές. Όσο μεγαλύτερες υποπεριοχές εξετάζονται τόσο καλύτερες οι λύσεις που παράγονται αλλά και πιο απαιτητική σε υπολογιστικό χρόνο η διαδικασία της αναζήτησης. Επίσης, στις περιπτώσεις που το τοπικό βέλτιστο υποπεριοχών δεν είναι μοναδικό, η μυωπική λειτουργία της μεθόδου δεν εξα-

σφαλίζει ότι επιλέγεται πάντα η καλύτερη κατεύθυνση για τη βελτίωση των λύσεων. Για την αντιμετώπιση των προβλημάτων αυτών και τη βελτίωση της απόδοσης της μεθόδου σχεδιάστηκε ένας μηχανισμός αποκλεισμού κατευθύνσεων από το χώρο αναζήτησης, ο οποίος αποδείχθηκε αποτελεσματικός και για τα δύο προβλήματα ροϊκής παραγωγής που μελετήθηκαν. Ειδικά για το μονοκριτηριακό πρόβλημα αναπτύχθηκε ένας μηχανισμός δενδροειδούς αναζήτησης, ο οποίος επιτρέπει την αξιολόγηση ισοδύναμων κατευθύνσεων βελτίωσης λύσεων. Τέλος, η φύση της λειτουργίας της διαδικασίας αναζήτησης μας οδήγησε στην παραλληλοποίησή της με αποτέλεσμα τη δραστική βελτίωση της απόδοσής της.

Επόπτης : Πάνος Κωνσταντόπουλος
Αναπληρωτής Καθηγητής, Τμήματος Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Χρονικός προγραμματισμός

Τα προβλήματα χρονικού προγραμματισμού στόχο τους έχουν τον προγραμματισμό ενός συνόλου εργασιών κατά τέτοιον τρόπο ώστε να βελτιστοποιούνται ορισμένα κριτήρια και να ικανοποιούνται κάποιοι περιορισμοί που συνήθως τίθενται από ανώτερα στάδια του προγραμματισμού της παραγωγής. Τα προβλήματα αυτά παρουσιάζουν ιδιαίτερο ενδιαφέρον καθώς πηγάζουν από το χώρο της βιομηχανικής παραγωγής, ενώ παράλληλα αποτελούν στην πλειοψηφία τους δύσκολα συνδυαστικά προβλήματα, κατά κανόνα NP-δύσκολα (NP-hard).

1.2 Ταξινόμηση προβλημάτων χρονικού προγραμματισμού

Τα προβλήματα χρονικού προγραμματισμού ταξινομούνται με διάφορα κριτήρια σε κατηγορίες [GLLK79, Gra81, HC84]. Τα σημαντικότερα κριτήρια ταξινόμησης είναι το είδος του εργοστασίου, η ύπαρξη και το είδος των χρόνων εξάρμωσης και το κριτήριο βελτίστου που χρησιμοποιούμε στον προγραμματισμό.

Το είδος του εργοστασίου καθορίζεται από τη συγκρότηση της διαδικασίας παραγωγής. Γενικά, διακρίνουμε τέσσερεις περιπτώσεις :

- **Μοναδική Μηχανή (Single Machine)**

  Κάθε εργασία αποτελείται από μία μοναδική κατεργασία, η οποία εκτελείται στη

μοναδική μηχανή του εργοστασίου.

- **Παράλληλες Μηχανές (Parallel Machines)**

  Μπορεί να θεωρηθεί ως επέκταση της *μοναδικής μηχανής*, καθώς οι εργασίες αποτελούνται από μια μοναδική κατεργασία, η οποία μπορεί να εκτελεσθεί σε οποιαδήποτε από ένα πλήθος μηχανών που υπάρχουν στο εργοστάσιο.

- **Ροϊκή Παραγωγή (Flowshop)**

  Οι εργασίες διαιρούνται σε κατεργασίες, μία για κάθε μηχανή του εργοστασίου. Κάθε κατεργασία εκτελείται σε μία μηχανή και η σειρά διέλευσης από τις μηχανές είναι σταθερή.

- **Γενικό Εργοστάσιο (General Jobshop)**

  Το γενικό εργοστάσιο αντιπροσωπεύει το γενικώτερο πρόβλημα. Κάθε εργασία μπορεί να περιλαμβάνει κατεργασίες σε οποιεσδήποτε μηχανές του εργοστασίου. Οι κατεργασίες μπορούν να εκτελεσθούν με οποιαδήποτε σειρά στις μηχανές ή οι δυνατές αλληλουχίες να καθορίζονται από μια μερική διάταξη διαδοχής.

Το δεύτερο κριτήριο ταξινόμησης είναι η ύπαρξη και το είδος χρόνων εξάρμωσης των μηχανών (setup time). Χρόνο εξάρμωσης μιας μηχανής ονομάζουμε το χρονικό διάστημα που απαιτείται ώστε η μηχανή να προετοιμασθεί, για να εκτελέσει μια εργασία. Ο χρόνος εξάρμωσης μιας μηχανής λέγεται ότι είναι ανεξάρτητος ακολουθίας (sequence independent) αν εξαρτάται μόνο από το είδος της εργασίας που πρόκειται να εκτελεσθεί στη μηχανή. Αν εξαρτάται και από το είδος της εργασίας της οποίας η επεξεργασία προηγήθηκε στη μηχανή, τότε λέγεται ότι είναι εξαρτημένος ακολουθίας (sequence dependent).

Το κριτήριο βελτίστου είναι ένα από τα σημαντικότερα χαρακτηριστικά των προβλημάτων χρονικού προγραμματισμού. Τα περισσότερα κριτήρια που αναφέρονται στη βιβλιογραφία ανήκουν στη κατηγορία των λεγομένων κανονικών (regular) κριτηρίων.

Ένα κριτήριο βελτίστου λέγεται κανονικό όταν ικανοποιεί τις παρακάτω τρεις συνθήκες :

1. Είναι συνάρτηση των χρόνων περάτωσης των εργασιών
2. Η τιμή του πρέπει να ελαχιστοποιηθεί.

3. Η τιμή του αυξάνει, μόνο αν ένας τουλάχιστον από τους χρόνους περάτωσης των εργασιών αυξηθεί.

Μερικά από τα συνηθέστερα κανονικά κριτήρια είναι ο χρόνος περάτωσης προγράμματος (κριτήριο C_{max}), ο μέσος χρόνος περάτωσης των εργασιών (κριτήριο $\sum_{i=1}^{N} C_i$), στην περίπτωση που υπάρχουν προθεσμίες παράδοσης η μέγιστη καθυστέρηση (κριτήριο T_{max}), η μέση καθυστέρηση (κριτήριο $\sum_{i=1}^{N} T_i$) και το πλήθος των καθυστερημένων εργασιών.

Επειδή ένα συγκεκριμένο πρόβλημα χρονικού προγραμματισμού μπορεί να ανήκει σε πολλές κατηγορίες ταυτόχρονα, είναι απαραίτητο ένα γενικό σχήμα για τη συνοπτική περιγραφή των προβλημάτων. Στην παρούσα εργασία θα χρησιμοποιήσουμε τον ακόλουθο, ευρύτατα διαδεδομένο συμβολισμό [GLLK79, KK88] :

α / β / γ / δ

όπου:

α : ο αριθμός των εργασιών προς προγραμματισμό

β : ο αριθμός των μηχανών

γ : περιορισμοί στα δεδομένα του προβλήματος

δ : το κριτήριο βελτίστου

Να σημειωθεί ότι το πεδίο γ ενδέχεται να είναι κενό, οπότε δεν έχουμε κανένα περιορισμό ή να περιέχει περισσότερους από ένα περιορισμούς. Επίσης, το πεδίο δ μπορεί να περιλαμβάνει περισσότερα από ένα κριτήρια βελτίστου ή να είναι κενό οπότε αναφερόμαστε σε ένα οποιοδήποτε κριτήριο βελτίστου. Ο παραπάνω τρόπος ταξινόμησης αν και δεν παρέχει όλες τις λεπτομέρειες που συνθέτουν ένα πρόβλημα χρονικού προγραμματισμού, δίνει συνοπτικά τα σπουδαιότερα χαρακτηριστικά του.

1.3 Ορισμός του προβλήματος

Η ειδική κατηγορία προβλημάτων χρονικού προγραμματισμού που εξετάζονται στην παρούσα εργασία αφορά την επεξεργασία εργασιών σε αλυσίδα μηχανών. Κάθε εργασία

αποτελείται από μια αυστηρά διατεταγμένη ακολουθία κατεργασιών, οι οποίες εκτελούνται στις μηχανές. Επειδή η ροή επεξεργασίας των κατεργασιών στις μηχανές είναι ενιαία για όλες τις εργασίες, το σύστημα χαρακτηρίζεται ως σύστημα ροϊκής παραγωγής. Για τα προβλήματα που μελετάμε στην παρούσα εργασία κάνουμε τις παρακάτω υποθέσεις :

- Κάθε εργασία χωρίζεται σε M κατεργασίες, οι οποίες εκτελούνται σε διαφορετική μηχανή η καθεμία. Η σειρά εκτέλεσης των κατεργασιών είναι η ίδια για όλες τις εργασίες.
- Μια εργασία είναι διαθέσιμη σε μια 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 μια εργασία σε δοσμένη χρονική στιγμή. Η εκτέλεση μιας εργασίας δε διακόπτεται (non-preemption). Η σειρά εκτέλεσης των εργασιών είναι η ίδια σε όλες τις μηχανές της γραμμής παραγωγής (μεταθετικά προγράμματα).
- Ο μόνος κρίσιμος πόρος του συστήματος είναι η μηχανή. Όλοι οι άλλοι πόροι θεωρούνται επαρκείς.
- Η επεξεργασία κάθε εργασίας σε κάθε μηχανή εν γένει απαιτεί εξάρμωση της μηχανής. Οι χρόνοι εξάρμωσης των μηχανών θεωρούνται ανεξάρτητοι ακολουθίας.
- Οι εργασίες χωρίζονται σε B ομάδες ανάλογα με το είδος τους. Μεταξύ διαδοχικών εργασιών της ίδιας ομάδας δεν απαιτείται εξάρμωση μηχανής. Οι ομάδες των εργασιών είναι ίδιες για όλες τις μηχανές.
- Οι χρόνοι εξάρμωσης των μηχανών, οι χρόνοι επεξεργασίας των εργασιών και οι προθεσμίες τους, όποτε αυτές υπάρχουν, θεωρούνται γνωστές *a priori*.

Στην εργασία αυτή μελετώνται δύο προβλήματα ροϊκής παραγωγής.

- $N/\ M/\ F,\ Seq-Ind,\ perm/\ C_{max}$

  Στο πρόβλημα αυτό θεωρούμε τη διάταξη N εργασιών σε M μηχανές. Στο πεδίο των περιορισμών το F δηλώνει ότι πρόκειται για πρόβλημα ροϊκής παραγωγής και το $perm$ ότι οι λύσεις περιορίζονται σε μεταθετικά προγράμματα μόνο. Υπάρχουν χρόνοι εξάρμωσης μηχανών, οι οποίοι είναι ανεξάρτητοι ακολουθίας ($Seq-Ind$).

Το κριτήριο βελτίστου είναι η ελαχιστοποίηση του χρόνου περάτωσης C_{max} του προγράμματος εργασιών.

- $N/\ M/\ F,\ Seq-Ind,\ perm/\ T_{max},\ C_{max}$

  Στο δεύτερο πρόβλημα θεωρούμε πάλι τη διάταξη N εργασιών σε M μηχανές. Το πρόβλημα είναι ροϊκής παραγωγής και υπάρχουν χρόνοι εξάρμωσης μηχανών, οι οποίοι είναι ανεξάρτητοι ακολουθίας και οι λύσεις είναι μεταθετικές. Κάθε εργασία έχει μία προθεσμία παράδοσης, η οποία μπορεί να της έχει ανατεθεί από ανώτερο στάδιο προγραμματισμού. Μια εργασία έχει καθυστέρηση αν η εκτέλεσή της ολοκληρώνεται μετά την προθεσμία της. Ένα πρόγραμμα στο οποίο δεν υπάρχουν καθυστερημένες εργασίες λέγεται εφικτό. Πρωτεύον κριτήριο βελτίστου είναι η ελαχιστοποίηση της μέγιστης καθυστέρησης T_{max} των εργασιών και η εύρεση εφικτού προγράμματος $(T_{max} = 0)$. Δευτερεύον κριτήριο βελτίστου είναι η ελαχιστοποίηση του χρόνου περάτωσης του προγράμματος.

1.4 Περιεχόμενο της εργασίας

Η υπολογιστική πλοκή των προβλημάτων ροϊκής παραγωγής και ο χαρακτηρισμός τους ως *NP*-πλήρων [GJ91] δεν επιτρέπει την ανάπτυξη ταχέων αλγορίθμων για τη βέλτιστη επίλυσή τους. Για το λόγο αυτό, η έρευνα στράφηκε σε προσεγγιστικούς αλγορίθμους και στην εύρεση υποβέλτιστων λύσεων. Καθώς, όμως, το μέγεθος των προβλημάτων αυξάνει και ο χώρος αναζήτησης του βελτίστου μεγαλώνει, η απόδοση πολλών ευρηματικών αλγορίθμων μειώνεται, με αποτέλεσμα η λύση των προβλημάτων άλλοτε να μην είναι ικανοποιητική και άλλοτε να είναι χρονοβόρος. Η ανάπτυξη παράλληλων μηχανών και η ευρεία χρήση τους τα τελευταία χρόνια έδωσε ώθηση στην ανάπτυξη παράλληλων αλγορίθμων για τη γρήγορη, όσον αφορά τον υπολογιστικό χρόνο που απαιτείται, και αποτελεσματική, όσον αφορά την ποιότητα των παραγόμενων λύσεων, επίλυση προβλημάτων μεγάλου μεγέθους. Στην παρούσα εργασία παρουσιάζουμε παράλληλες μεθόδους επαναληπτικής βελτίωσης προγραμμάτων με απαγόρευση κινήσεων και αναπτύσσουμε νέους ευρηματικούς αλγορίθμους, οι οποίοι χρησιμοποιούν τις μεθόδους αυτές, για τη λύση των προβλημάτων $N/\ M/F,\ Seq-Ind,\ perm/C_{max}$ και $N/\ M/F,\ Seq-Ind,\ perm/T_{max},\ C_{max}$

Η μέθοδος επαναληπτικής βελτίωσης λύσεων με απαγόρευση κινήσεων (tabu search) προτάθηκε από τον Glover [Glo90b, Glo89, Glo90a] για τη λύση συνδυαστικών προβλημά-

των και υπερέχει των γνωστών μεθόδων τοπικής αναζήτησης, Η απαγόρευση ορισμένων κινήσεων σε κάθε βήμα κατευθύνει την έρευνα στο χώρο των λύσεων αποφεύγοντας κύκλους στην αναζήτηση και αποτρέποντας την παγίδευση σε τοπικά βέλτιστα. Η εφαρμογή της στη λύση προβλημάτων ροϊκής παραγωγής [WH89, LBG89, BL91] έδειξε ότι υπερτερεί άλλων ευρηματικών μεθόδων.

Μελετώντας την εφαρμογή της μεθόδου αυτής στη λύση προβλημάτων ροϊκής παραγωγής, διαπιστώσαμε ότι η απόδοσή της εξαρτάται κυρίως από το μέγεθος της υποπεριοχής του χώρου έρευνας που εξετάζεται σε κάθε βήμα για την εύρεση τοπικού βελτίστου και από το πλήθος των τοπικών βελτίστων που υπάρχουν στις υποπεριοχές αυτές. Όσο μεγαλύτερες υποπεριοχές εξετάζονται τόσο καλύτερες οι λύσεις που παράγονται αλλά και πιο απαιτητική σε υπολογιστικό χρόνο η διαδικασία αναζήτησης. Επίσης, στις περιπτώσεις που το τοπικό ελάχιστο υποπεριοχών δεν είναι μοναδικό, η μυωπική λειτουργία της μεθόδου δεν εξασφαλίζει ότι επιλέγεται η καλύτερη κατεύθυνση για τη βελτίωση των λύσεων. Για την αντιμετώπιση των προβλημάτων αυτών και τη βελτίωση της απόδοσης της μεθόδου σχεδιάσαμε ένα μηχανισμό αποκλεισμού κατευθύνσεων από το χώρο αναζήτησης, ο οποίος αποδείχθηκε αποτελεσματικός και για τα δύο προβλήματα ροϊκής παραγωγής που μελετάμε. Ειδικά για το πρόβλημα $N/M/F, Seq-Ind, perm/C_{max}$ αναπτύξαμε ένα μηχανισμό δενδροειδούς αναζήτησης, ο οποίος επιτρέπει την αξιολόγηση ισοδύναμων κατευθύνσεων βελτίωσης λύσεων. Τέλος, η φύση της λειτουργίας της διαδικασίας αναζήτησης μας οδήγησε στην παραλληλοποίησή της με αποτέλεσμα τη δραστική βελτίωση της απόδοσής της.

Επιλέξαμε ως μέτρο αξιολόγησης των επιδόσεων των αλγορίθμων που προτείνονται στην παρούσα εργασία, τους αλγορίθμους του Κοπιδάκη [Κοπ93]. Οδηγηθήκαμε στην επιλογή αυτή, αφενός γιατί στο μονοκριτηριακό πρόβλημα που μελετάμε ο αλγόριθμος του Κοπιδάκη εμφανίζεται με τις καλύτερες επιδόσεις μεταξύ άλλων ευρηματικών μεθόδων [Naw83, WH89, Κοπ93], αφετέρου γιατί για το δικριτηριακό πρόβλημα αποτελεί το μοναδικό γνωστό αλγόριθμο επίλυσής του.

Συστηματικές δοκιμές των σειριακών αλγορίθμων για διάφορα μεγέθη τυχαίων προβλημάτων και μεταβαλλόμενες τιμές παραμέτρων έδειξαν ότι ο αλγόριθμος που προτείνουμε για την επίλυση του μονοκριτηριακού προβλήματος εμφανίζεται να παράγει λύσεις ποιοτικά ισάξιες των λύσεων του αλγορίθμου του Κοπιδάκη. Η χρήση του μηχανισμού της δενδροειδούς αναζήτησης βελτιώνει τις παραγόμενες λύσεις 1% κατά μέσο όρο. Στο δικριτηριακό πρόβλημα, ο αλγόριθμος που αναπτύξαμε για την επίλυσή του υπερέχει 16,7% κατά μέσο όρο για το πρωτεύον κριτήριο και 3,5% κατά μέσο όρο

για το δευτερεύον, ως προς τον αλγόριθμο του Κοπιδάκη. Ο μηχανισμός αποκλεισμού κατευθύνσεων από το χώρο αναζήτησης μείωσε 25% κατά μέσο όρο τον υπολογιστικό χρόνο της εκτέλεσής τους, χωρίς να επηρεάζεται η ποιότητα των παραγόμενων λύσεων. Από τις δοκιμές και τη σύγκριση των παράλληλων με τους σειριακούς αλγορίθμους προέκυψε ότι με την παραλληλοποίηση της διαδικασίας αναζήτησης επιτυγχάνεται σχεδόν γραμμική επιτάχυνση (linear speedup). Συγκεκριμένα, η εκτέλεση των αλγορίθμων σε παράλληλη μηχανή με τέσσερεις επεξεργαστές και ο διαμερισμός των υποπεριοχών που εξετάζονται σε δύο και τέσσερα μέρη προκάλεσε κατά μέσο όρο επιτάχυνση $1,9$ και $3,7$ αντίστοιχα.

Η δομή της υπόλοιπης εργασίας ακολουθεί σε γενικές γραμμές τη σειρά παρουσίασης των παραπάνω θεμάτων. Στο κεφάλαιο 2 μελετάται η φύση του προβλήματος ροϊκής παραγωγής και αναλύονται οι ιδιότητες των λύσεων του. Στη συνέχεια, στο κεφάλαιο 3, παρουσιάζονται τα σημαντικότερα αναλυτικά αποτελέσματα των προβλημάτων $N/\,M/F,\, Seq-Ind,\, perm/C_{max}$ και $N/\,M/F,\, Seq-Ind,\, perm/T_{max},\, C_{max}$ και μια ανασκόπηση των γνωστότερων μεθόδων επίλυσής τους. Στο κεφάλαιο 4 αναλύονται η μέθοδος της επαναληπτικής βελτίωσης προγράμματος με παρεμβολή εργασιών και ο μηχανισμός απαγόρευσης κινήσεων και αναπτύσσονται δύο νέοι ευρηματικοί σειριακοί αλγόριθμοι για τη λύση των προβλημάτων. Οι λόγοι που μας ώθησαν στην ανάπτυξη παράλληλων αλγορίθμων περιγράφονται στο κεφάλαιο 5. Αναλύεται η μέθοδος της παράλληλης επαναληπτικής βελτίωσης, αναπτύσσονται οι ευρηματικοί παράλληλοι αλγόριθμοι και συγκρίνονται με τους σειριακούς. Στο κεφάλαιο 6 αναλύεται το πρόβλημα των πολλαπλών τοπικών βελτίστων στις περιοχές αναζήτησης και προτείνεται η τεχνική της δενδροειδούς αναζήτησης για την αντιμετώπισή του. Τα αποτελέσματα των πειραματικών συγκρίσεων των αλγορίθμων και η αξιολόγηση των τεχνικών που προτάθηκαν παρουσιάζεται στο κεφάλαιο 7. Τέλος, στο κεφάλαιο 8 παρατίθενται τα συμπεράσματα της εργασίας για τη φύση του προβλήματος και την αποτελεσματικότητα των αλγορίθμων που αναπτύχθηκαν, ενώ προτείνονται επεκτάσεις τους.

Κεφάλαιο 2

Περιγραφή του προβλήματος

2.1 Υπολογισμός χρόνου περάτωσης προγράμματος

Στο πρόβλημα ροϊκής παραγωγής ο χρόνος περάτωσης ενός προγράμματος N εργασιών σε M μηχανές, ισούται με το μέγιστο χρόνο εκτέλεσης των εργασιών στην τελευταία μηχανή. Αν συμβολίσουμε C_{max} το χρόνο περάτωσης του προγράμματος, τότε ισχύει :

$$C_{max} = \max(C_{iM}), \quad i = 1, 2, ..., N \tag{2.1}$$

Από τη σχέση αυτή γίνεται φανερό ότι για τον υπολογισμό του χρόνου περάτωσης ενός προγράμματος είναι απαραίτητος ένας μηχανισμός υπολογισμού του χρόνου περάτωσης των εργασιών σε κάθε μηχανή του συστήματος. Ένας τέτοιος μηχανισμός περιγράφεται στην βιβλιογραφία [Bak75, PGD91], τον οποίο θα επεκτείνουμε για να συμπεριλάβει και τους χρόνους εξάρμωσης των μηχανών. Να υπενθυμίσουμε ότι οι χρόνοι εξάρμωσης είναι ανεξάρτητοι ακολουθίας.

Κατά την άφιξη μιας εργασίας i σε μια μηχανή j της γραμμής παραγωγής διακρίνουμε δύο περιπτώσεις :

- Η μηχανή j είναι απασχολημένη με την εκτέλεση της προηγούμενης εργασίας του προγράμματος ή με την εξάρμωση της μηχανής για την εκτέλεση της εργασίας i. Στην περίπτωση αυτή η εργασία i αποθηκεύεται μέχρι να ολοκληρωθεί η εκτέλεση της προηγούμενης εργασίας και η εξάρμωση της μηχανής για την εργασία i. Ο χρόνος που μεσολαβεί από τη χρονική στιγμή $C_{i-1,j}$ που η εκτέλεση της προηγούμενης εργασίας ολοκληρώνεται στη μηχανή j , έως τη χρονική στιγμή C_{ij}

που ολοκληρώνεται η εκτέλεση της εργασίας i στην ίδια μηχανή, ισούται με το άθροισμα των χρόνων εξάρμωσης και επεξεργασίας της εργασίας i.

Επομένως, ισχύει :

$$C_{ij} \ = \ C_{i-1,j} + s_{ij} + p_{ij} \tag{2.2}$$

- Η μηχανή j είναι αμέσως διαθέσιμη για την εκτέλεση της εργασίας i και η επεξεργασία της αρχίζει τη χρονική στιγμή που η εκτέλεση της εργασίας i ολοκληρώνεται στην προηγούμενη μηχανή. Στην περίπτωση αυτή πρέπει να έχει ολοκληρωθεί και η εκτέλεση της προηγούμενης εργασίας και η εξάρμωση της μηχανής j για την εργασία i.

  Επομένως, ισχύει :

$$C_{ij} \ = \ C_{i,j-1} + p_{ij} \tag{2.3}$$

  Στη δεύτερη περίπτωση είναι πιθανή η εμφάνιση νεκρού χρόνου (idle time) στη μηχανή j μεταξύ της εκτέλεσης των εργασιών $i-1$ και i . Κατά το χρόνο αυτό η μηχανή j μένει ανεκμετάλευτη.

Συνδυάζοντας τις (2) και (3) προκύπτει ότι στη γενική περίπτωση ο χρόνος περάτωσης της εργασίας i στη μηχανή j δίνεται από τη σχέση :

$$C_{ij} \ = \ \max(C_{i-1,j} + s_{ij}, \ C_{i,j-1}) \ + \ p_{ij} \tag{2.4}$$

Από τη σχέση (2.4) μπορούμε να υπολογίσουμε το χρόνο περάτωσης των εργασιών στις μηχανές της γραμμής παραγωγής.

2.2 Ύπαρξη νεκρών χρόνων στις μηχανές

Νεκρό χρόνο (idle time) ονομάζουμε το χρονικό διάστημα μέσα στο οποίο μια μηχανή του συστήματος δε χρησιμοποιείται. Νεκρός χρόνος εμφανίζεται στην περίπτωση που μια μηχανή έχει ολοκληρώσει τόσο την επεξεργασία μιας κατεργασίας, όσο και την εξάρμωση για την εκτέλεση της επόμενης, ενώ η επεξεργασία της επόμενης δεν έχει ακόμη ολοκληρωθεί στην προηγούμενη μηχανή του συστήματος.

Έστω ότι η εργασία i εκτελείται στη μηχανή $j-1$ και η προηγούμενή της $i-1$ εκτελείται στη μηχανή j. Αν η επεξεργασία της εργασίας $i-1$ ολοκληρωθεί στη μηχανή

j και η μηχανή εξαρμωθεί για την εργασία i, προτού η επεξεργασία της τελευταίας ολοκληρωθεί στην προηγούμενη μηχανή, τότε στη μηχανή j θα εμφανιστεί νεκρός χρόνος I_{ij}, ο οποίος ισούται με $C_{i,j-1} - C_{i-1,j} - s_{ij}$. Στην αντίθετη περίπτωση που η εκτέλεση της εργασίας i έχει ολοκληρωθεί στη μηχανή $j-1$, ενώ η μηχανή j δεν είναι έτοιμη να την επεξεργαστεί, δεν έχουμε νεκρό χρόνο. Ισχύει, λοιπόν, ότι :

$$I_{ij} = \max(C_{i,j-1} - C_{i-1,j} - s_{ij},\ 0) \tag{2.5}$$

Στην ενότητα 2.1 είδαμε ότι ο συνολικός χρόνος εκτέλεσης C_{max} των N εργασιών ενός προγράμματος ισούται με το μέγιστο χρόνο εκτέλεσης των εργασιών στην τελευταία μηχανή του συστήματος (σχέση 2.1). Αυτό σημαίνει ότι ο χρόνος C_{max} μπορεί να υπολογισθεί ως το άθροισμα των χρόνων εξάρμωσης και επεξεργασίας των N εργασιών πάνω στην τελευταία μηχανή συν το νεκρό χρόνο της μηχανής αυτής. Επομένως, ισχύει ότι :

$$C_{max} = \sum_{i=1}^{N} p_{iM} + \sum_{i=1}^{N} s_{iM} + \sum_{i=1}^{N} I_{iM} \tag{2.6}$$

Από τη σχέση (2.6) βλέπουμε ότι η εμφάνιση νεκρών χρόνων στις μηχανές του συστήματος έχει ως συνέπεια την καθυστέρηση της ροής της παραγωγής. Ειδικότερα, η ελαχιστοποίηση του νεκρού χρόνου της τελευταίας μηχανής της γραμμής παραγωγής ελαχιστοποιεί το συνολικό χρόνο εκτέλεσης των εργασιών.

2.3 Παρτιδοποίηση εργασιών

Δύο ή περισσότερες εργασίες ονομάζονται ομοειδείς αν κατά τη διαδοχική τους εκτέλεση δεν απαιτείται εξάρμωση της μηχανής, στην οποία εκτελούνται. Η ομαδοποίηση ομοειδών εργασιών ώστε να εκτελούνται συνεχόμενες χωρίς εξάρμωση ονομάζεται παρτιδοποίηση. Μια ομάδα διαδοχικών ομοειδών εργασιών ονομάζεται παρτίδα.

Η σχέση (2.4) που χρησιμοποιήσαμε για τον υπολογισμό του χρόνου περάτωσης ενός προγράμματος, χρειάζεται να τροποποιηθεί ώστε να μη συμπεριλαμβάνει τους χρόνους εξάρμωσης μεταξύ ομοειδών εργασιών. Όπως και στην παράγραφο 2.1, οι χρόνοι εξάρμωσης θεωρούνται ανεξάρτητοι ακολουθίας. Οπότε, η σχέση (2.4) παίρνει τη μορφή :

$$C_{ij} = \max(C_{i-1,j} + s^{g}_{ij},\ C_{i,j-1}) + p_{ij} \tag{2.7}$$

με $s_{ij}^{g} = \begin{cases} 0 & \text{αν } G_{i-1} = G_i \\ s_{ij} & \text{αλλιώς} \end{cases}$

όπου G_i η ομάδα στην οποία ανήκει η εργασία i.

Όμοια τροποποιείται και η σχέση (2.6)

$$C_{max} = \sum_{i=1}^{N} p_{iM} + \sum_{i=1}^{N} s_{iM}^{g} + \sum_{i=1}^{N} I_{iM} \tag{2.8}$$

Από τις σχέσεις (2.7) και (2.8) βλέπουμε ότι η παρτιδοποίηση των εργασιών συμβάλλει στην ελαχιστοποίηση του χρόνου περάτωσης C_{max} ενός προγράμματος, καθώς μειώνει το συνολικό χρόνο εξάρμωσης των μηχανών. Στα προβλήματα μίας μηχανής $N \,/\, 1 \,/\, Seq - Ind \,/$ η παρτιδοποίηση των ομοειδών εργασιών αποτελεί το κρισιμότερο σημείο. Συνήθως, επιδιώκεται το μέγεθος των παρτίδων να είναι μέγιστο, ώστε ο συνολικός χρόνος εξάρμωσης των μηχανών να είναι ο ελάχιστος δυνατός. Το μέγεθος των παρτίδων καθώς και η σειρά εκτέλεσης των εργασιών σε κάθε παρτίδα καθορίζεται από το κριτήριο βελτίστου που έχουμε επιλέξει. Αν οι εργασίες έχουν προθεσμίες παράδοσης και το κριτήριο βελτίστου είναι η μέγιστη καθυστέρηση T_{max} τότε οι εργασίες σε κάθε παρτίδα εκτελούνται κατά αύξουσα σειρά των προθεσμιών τους [MP89], ενώ στην ειδική περίπτωση που το κριτήριο βελτίστου είναι ο χρόνος περάτωσης C_{max} του προγράμματος, το μέγεθος των παρτίδων είναι μέγιστο και ο αριθμός τους ισούται με το πλήθος των ομάδων [HC84] .

Στο πρόβλημα ροϊκής παραγωγής, όμως, η παρτιδοποίηση των ομοειδών εργασιών δεν αποτελεί τον κρισιμότερο παράγοντα στην αναζήτηση βέλτιστης λύσης [Κοπ93] . Σε πολλές περιπτώσεις η εξάρμωση μιας μηχανής μπορεί να εκτελεσθεί σε νεκρό χρόνο με αποτέλεσμα να μην επηρεάζει το χρόνο περάτωσης του προγράμματος, όπως φαίνεται από τη σχέση (2.3). Σε άλλες περιπτώσεις η ελαχιστοποίηση των χρόνων εξάρμωσης ενδέχεται να αυξήσει τους νεκρούς χρόνους στις μηχανές με αποτέλεσμα την αύξηση του συνολικού χρόνου εκτέλεσης των εργασιών.

2.4 Μεταθετικά προγράμματα

Μεταθετικό (permutation schedule) ονομάζουμε το πρόγραμμα στο οποίο η διάταξη των εργασιών είναι η ίδια σε κάθε μηχανή. Δεν επιτρέπονται, επομένως, αλλαγές στη

σειρά εκτέλεσης των εργασιών από μηχανή σε μηχανή. Κατά συνέπεια, υπάρχουν $N!$ διαφορετικές διατάξεις των N εργασιών, κάθεμια από τις οποίες περιγράφει πλήρως ένα μεταθετικό πρόγραμμα.

Αντίθετα, σ' ένα μη μεταθετικό πρόγραμμα (non-permutation schedule) επιτρέπονται αλλαγές στη σειρά διέλευσης των εργασιών στις μηχανές. Επομένως, είναι πιθανό η σειρά εκτέλεσης των εργασιών να διαφέρει από μηχανή σε μηχανή. Κατά συνέπεια, για να περιγραφεί ένα μη μεταθετικό πρόγραμμα απαιτείται από μία διάταξη των N εργασιών για κάθεμια από τις M μηχανές. Άρα υπάρχουν $(N!)^M$ μη μεταθετικά προγράμματα.

Υπάρχουν, όμως, δύο σημαντικά θεωρήματα, τα οποία περιορίζουν το πλήθος των μη μεταθετικών προγραμμάτων. Οι αποδείξεις βρίσκονται στους Baker [Bak74] και Conway [CMM67].

Θεώρημα 1

- Στο πρόβλημα ροϊκής παραγωγής, σε κάθε βέλτιστο πρόγραμμα ως προς οποιοδήποτε κανονικό κριτήριο, διατηρείται η ίδια διάταξη των εργασιών στις δύο πρώτες μηχανές του συστήματος.

Θεώρημα 2

- Στο πρόβλημα ροϊκής παραγωγής, σε κάθε πρόγραμμα που ελαχιστοποιεί το συνολικό χρόνο εκτέλεσης C_{max}, διατηρείται η ίδια διάταξη των εργασιών στις δύο πρώτες μηχανές και η ίδια διάταξη στις δύο τελευταίες μηχανές του συστήματος.

Από τα παραπάνω θεωρήματα εξάγονται δύο συμπεράσματα :

1. Στο πρόβλημα $N\,/\,2\,/\,F,\ Seq-Ind\,/\,/$ το βέλτιστο πρόγραμμα ως προς οποιοδήποτε κανονικό κριτήριο είναι μεταθετικό [GW64] .
2. Στο πρόβλημα $N\ /\ 3\ /\ F,\ Seq-Ind\ /\ C_{max}$ το βέλτιστο πρόγραμμα ως προς το συνολικό χρόνο εκτέλεσης των εργασιών είναι μεταθετικό [Joh54] .

Οι ανομοιομορφίες των χρόνων εξάρμωσης και επεξεργασίας των εργασιών είναι η αιτία που τα μεταθετικά προγράμματα εμφανίζουν νεκρούς χρόνους στις μηχανές. Αντίθετα, η δυνατότητα που παρέχουν τα μη μεταθετικά προγράμματα για αλλαγή της σειράς εκτέλεσης των εργασιών στις μηχανές έχει ως αποτέλεσμα τη σπανιότερη

εμφάνιση νεκρών χρόνων. Για το λόγο αυτό οι μη μεταθετικές λύσεις του προβλήματος ροϊκής παραγωγής υπερτερούν των μεταθετικών.

Στην παρούσα εργασία, όπως και στο πλείστον της βιβλιογραφίας, μας απασχολούν μόνο μεταθετικές λύσεις. Τρεις λόγοι μας ωθούν στην επιλογή αυτή.

- Αν και οι μη μεταθετικές λύσεις υπερτερούν των μεταθετικών, έχουμε λόγους να πιστεύουμε ότι η καλύτερη μεταθετική λύση δεν είναι πολύ χειρότερη από τη βέλτιστη. Έστω μια εργασία j, η οποία εκτελείται πρώτη στη μηχανή k. Αν στη μηχανή $k-1$ εκτελούνται i άλλες εργασίες πριν την εργασία j, τότε αυτές οι i εργασίες για να εκτελεσθούν στη μηχανή k πρέπει να περιμένουν μέχρι να ολοκληρωθεί η επεξεργασία της j και στη μηχανή $k-1$ και στη μηχανή k. Η μετατόπιση της εργασίας j από τη θέση $i+1$ στην πρώτη θέση, είναι φανερό ότι καθυστερεί τη ροή της παραγωγής. Επομένως, φαίνεται λογική η υπόθεση ότι ένα μεταθετικό πρόγραμμα που διατηρεί αμετάβλητη τη διάταξη των εργασιών στις μηχανές $k-1$ και k, θα δώσει λύση εξίσου καλή με το μη μεταθετικό πρόγραμμα.

- Το σύνολο των εφικτών μεταθετικών λύσεων είναι πολύ μικρότερο από το αντίστοιχο σύνολο των μη μεταθετικών λύσεων. Αν και η εξέταση όλων των μεταθετικών λύσεων είναι πρακτικά αδύνατη, το περιορισμένο πλήθος τους διευκολύνει την αναζήτηση της βέλτιστης.

- Τα μη μεταθετικά προγράμματα απαιτούν αναδιατάξεις των εργασιών από μηχανή σε μηχανή. Πολύ συχνά, όμως, η δομή των ουρών αποθήκευσης των ενδιάμεσων προϊόντων στις μηχανές δεν επιτρέπει τέτοιες αναδιατάξεις. Για παράδειγμα, στις γραμμές παραγωγής με μεγάλο βαθμό αυτοματοποίησης οι ουρές αποθήκευσης είναι τύπου FIFO, με αποτέλεσμα η ροή παραγωγής να παίρνει αναγκαστικά τη μορφή μεταθετικού προγράμματος. Στην πράξη, επίσης, υπάρχουν περιορισμοί στο πλήθος των ενδιάμεσων προϊόντων που μπορούν να κρατηθούν στις ουρές αποθήκευσης των μηχανών, με αποτέλεσμα ν' αποφεύγονται οι αναδιατάξεις των εργασιών και να προτιμώνται μεταθετικά προγράμματα.

2.5 Προθεσμίες εργασιών

Πολλές φορές είναι απαραίτητο η εκτέλεση μιας εργασίας να έχει ολοκληρωθεί πριν από κάποια χρονική στιγμή. Απαιτήσεις αυτής της μορφής προέρχονται από το χώρο της

βιομηχανικής παραγωγής, όπου τα παραγόμενα προϊόντα συνήθως έχουν συγκεκριμένες προθεσμίες παράδοσης, οι οποίες καθορίζονται από τις παραγγελίες των πελατών. Σε ορισμένες περιπτώσεις η εκπρόθεσμη παραγωγή ενός προϊόντος έχει ως αποτέλεσμα την ακύρωση της αντίστοιχης παραγγελίας. Είναι, λοιπόν, φανερό ότι ένας από τους κύριους στόχους του προγραμματισμού παραγωγής είναι η έγκαιρη εκτέλεση των εργασιών.

Αν ο συνολικός χρόνος εκτέλεσης μιας εργασίας i υπερβαίνει την προθεσμία της dd_i τότε η εργασία έχει καθυστέρηση T_i , η οποία ισούται με $C_i - dd_i$. Στην αντίθετη περίπτωση, η εργασία ολοκληρώνεται μέσα στα χρονικά περιθώρια της και δεν υπάρχει καθυστέρηση. Συνεπώς ισχύει:

$$T_i \ = \ \max(C_i - dd_i, \ 0) \tag{2.9}$$

όπου $C_i \ = \ C_{iM}$ ο χρόνος περάτωσης της εργασίας i.

Το πρόβλημα της ροϊκής παραγωγής με προθεσμίες δεν έχει μελετηθεί αναλυτικά στη βιβλιογραφία. Αυτό οφείλεται σε δύο κυρίως λόγους :

- Η πρωτοπόρος δουλειά του Johnson [Joh54] και η επιλογή του χρόνου περάτωσης προγράμματος C_{max} ως κριτηρίου βελτίστου επηρέασε πολλούς μετέπειτα ερευνητές ώστε να υιοθετήσουν το ίδιο κριτήριο βελτίστου. Να σημειωθεί ότι το κριτήριο C_{max} είναι το ευκολότερο από τα κριτήρια βελτίστου που έχουν χρησιμοποιηθεί [DPS92].

- Ο προγραμματισμός εργασιών με προθεσμίες παράδοσης παρουσιάζεται εξαιρετικά πολύπλοκος λόγω της δυσκολίας ανάθεσης προθεσμιών στις ενδιάμεσες κατεργασίες.

  Προκειμένου να προγραμματιστεί η χρονική στιγμή έναρξης μιας εργασίας στην πρώτη μηχανή του συστήματος, είναι αναγκαίο να προσδιοριστούν οι προθεσμίες της για κάθεμια μηχανή της γραμμής παραγωγής. Αυτό, όμως, δεν είναι δυνατό λόγω της εμφάνισης νεκρών χρόνων στις μηχανές, οι οποίοι δεν είναι γνωστοί και δεν μπορούν να υπολογιστούν εκ των προτέρων, καθώς εξαρτώνται από τη σειρά εκτέλεσης των εργασιών.

2.6 Συμβολισμοί

Ακολουθεί ο κατάλογος με τους ορισμούς των μεγεθών και των συμβόλων που χρησιμοποιούνται στην εργασία.

N : αριθμός εργασιών

M : αριθμός μηχανών

Π : πρόγραμμα εργασιών

p_{ij} : χρόνος επεξεργασίας της εργασίας i στη μηχανή j

s_{ij} : χρόνος εξάρμωσης της μηχανής j για την εκτέλεση της εργασίας i

I_{ij} : νεκρός χρόνος στη μηχανή j πριν την εκτέλεση της εργασίας i

C_{ij} : χρόνος περάτωσης της εργασίας i στη μηχανή j

C_i : χρόνος περάτωσης της εργασίας i

G_i : ομάδα στην οποία ανήκει η εργασία i

B : αριθμός διαφορετικών ομάδων εργασιών

dd_i : προθεσμία της εργασίας i

T_i : καθυστέρηση της εργασίας i

Κεφάλαιο 3

Ανασκόπηση αλγορίθμων

3.1 Ο αλγόριθμος του Johnson για δύο μηχανές

Το πρόβλημα της ροϊκής παραγωγής αντιμετωπίστηκε για πρώτη φορά από τον Johnson το 1954 [Joh54]. Ο Johnson θεώρησε N εργασίες που πρέπει να εκτελεστούν σε δύο μηχανές ώστε να ελαχιστοποιηθεί ο συνολικός χρόνος επεξεργασίας τους. Υποθέτοντας ότι οι χρόνοι εξάρμωσης των μηχανών είναι ανεξάρτητοι ακολουθίας και υποχρεωτικοί για κάθε εργασία ο Johnson τους ενσωμάτωσε στους χρόνους επεξεργασίας των εργασιών.

Ο Johnson απέδειξε ότι η βέλτιστη λύση του προβλήματος $N\ /2\ /F\ /C_{max}$ δίνεται από ένα μεταθετικό πρόγραμμα, στο οποίο μια εργασία i προηγείται μιας εργασίας j μόνο αν

$$\min(p_{i1},\ p_{j2}) \ \leq \ \min(p_{j1},\ p_{i2}) \tag{3.1}$$

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 την επεξεργασία τους. Οι εργασίες που έχουν τους μικρότερους χρόνους επεξεργασίας στη δεύτερη μηχανή, τοποθετούνται στο τέλος του προγράμματος, ώστε η εκτέλεσή του να ολοκληρωθεί γρήγορα. Χρησιμοποιώντας τον κανόνα αυτό ο Johnson διατύπωσε τον παρακάτω πολυωνυμικό αλγόριθμο, ο οποίος δίνει βέλτιστες λύσεις για το πρόβλημα των δύο μηχανών.

Βήμα 1 Προσδιόρισε τον ελάχιστο χρόνο επεξεργασίας από τους χρόνους εκτέλεσης των εργασιών.

Βήμα 2 Αν ο χρόνος αυτός ανήκει σε κατεργασία της πρώτης μηχανής τοποθέτησε την αντίστοιχη εργασία στην αρχή του προγράμματος. Αν ανήκει σε κατεργασία της δεύτερης μηχανής τοποθέτησε την αντίστοιχη εργασία στο τέλος του προγράμματος.

Βήμα 3 Προσδιόρισε τον ελάχιστο χρόνο επεξεργασίας από τους χρόνους εκτέλεσης των εργασιών που δεν έχουν ακόμα προγραμματιστεί.

Βήμα 4 Αν ο χρόνος αυτός ανήκει σε κατεργασία της πρώτης μηχανής τοποθέτησε την αντίστοιχη εργασία στην πρώτη ελεύθερη θέση από την αρχή του προγράμματος. Αν ανήκει σε κατεργασία της δεύτερης μηχανής τοποθέτησε την αντίστοιχη εργασία στην πρώτη ελεύθερη θέση από το τέλος του προγράμματος. Σε περίπτωση ισότητας χρόνων η επιλογή της εργασίας είναι αυθαίρετη.

Βήμα 5 Αν υπάρχουν άλλες απρογραμμάτιστες εργασίες πήγαινε στο βήμα 3, αλλιώς τέλος της διαδικασίας.

Στο πρόβλημα $N\ /3\ /F\ /C_{max}$, όπου ο προγραμματισμός των εργασιών γίνεται σε τρεις 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. Στην περίπτωση αυτή η δεύτερη μηχανή δεν είναι η κρισιμότερη, ως προς το φόρτο εργασίας, μηχανή της γραμμής παραγωγής. Επομένως, αν ισχύει ότι

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

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

$$\min(p_{i1}+p_{i2},\ p_{j3}+p_{j2})\ \leq\ \min(p_{j1}+p_{j2},\ p_{i3}+p_{i2}) \tag{3.2}$$

Να σημειωθεί ότι η σχέση (3.2) προκύπτει από τη σχέση (3.1) αν θέσουμε

$$p'_{k1}\ =\ (p_{k1}+p_{k2}) \quad \text{και} \quad p'_{k2}\ =\ (p_{k3}+p_{k2}).$$

Οι Burns και Rooker έδειξαν το 1978 [BR78] ότι το πρόβλημα προγραμματισμού N εργασιών σε τρεις μηχανές λύνεται βέλτιστα από τον αλγόριθμο του Johnson και στην περίπτωση που κάθε εργασία ικανοποιεί τη συνθήκη :

$$p_{i2}\ \leq\ \min(p_{i1},\ p_{i3})$$

3.2 Επέκταση σε Μ μηχανές

Η πρωτοπόρος εργασία του Johnson έδωσε ώθηση στην έρευνα του γενικότερου προβλήματος προγραμματισμού N εργασιών σε M μηχανές κατά βέλτιστο τρόπο. Η επιλογή από τον Johnson ως κριτηρίου βελτίστου του συνολικού χρόνου εκτέλεσης C_{max} των εργασιών οδήγησε πολλούς ερευνητές να υιοθετήσουν το ίδιο κριτήριο. Τα πρώτα αποτέλεσματα της έρευνας του γενικότερου προβλήματος ροϊκής παραγωγής ήταν οι αλγόριθμοι συνδυαστικής βελτιστοποίησης, οι οποίοι αναπτύχθηκαν το 1964 από τους Dudek και Teuton [DT64] και το 1967 από τους Smith και Dudek [SD67]. Την ίδια εποχή οι Giglio και Wagner [GW64] διατύπωσαν το πρόβλημα προγραμματισμού N εργασιών σε τρεις μηχανές ως πρόβλημα δυαδικού ακέραιου προγραμματισμού, στο οποίο κάθε δυαδική μεταβλητή Z_{ij} ορίζεται να παίρνει την τιμή 1 αν η εργασία i βρίσκεται στη θέση j της διάταξης. Σε κάθε άλλη περίπτωση η τιμή της Z_{ij} είναι 0. Ως αντικειμενική συνάρτηση ορίζεται ο συνολικός νεκρός χρόνος της τελευταίας μηχανής του συστήματος, η ελαχιστοποίηση του οποίου ισοδυναμεί με την ελαχιστοποίηση του χρόνου περάτωσης του προγράμματος. Η μέθοδος αυτή, όμως, δεν μπόρεσε να εφαρμοστεί στην πράξη, καθώς απαιτείται μεγάλο πλήθος περιορισμών και μεταβλητών ακόμη και για μικρού μεγέθους προβλήματα.

Παρά τη σημαντική προσπάθεια κανένα ουσιαστικό βήμα δεν πραγματοποιήθηκε προς την επέκταση των αποτελεσμάτων της εργασίας του Johnson στο γενικότερο πρόβλημα ροϊκής παραγωγής. Η ανάπτυξη της θεωρίας υπολογιστικής πλοκής επιβεβαίωσε το συμπέρασμα ότι το γενικότερο πρόβλημα είναι πιο πολύπλοκο από αυτό των δύο μηχανών. Τόσο το πρόβλημα $N/\ M/F,\ Seq-Ind,\ perm/C_{max}$ όσο και το πρόβλημα $N/\ M/F,\ Seq-Ind,\ perm/T_{max},\ C_{max}$ αποδείχθηκε ότι ανήκουν στην κατηγορία των *NP*-πλήρων προβλημάτων. Η απόδειξη για την περίπτωση προγραμμάτων χωρίς διακοπές (nonpreemtive) έγινε το 1976 από τους Garey, Johnson και Sethi [GJS76], ενώ για την περίπτωση των προγραμμάτων με διακοπές (preemtive) έγινε το 1978 από τους Gonzalez και Sahni [GS78]. Επίσης, το 1978 οι Bruno και Downey [BD78] απέδειξαν ότι, στην περίπτωση που οι εργασίες έχουν προθεσμίες παράδοσης, τα προβλήματα εύρεσης εφικτού προγράμματος ($T_{max} = 0$) και εύρεσης εφικτού προγράμματος που να ελαχιστοποιεί το C_{max} είναι *NP*-πλήρη. Ο χαρακτηρισμός του προβλήματος ροϊκής παραγωγής ως δυσεπίλητο (intractable), εκτός ειδικών περιπτώσεων, είχε ως αποτέλεσμα η ερευνητική προσπάθεια να στραφεί πλέον στην ανάπτυξη ευρηματικών αλγορίθμων για την εύρεση προσεγγιστικών λύσεων.

Η πιο σημαντική προσπάθεια ανάπτυξης μιας γενικής μεθόδου για την εύρεση υποβέλτιστων λύσεων ήταν η χρήση της τεχνικής διαμερισμού και φραγής (branch and bound). Οι πρώτοι αλγόριθμοι, που χρησιμοποιούσαν αυτή την τεχνική για τη λύση του προβλήματος ροϊκής παραγωγής, αναπτύχθηκαν, ανεξάρτητα, από τους Ignall και Schrage [IS65] και Lomnicki [Lom65]. Κρίσιμο σημείο στην απόδοση των αλγορίθμων διαμερισμού και φραγής αποτελεί ο υπολογισμός φραγμάτων για τις λύσεις, τα οποία περιορίζουν το χώρο αναζήτησης. Η έρευνα για την ανάπτυξη μηχανισμών υπολογισμού σφικτών (sharp) φραγμάτων για τις λύσεις παρουσίασε ενδιαφέροντα αποτελέσματα για μικρού μεγέθους προβλήματα [MB67, Gup70, Szw73, Szw78, GR78, GSS87, Kim93]. Τα αποτελέσματα αυτά, όμως, δεν μπόρεσαν να επεκταθούν σε μεγάλου μεγέθους προβλήματα. Η εξαγωγή ιδιοτήτων των βέλτιστων λύσεων του προβλήματος στη γενική μορφή του αποδείχθηκε εξαιρετικά δύσκολη λόγω της πολυπλοκότητας του. Η εξάρτηση της δυσκολίας του προβλήματος από τους χρόνους επεξεργασίας των εργασιών και τους νεκρούς χρόνους των μηχανών περιορίζει σημαντικά τις δυνατότητες υπολογισμού αποδοτικών φραγμάτων [Ash70, HC84]. Η ανάπτυξη παράλληλων αλγορίθμων διαμερισμού και φραγής δεν έδωσε θεαματικά αποτελέσματα, αν και αντιμετώπισε αρκετές αδυναμίες της μεθόδου [JGAM88, TO89].

3.3 Ευρηματικές λύσεις

3.3.1 Ο αλγόριθμος του Palmer

Το 1965 ο Palmer δημοσίευσε έναν ευρηματικό αλγόριθμο για την εύρεση υποβέλτιστων λύσεων του προβλήματος προγραμματισμού N εργασιών σε M μηχανές με κριτήριο το χρόνο περάτωσης C_{max} του προγράμματος [Pal65]. Ο αλγόριθμος του Palmer προγραμματίζει τις εργασίες με βάση τις διαφορές των χρόνων επεξεργασίας τους στις μηχανές του συστήματος. Για το λόγο αυτό, αρχικά υπολογίζεται ένας δείκτης κλίσης (slope index) για κάθε εργασία και στο τελικό πρόγραμμα οι εργασίες είναι διατεταγμένες σε φθίνουσα σειρά ως προς το δείκτη αυτό. Ο δείκτης κλίσης S_i μιας εργασίας i υπολογίζεται από τη σχέση :

$$S_i = -\sum_{j=1}^{M}(M-2j+1)p_{ij}$$

3.3.2 Ο αλγόριθμος του Dannenbring

Ο αλγόριθμος που προτάθηκε από τον Dannenbring [Dan77] λειτουργεί σε δύο φάσεις. Στην πρώτη φάση δημιουργείται ένα τεχνητό πρόβλημα δύο μηχανών, το οποίο λύνεται με τον αλγόριθμο του Johnson. Στη δεύτερη φάση βελτιώνεται η αρχική λύση με μια επαναληπτική διαδικασία αντιμετάθεσης γειτονικών εργασιών. Ο αλγόριθμος του Dannenbring ανήκει στη γενική κατηγορία των αλγορίθμων τοπικής αναζήτησης (local search optimization), καθώς με ευρηματικό τρόπο παράγει μια αρχική λύση, την οποία βελτιώνει στη συνέχεια.

Στην ενότητα αυτή, θα μας απασχολήσει η πρώτη φάση του αλγορίθμου. Στη φάση αυτή το γενικό πρόβλημα ανάγεται σε πρόβλημα δύο μηχανών υπολογίζοντας για κάθε εργασία δύο σταθμισμένα αθροίσματα των χρόνων επεξεργασίας της, τα οποία αποτελούν τους νέους χρόνους επεξεργασίας στο βοηθητικό πρόβλημα. Οι νέοι χρόνοι υπολογίζονται από τις παρακάτω σχέσεις :

$$p'_{i1} = \sum_{j=1}^{M} (M - j + 1) p_{ij}$$

και

$$p'_{i2} = \sum_{j=1}^{M} j p_{ij}$$

όπου p'_{i1} και p'_{i2} οι χρόνοι επεξεργασίας της εργασίας i στην πρώτη και δεύτερη μηχανή του βοηθητικού προβλήματος.

3.3.3 Ο αλγοριθμος της ελάχιστης παρεμβολής

Ο αλγόριθμος της ελάχιστης παρεμβολής είναι ευρηματικός και αναπτύχθηκε για τη λύση του γνωστού συνδυαστικού προβλήματος του πλανόδιου πωλητή (Traveling Salesman Problem). Η συνδυαστική φύση των προβλημάτων χρονικού προγραμματισμού και οι ομοιότητες που παρουσιάζουν με το πρόβλημα του πλανόδιου πωλητή [LLKS85] επιτρέπουν την εφαρμογή του αλγορίθμου ελάχιστης παρεμβολής στο πρόβλημα ροϊκής παραγωγής.

Ο αλγόριθμος κατασκευάζει σταδιακά το πρόγραμμα εκτέλεσης των εργασιών

στις μηχανές του συστήματος, με κριτήριο την ελάχιστη δυνατή αύξηση του χρόνου περάτωσης C_{max} του προγράμματος σε κάθε βήμα. Ακολουθεί η περιγραφή του αλγορίθμου :

Βήμα 1 Έστω a εργασία για την οποία ισχύει :

$$\sum_{j=1}^{N} p_{aj} \;=\; \min_{1 \le i \le N} \; (\sum_{j=1}^{M} p_{ij})$$

Τοποθέτησε την εργασία a στην αρχή του προγράμματος, για να σχηματίσεις το μερικό πρόγραμμα Π^1 με μήκος 1. Θέσε $k \;=\; 1$.

Βήμα 2 Όσο $k \;\le\; N$, για καθεμία εργασία j , από τις $N - k + 1$ που δεν έχουν προγραμματιστεί ακόμη, κάνε τα εξής :

1. Πάρε την εργασία j και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις του μερικού προγράμματος Π^{k-1} μήκους $k - 1$. Από τα k παραγόμενα προγράμματα Π^k_λ , με $\lambda \;=\; 1, \ldots, k$ (στο μερικό πρόγραμμα Π^k_λ με μήκος k η εργασία j βρίσκεται στη θέση λ) διάλεξε αυτό με το μικρότερο C_{max} .
2. Θέσε $\Pi^k \;=\; \Pi^k_\lambda$ και $k \;=\; k + 1$.

Έξοδος Δώσε ως τελική λύση το πρόγραμμα Π^N .

3.4 Γενετικοί αλγόριθμοι

Οι γενετικοί αλγόριθμοι είναι στοχαστικές μέθοδοι εξελικτικής βελτίωσης, οι οποίοι αναπτύχθηκαν και θεμελιώθηκαν θεωρητικά από τον Holland [Hol75]. Οι λειτουργία των γενετικών αλγορίθμων βασίζεται στη γενετική αναπαραγωγή ειδών, στη διασταύρωση, μετάλλαξη και επιλογή τους, διαδικασίες με τις οποίες η φύση εξέλιξε πρωτόγονους σε πολύπλοκους οργανισμούς.

Παρόλο που υπάρχουν πολλές πιθανές παραλλαγές του βασικού γενετικού αλγορίθμου που παρουσίασε ο Holland, ο υπολανθάνων μηχανισμός, ο οποίος εφαρμόζεται σε έναν πληθυσμό ειδών (individuals), παραμένει σταθερός και αποτελείται από τρεις λειτουργίες :

1. Αξιολόγηση των ειδών με κάποια κριτήρια.
2. Σχηματισμός μιας γενετικής ομάδας (gene pool).

3. Διασταύρωση (crossover) και μετάλλαξη (mutation) των ειδών της γενετικής ομάδας.

Τα είδη που προκύπτουν από τις λειτουργίες αυτές σχηματίζουν τον πληθυσμό της επόμενης γενιάς. Η διαδικασία επαναλαμβάνεται μέχρι το σύστημα να πάψει να βελτιώνεται. Στο σημείο αυτό ο πληθυσμός αποτελείται, συνήθως, από λίγα είδη, τα οποία επιβιώνουν επειδή βελτιώνονται ως προς τα κριτήρια αξιολόγησης. Ο ορισμός κριτηρίων για τα είδη είναι σημαντικός για την απόδοση των γενετικών αλγορίθμων, καθώς τα κριτήρια αυτά καθορίζουν την επιλογή και την απόρριψη των ειδών. Είδη με προοπτικές βελτίωσης συμμετέχουν με πολλαπλά αντίγραφα στη γενετική ομάδα, ενώ είδη με προοπτικές χειροτέρευσης συμμετέχουν με λιγότερα και σε ορισμένες περιπτώσεις καθόλου.

Στη συνέχεια παρουσιάζουμε το βασικό σχήμα των γενετικών αλγορίθμων :

Βήμα 1 Διάλεξε έναν επιθυμητό πληθυσμό μεγέθους $\mathcal{S}$.

Βήμα 2 Αξιολόγησε τα είδη του πληθυσμού ως προς τα κριτήρια βελτιστοποίησης.

Βήμα 3 Πιθανοκρατικά επίλεξε ορισμένα αξιόλογα είδη, για να σχηματίσεις μια γενετική ομάδα μεγέθους $\mathcal{S}$.

Βήμα 4 Όσο το μέγεθος του πληθυσμού της επόμενης γενιάς είναι μικρότερο από $\mathcal{S}$, επίλεξε τυχαία ένα γονέα από τη γενετική ομάδα. Με πιθανότητα P_1 εισάγαγε τον γονέα αυτό στον πληθυσμό της επόμενης γενιάς, ενώ με πιθανότητα $1 - P_1$ διάλεξε κι ένα δεύτερο γονέα, διασταύρωσε τους δύο γονείς και εισάγαγε τον απόγονό τους στον πληθυσμό της επόμενης γενιάς.

Βήμα 5 Με μικρή πιθανότητα P_2 επίλεξε ορισμένα είδη της επόμενης γενιάς και μετάλλαξε τα.

Βήμα 6 Αν τα κριτήρια τερματισμού ικανοποιούνται τότε δώσε ως τελική λύση τον πληθυσμό που προέκυψε από το προηγούμενο βήμα και σταμάτησε. Αλλιώς, πήγαινε στο βήμα 2.

Σχήματα γενετικών αλγορίθμων, όπως αυτό που παρουσιάσαμε αλλά και πιο σύνθετα, χρησιμοποιήθηκαν για τη λύση πληθώρας προβλημάτων [Gol89, LH89]. Οι μεγάλοι χρόνοι που απαιτεί η εκτέλεσή τους και η υπεροχή των λύσεων άλλων αλγορίθμων [Sin93] περιόρισε την χρήση τους στη λύση προβλημάτων που δεν μπορούν να αντιμετωπισθούν εύκολα με συμβατικές υπολογιστικές μεθόδους. Η εφαρμογή τους για τη

λύση συνδυαστικών προβλημάτων, όπως τα προβλήματα χρονικού προγραμματισμού που μελετάμε στην παρούσα εργασία, δεν έδωσε ικανοποιητικά αποτελέσματα [LH89]. Η μελέτη της φύσης του προβλήματος ροϊκής παραγωγής έδειξε ότι είναι ιδιαίτερα δύσκολη η αναπαράστασή του κατά τρόπο τέτοιο, ώστε γενετικοί αλγόριθμοι να μπορέσουν το λύσουν αποδοτικά.

Ο εγγενής παραλληλισμός των γενετικών αλγορίθμων [BD93, LH89] έδειξε ότι πολλές από τις αδυναμίες που παρουσιάζουν μπορούν να ξεπεραστούν με την παραλληλοποίησή τους. Ορισμένα αξιόλογα αποτελέσματα, όπως η υπεργραμμική επιτάχυνση (superlinear speedup) που παρουσίασε ο παράλληλος γενετικός αλγόριθμος των Muehlenbein, Schleuter και Kraemer [MGSK88] για τη λύση του προβλήματος του πλανόδιου πωλητή, αφήνουν πολλές υποσχέσεις για την αποδοτική εφαρμογή στο μέλλον γενετικών αλγορίθμων για τη λύση των προβλημάτων ροϊκής παραγωγής.

3.5 Simulated Annealing

Η μέθοδος simulated annealing είναι μια επαναληπτική στοχαστική μέθοδος εύρεσης υποβέλτιστων λύσεων, η οποία βασίζεται σε ιδέες της στατιστικής φυσικής και προτάθηκε για πρώτη φορά από τους Metropolis et al. [MRR$^+$53], ενώ αργότερα οι Kirkpatrick et al. [KGV83] και ο Cerny [Cer85] τη χρησιμοποίησαν για τη λύση δύσκολων συνδυαστικών προβλημάτων.

Η μέθοδος ανήκει στη γενικότερη κατηγορία μεθόδων τοπικής αναζήτησης, οι οποίες παράγουν με ευρηματικό ή τυχαίο τρόπο μιαν αρχική λύση, την οποία στη συνέχεια βελτιώνουν επαναληπτικά. Καθώς η απόδοση όλων των μεθόδων τοπικής αναζήτησης εξαρτάται από τη συχνότητα παγίδευσής τους σε τοπικά βέλτιστα, τα οποία συνήθως απέχουν πολύ από το ολικό βέλτιστο, η μέθοδος simulated annealing περιστασιακά δέχεται λύσεις, οι οποίες απομακρύνουν την αναζήτηση από τοπικά βέλτιστα. Για το σκοπό αυτό, χρησιμοποιείται σε κάθε βήμα k μια πιθανότητα P_k για την επιλογή των λύσεων, η οποία εξαρτάται από το λόγο της απόστασης ΔC μιας λύσης από κάποιο τοπικό βέλτιστο προς μια μεταβλητή ελέγχου Θ_k , η οποία παραδοσιακά ονομάζεται θερμοκρασία. Η αρχική τιμή Θ_0 της θερμοκρασίας επιλέγεται υψηλή, ώστε η πιθανότητα ο αλγόριθμος simulated annealing να επιλέγει στα πρώτα βήματά του λύσεις, οι οποίες απέχουν αρκετά από τοπικά βέλτιστα, να είναι μεγάλη. Καθώς, όμως, η θερμοκρασία Θ_k μειώνεται σε κάθε βήμα, ο αλγόριθμος σταδιακά κατευθύνεται σε κάποιο τοπικό ελάχιστο και τερματίζει όταν η θερμοκρασία πέσει κάτω από την τιμή Θ_τ $(\Theta_\tau < \Theta_0)$.

Στη συνέχεια περιγράφουμε το βασικό σχήμα του αλγορίθμου simulated annealing για το πρόβλημα εύρεσης προγράμματος εργασιών με ελάχιστο χρόνο περάτωσης C_{max} .

Είσοδος Η αρχική θερμοκρασία Θ_0, η τελική θερμοκρασία Θ_τ και ο ρυθμός μεταβολής της θερμοκρασίας $\Delta\theta_k$.

Βήμα 1 Δημιούργησε με κάποιο τρόπο την αρχική λύση Π_0.

Θέσε $\Pi_{\beta\varepsilon\lambda\tau} = \Pi_0$ και $k = 1$.

Βήμα 2 Ψάξε στην υποπεριοχή $U(\Pi_k)$ για το τοπικό ελάχιστο Π_l.

Βήμα 3 Αν $C_{max}(\Pi_l) < C_{max}(\Pi_k)$ τότε θέσε $\Pi_{k+1} = \Pi_l$.

Αλλιώς θέσε $\Pi_{k+1} = \begin{cases} \Pi_l & \text{με πιθανότητα } P_k \\ \Pi_k & \text{με πιθανότητα } 1 - P_k \end{cases}$

όπου $P_k = \min(1,\ e^{-\Delta C/\Theta_k})$

και $\Delta C = C_{max}(\Pi_k) - C_{max}(\Pi_l)$.

Βήμα 4 Θέσε $\Theta_{k+1} = \Theta_k - \Delta\theta_k$

και $\Pi_{\beta\varepsilon\lambda\tau} = \min(\Pi_{\beta\varepsilon\lambda\tau},\ \Pi_k)$.

Βήμα 5 Θέσε $k = k + 1$.

Αν $\Theta_k \leq \Theta_\tau$ τότε πήγαινε στο βήμα 2.

Αλλιώς, τερμάτισε και δώσε ως τελική λύση το πρόγραμμα $\Pi_{\beta\varepsilon\lambda\tau}$.

Σχήματα simulated annealing, όπως αυτό που παρουσιάσαμε αλλά και πιο σύνθετα, αναπτύχθηκαν για τη λύση πληθώρας προβλημάτων [Kir84, AK89, JAMS89], ενώ αξιόλογες είναι οι προσπάθειες παραλληλοποίησης της μεθόδου [AC89, BL93, MGPO89]. Η εφαρμογή αλγορίθμων simulated annealing για τη λύση προβλημάτων χρονικού προγραμματισμού [LAL92] και ροϊκής παραγωγής [MSS89, OS90, MRR93] έδειξε ότι η μέθοδος υπερτερεί σε ακρίβεια πολλών ευρηματικών και γενικών μεθόδων λύσης, απαιτεί, όμως, υπερβολικά μεγάλο χρόνο για την εκτέλεσή της.

3.6 Ο αλγόριθμος του Κοπιδάκη

Το 1993 ο Κοπιδάκης [Κοπ93] πρότεινε δύο ευρηματικούς αλγόριθμους για τη λύση των προβλημάτων $N/M/F, Seq{-}Ind, perm/C_{max}$ και $N/M/F, Seq{-}Ind, perm/T_{max}, C_{max}$,

οι οποίοι χρησιμοποιούν τη μέθοδο σταδιακής κατασκευής της τελικής λύσης με παρεμβολή εργασιών. Οι αλγόριθμοι αυτοί αποτελούν επέκταση του αλγορίθμου που προτάθηκε από τον Nawasz [Naw83] για το πρόβλημα ελαχιστοποίησης του χρόνου περάτωσης C_{max} προγράμματος.

Στη συνέχεια περιγράφουμε το γενικό σχήμα του αλγορίθμου του Κοπιδάκη :

Βήμα 1 Δημιούργησε την αρχική ουρά προγραμματισμού των εργασιών

Βήμα 2 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος, για να σχηματίσεις το μερικό πρόγραμμα $\Pi^1_{\text{βελτ}}$ με μήκος 1. Θέσε $k = 2$.

Βήμα 3 Πάρε την k εργασία της ουράς προγραμματισμού και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις του μερικού προγράμματος $\Pi^{k-1}_{\text{βελτ}}$ μήκους $k-1$. Από τα k παραγόμενα εφικτά προγράμματα διάλεξε αυτό με το μικρότερο C_{max} και ονόμασέ το $\Pi^k_{\text{βελτ}}$.

Βήμα 4 Αν $k < N$ τότε θέσε $k = k+1$ και εκτέλεσε το βήμα 3. Αλλιώς πήγαινε στο βήμα 5.

Βήμα 5 Δώσε ως τελική λύση το $\Pi^N_{\text{βελτ}}$.

Για να βελτιώσει την ποιότητα των λύσεων που παράγονται από τον αλγόριθμο, ο Κοπιδάκης ανέπτυξε δύο μηχανισμούς βελτίωσης των λύσεων : το μηχανισμό αποθήκευσης συνόλου μερικών διατάξεων, για να μπορεί ο αλγόριθμος να αναθεωρεί αποφάσεις προηγουμένων βημάτων και το μηχανισμό επαναληπτικής παρεμβολής με τυχαίες αρχικές ουρές προγραμματισμού, για να μπορεί ο αλγόριθμος να εξετάζει μεγάλο αριθμό λύσεων. Ο δεύτερος μηχανισμός εφαρμόστηκε μόνο στο πρόβλημα $N/M/F, Seq-Ind, perm/C_{max}$, ενώ στο πρόβλημα $N/M/F, Seq-Ind, perm/T_{max}, C_{max}$ η αρχική ουρά προγραμματισμού σχηματίζεται με τη διάταξη των εργασιών σε αύξουσα σειρά των προθεσμιών τους (EDD). Η ενσωμάτωση των νέων αυτών μηχανισμών στους αλγορίθμους και η πειραματική τους αξιολόγηση έδειξε ότι βελτιώνουν σημαντικά τις λύσεις που παράγονται.

Κεφάλαιο 4

Τοπική αναζήτηση με απαγόρευση κινήσεων

4.1 Εισαγωγή

Η μέθοδος τοπικής αναζήτησης εφαρμόζεται σε προβλήματα βελτιστοποίησης, τα οποία έχουν τη μορφή

$$\min\ f(x), \quad x \in \mathcal{S}$$

όπου $\mathcal{S}$ ένα μεγάλου μεγέθους πεπερασμένο σύνολο, το οποίο αποτελεί το χώρο αναζήτησης της βέλτιστης λύσης. Για να εφαρμόσουμε μεθόδους τοπικής αναζήτησης στα δύο προβλήματα ροϊκής παραγωγής που μελετάμε στην παρούσα εργασία, είναι απαραίτητο να τα εκφράσουμε στην παραπάνω μορφή. Στην περίπτωση του προβλήματος ελαχιστοποίησης του χρόνου περάτωσης C_{max} η αντικειμενική συνάρτηση f, η τιμή της οποίας ζητείται να ελαχιστοποιηθεί, είναι ο χρόνος αυτός, ενώ ο χώρος αναζήτησης είναι το σύνολο όλων των μεταθετικών προγραμμάτων. Οπότε, το πρόβλημα $N/\,M/\,F,\ Seq-Ind,\ perm/\,C_{max}$ ορίζεται ως εξής :

$$\min\ C_{max}(\Pi), \quad \Pi \in \mathcal{S}_N \tag{4.1}$$

όπου $\mathcal{S}_N\ =\ \{$ Τα $N!$ μεταθετικά προγράμματα N εργασιών $\}$.

Όταν οι εργασίες έχουν προθεσμίες παράδοσης τότε ο χώρος αναζήτησης περιορίζεται στο σύνολο των εφικτών προγραμμάτων και ζητείται αυτό με τον ελάχιστο χρόνο περάτωσης. Επομένως, το πρόβλημα $N/\,M/\,F,\ Seq-Ind,\ perm/\,T_{max},\ C_{max}$ ορίζεται ως

εξής :

$$\min\ C_{max}(\Pi), \quad T_{max}(\Pi)\ =\ 0 \tag{4.2}$$

Ο περιορισμός $T_{max} = 0$ προϋποθέτει ότι η εύρεση εφικτού προγράμματος είναι πάντοτε δυνατή. Υπάρχουν, όμως, περιπτώσεις όπου ο περιορισμός αυτός δεν μπορεί να ικανοποιηθεί επειδή είτε δεν υπάρχει εφικτό πρόγραμμα είτε η μέθοδος τοπικής αναζήτησης αδυνατεί να το εντοπίσει. Για το λόγο αυτό ο περιορισμός χαλαρώνεται και το πρόβλημα αναλύεται σε δύο επιμέρους προβλήματα. Αρχικά επιδιώκεται η ελαχιστοποίηση της μέγιστης καθυστέρησης των εργασιών και στη συνέχεια η ελαχιστοποίηση του χρόνου περάτωσης προγράμματος. Έχουμε, λοιπόν :

$$\min\ T_{max}(\Pi), \quad \Pi \in \mathcal{S}_N \tag{4.3}$$

και στη συνέχεια :

$$\min\ C_{max}(\Pi), \quad T_{max}(\Pi) \leq T^*_{max} \tag{4.4}$$

όπου T^*_{max} η λύση του 4.3 . Να σημειωθεί ότι η απαίτηση για την επίλυση της 4.3 πρώτα και της 4.4 έπειτα, ιεραρχεί τα δύο κριτήρια βελτίστου που χρησιμοποιούνται. Έτσι, πρωτεύον κριτήριο αποτελεί η ελαχιστοποίηση της μέγιστης καθυστέρησης των εργασιών, ενώ δευτερεύον κριτήριο είναι η ελαχιστοποίηση του χρόνου περάτωσης προγράμματος.

Για λόγους απλότητας, στο υπόλοιπο κεφάλαιο θα περιορίσουμε τη μελέτη της μεθόδου τοπικής αναζήτησης στο πρόβλημα $N/\ M/\ F,\ Seq-Ind,\ perm/\ C_{max}$. Ο όμοιος τρόπος, με τον οποίο η μέθοδος αντιμετωπίζει τα δύο προβλήματα (σχέσεις 4.1, 4.3-4), το επιτρέπει. Με το πρόβλημα $N/\ M/\ F,\ Seq-Ind,\ perm/\ T_{max},\ C_{max}$ θα ασχοληθούμε στην ενότητα 4.9, όπου θα περιγράψουμε ένα γενικό αλγόριθμο τοπικής αναζήτησης για την επίλυσή του.

4.2 Τοπική αναζήτηση

Η μέθοδος της τοπικής αναζήτησης (local search) είναι μια ευρηματική μέθοδος εύρεσης υποβέλτιστων λύσεων. Η λειτουργία της βασίζεται στη σταδιακή βελτίωση κάποιας αρχικής λύσης, η οποία παράγεται με τυχαίο ή ευρηματικό τρόπο. Αν Π_0 είναι το πρόγραμμα που αποτελεί την αρχική λύση, τότε κατά την αναζήτηση εξετάζονται τα προγράμματα μιας περιοχής $U(\Pi_0)$ του χώρου $\mathcal{S}_N$, για να επιλεγεί ένα πρόγραμμα

Π_1 με την ιδιότητα $C_{max}(\Pi_1) < C_{max}(\Pi_0)$. Η περιοχή $U(\Pi_0)$ ονομάζεται γειτονιά 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 γειτονιών και η επιλογή του ελαχίστου καθεμίας έχει ως αποτέλεσμα το σχηματισμό μιας ακολουθίας $\{\Pi_k\}$ προγραμμάτων, όπου $\Pi_{k+1} \in U(\Pi_k)$ και $C_{max}(\Pi_{k+1}) < C_{max}(\Pi_k)$. Η εκτέλεση της μεθόδου τερματίζεται με τον εντοπισμό ενός τοπικού ελαχίστου Π^* του χώρου $\mathcal{S}_N$, οπότε η αναζήτηση δεν μπορεί πλέον να συνεχιστεί καθώς $C_{max}(\Pi^*) \leq C_{max}(\Pi)$ για κάθε $\Pi \in U(\Pi^*)$. Το τοπικό ελάχιστο Π^* αποτελεί και την τελική λύση της τοπικής αναζήτησης.

Μπορούμε, τώρα, να περιγράψουμε το βασικό αλγοριθμικό σχήμα της μεθόδου τοπικής αναζήτησης, το οποίο θα επεκτείνουμε αργότερα.

Βήμα 1 Δημιούργησε με τυχαίο ή ευρηματικό τρόπο ένα αρχικό πρόγραμμα εργασιών. Ονόμασε το πρόγραμμα αυτό Π_0.

Βήμα 2 Θέσε $\Pi_{\text{βελτ}} = \Pi_0$ και $k = 1$.

Βήμα 3 Βρες το $\Pi_l \in U(\Pi_k)$, όπου $C_{max}(\Pi_l) = \min\{ C_{max}(\Pi), \quad \Pi \in U(\Pi_k)\}$ Αν το πρόγραμμα Π_l δεν είναι μοναδικό διάλεξε κάποιο με τυχαίο τρόπο.

Βήμα 4 Αν $C_{max}(\Pi_l) < C_{max}(\Pi_{\text{βελτ}})$ τότε θέσε $\Pi_{\text{βελτ}} = \Pi_l$.
Αλλιώς τερμάτισε.

Βήμα 5 Θέσε $k = k + 1$ και πήγαινε στο βήμα 3.

Έξοδος Δώσε ως τελική λύση το πρόγραμμα $\Pi_{\text{βελτ}}$.

Η απόδοση του αλγορίθμου της τοπικής αναζήτησης που παρουσιάσαμε εξαρτάται από τέσσερις παράγοντες : τον τρόπο παραγωγής της αρχικής λύσης, τη μέθοδο σχηματισμού των γειτονιών, το πλήθος των ελαχίστων που περιέχονται σε αυτές και τέλος, το κριτήριο τερματισμού του αλγορίθμου.

Ο αλγόριθμος της τοπικής αναζήτησης βελτιώνει την αρχική λύση, την οποία παράγει στο πρώτο βήμα, εξετάζοντας διαδοχικές γειτονιές προγραμμάτων. Επειδή, όμως, τα προγράμματα που ανήκουν σε μια γειτονιά δε διαφέρουν σημαντικά μεταξύ τους, η βελτίωση που επιτυγχάνεται σε κάθε επανάληψη είναι συνήθως μικρή. Για το λόγο αυτό, η μέθοδος με την οποία παράγεται η αρχική λύση του αλγορίθμου είναι σημαντική, καθώς η ποιότητα της αρχικής λύσης καθορίζει σε μεγάλο βαθμό την ποιότητα της τελικής λύσης.

Η μέθοδος με την οποία σχηματίζονται οι γειτονιές αναζήτησης και ειδικά το μέγεθός τους, είναι καθοριστικό για την απόδοση του αλγορίθμου. Όσο μεγαλύτερο είναι το μέγεθος μιας γειτονιάς τόσο απαιτητικότερη σε υπολογιστικό χρόνο γίνεται η αναζήτηση αλλά και τόσο καλύτερη η τελική λύση που παράγεται. Επομένως, η επιλογή της γειτονιάς πρέπει να εξασφαλίζει την εξέταση ενός ικανοποιητικού αριθμού προγραμμάτων χωρίς να επιβαρύνεται χρονικά η εκτέλεση του αλγορίθμου.

Στην περίπτωση που το ελάχιστο μιας γειτονιάς δεν είναι μοναδικό επιλέγεται αυθαίρετα ένα από αυτά για τη συνέχιση της αναζήτησης. Η τυχαία επιλογή κατεύθυνσης δεν εξασφαλίζει ότι η αναζήτηση συνεχίζεται με τον καλύτερο δυνατό τρόπο. Είναι, λοιπόν, απαραίτητη η αξιολόγηση φαινομενικά ισοδύναμων κατευθύνσεων για την επιλογή της καλύτερης.

Η εκτέλεση του αλγορίθμου τερματίζεται με τον εντοπισμό τοπικού ελαχίστου. Αν και δεν μπορούμε να υπολογίσουμε τον ακριβή αριθμό τοπικών ελαχίστων που υπάρχουν στο χώρο $\mathcal{S}_N$ υποθέτουμε ότι ο αριθμός τους πρέπει να είναι σημαντικός, λόγω του τεράστιου μέγεθους του χώρου αυτού. Επομένως, το τοπικό ελάχιστο που αποτελεί και την τελική λύση ενδέχεται να απέχει σημαντικά από το ολικό. Για το λόγο αυτό, είναι απαραίτητο η αναζήτηση να μην ολοκληρώνεται στο πρώτο τοπικό ελάχιστο που ενοπίζεται αλλά να συνεχίζεται εξετάζοντας το μεγαλύτερο δυνατό αριθμό τοπικών ελαχίστων.

4.3 Μέθοδοι παραγωγής αρχικών προγραμμάτων

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

Αρχικά προγράμματα μπορούν να παραχθούν είτε με τυχαίο είτε με ευρηματικό τρόπο. Στην περίπτωση που τα αρχικά προγράμματα επιλέγονται τυχαία από τα προγράμματα του χώρου $\mathcal{S}_N$, η ποιότητα των αρχικών λύσεων εξαρτάται άμεσα από το μέγεθος του χώρου αυτού. Όσο μεγαλύτερο είναι το πλήθος των προγραμμάτων που περιέχονται στο χώρο $\mathcal{S}_N$ τόσο μικρότερη είναι η πιθανότητα να επιλεγεί με τυχαίο τρόπο καλή αρχική λύση. Η αναποτελεσματικότητα της μεθόδου αυτής, ακόμα και για μετρίου μεγέθους προβλήματα ($N > 6$), έχει ως αποτέλεσμα να χρησιμοποιούνται στην πράξη

ευρηματικές μέθοδοι για την παραγωγή αρχικών προγραμμάτων. Αν και στην κατηγορία αυτή περιλαμβάνονται όλοι οι ευρηματικοί αλγόριθμοι που έχουν αναπτυχθεί για τη λύση του προβλήματος της ροϊκής παραγωγής πρακτικό ενδιαφέρον έχουν οι λιγότερο απαιτητικοί σε υπολογιστικό χρόνο.

Η πολυπλοκότητα του προβλήματος της ροϊκής παραγωγής καθιστά δύσκολη την αξιολόγηση των διάφορων ευρηματικών μεθόδων και την επιλογή της καλύτερης να χρησιμοποιηθεί. Για το λόγο αυτό στην παρούσα εργασία αξιολογήθηκαν πειραματικά πέντε μέθοδοι, τα αποτελέσματα των οποίων παρουσιάζονται στο κεφάλαιο 7. Οι τρεις πρώτες είναι οι γνωστοί αλγόριθμοι των Palmer, Dannenbring και της ελάχιστης παρεμβολής. Οι αλγόριθμοι αυτοί, που παρουσιάστηκαν στο κεφάλαιο 3, αναπτύχθηκαν για τη λύση του προβλήματος ροϊκής παραγωγής.

Για λόγους πληρότητας της μελέτης μας δοκιμάστηκαν δύο ακόμη μέθοδοι, οι οποίες σχεδιάστηκαν αποκλειστικά για την παραγωγή αρχικών προγραμμάτων : η μέθοδος των Widmer και Hertz, η οποία χρησιμοποιήθηκε για την παραγωγή αρχικών λύσεων στον αλγόριθμο τοπικής αναζήτησης SPIRIT [WH89] και η μέθοδος των νεκρών χρόνων, η οποία αναπτύχθηκε στα πλαίσια της παρούσας εργασίας. Και οι δύο μέθοδοι ανάγουν τη λύση του προβλήματος της ροϊκής παραγωγής στο συγγενές συνδυαστικό πρόβλημα του πλανόδιου πωλητή (Travelling Salesman Problem). Για το σκοπό αυτό οι εργασίες αντιστοιχίζονται σε πόλεις και η εύρεση ενός προγράμματος με ελάχιστο χρόνο περάτωσης ανάγεται στην εύρεση ενός μονοπατιού με ελάχιστο μήκος. Για να ολοκληρωθεί η αναγωγή, απαιτείται να οριστεί η απόσταση δύο εργασιών.

Οι Widmer και Hertz θεώρησαν την απόσταση δύο εργασιών ανάλογη των διαφορών που παρουσιάζουν οι χρόνοι εξάρμωσης και επεξεργασίας τους στις μηχανές του συστήματος και όρισαν την απόσταση D_{ij} δύο εργασιών i και j με την ακόλουθη σχέση :

$$D_{ij} \;=\; t_{i1} \;+\; \sum_{k=2}^{M} (M-k) \mid t_{ik} - t_{j,k-1} \mid \;+ t_{jM}$$

όπου $t_{im} = s_{im} + p_{im}$.

Στη μέθοδο των νεκρών χρόνων η απόσταση δύο εργασιών αποτελεί ένα μέτρο του νεκρού χρόνου που εμφανίζεται στην τελευταία μηχανή του συστήματος από τη διαδοχική εκτέλεση των εργασιών αυτών. Η απόσταση D_{ij} δύο εργασιών i και j ορίζεται ως ο νεκρός χρόνος I_{jM} , ο οποίος θα εμφανιστεί στην τελευταία μηχανή του συστήματος, από τη διαδοχική εκτέλεση των εργασιών i και j στις δύο πρώτες θέσεις οποιουδήποτε προγράμματος. Να υπενθυμίσουμε ότι η ελαχιστοποίηση των νεκρών

χρόνων της τελευταίας μηχανής του συστήματος ισοδυναμεί με την ελαχιστοποίηση του χρόνου περάτωσης ενός προγράμματος.

4.4 Γειτονιές αναζήτησης

Η λειτουργία της μεθόδου τοπικής αναζήτησης απαιτεί το χωρισμό του χώρου $\mathcal{S}_N$ όλων των προγραμμάτων σε υποπεριοχές. Ένας τέτοιος χωρισμός προϋποθέτει την ύπαρξη μιας σχέσης γειτονίας των προγραμμάτων, η οποία ορίζεται με την απεικόνιση $U : \mathcal{S}_N \rightarrow 2^{\mathcal{S}_N}$. Με τη σχέση αυτή, σε κάθε πρόγραμμα Π αντιστοιχίζεται ένα υποσύνολο $U(\Pi)$ προγραμμάτων, το οποίο ονομάζεται γειτονιά του Π. Οι γειτονιές περιγράφονται επιλέγοντας πρώτα ένα είδος κίνησης, με την οποία μπορούμε από ένα πρόγραμμα Π να παράγουμε ένα καινούριο. Έπειτα, η γειτονιά $U(\Pi)$ ορίζεται ως το σύνολο όλων των προγραμμάτων, τα οποία μπορούν να παραχθούν από το Π σε μία μόνο κίνηση. Δοθείσας κίνησης K μπορούμε, λοιπόν, να ορίσουμε τη γειτονιά $U(\Pi)$ ενός προγράμματος Π ως εξής :

$$U(\Pi) \;=\; \{\, \Pi' \in \mathcal{S}_N \;\mid\; \Pi' = K(\Pi) \,\}, \quad \Pi \in \mathcal{S}_N \tag{4.5}$$

Ως κίνηση θεωρούμε τη μετάθεση μιας εργασίας ενός προγράμματος σε οποιαδήποτε θέση [Eva87, Wer93]. Συγκεκριμένα ορίζουμε :

$$K_{ij}(\Pi) \;=\; \begin{cases} (1, 2, \ldots, i-1, i+1, \ldots, j, i, j+1, \ldots, N) & \text{αν } i < j \\ (1, 2, \ldots, j-1, i, j, \ldots, i-1, i+1, \ldots, N) & \text{αν } i > j \end{cases} \tag{4.6}$$

Οι παράμετροι i, j και Π της κίνησης $K_{ij}(\Pi)$ θα παραλείπονται όποτε ο καθορισμός τους δεν είναι απαραίτητος και η κίνηση θα αναφέρεται απλά ως K.

Από τον ορισμό της κίνησης K εξάγονται τρεις χρήσιμες ιδιότητες.

Ιδιότητα 1 $\quad \Pi = K_{ji}(K_{ij}(\Pi))$

Απόδειξη

Έστω η εργασία a η οποία βρίσκεται στη θέση i του προγράμματος Π. Αν $\Pi' = K_{ij}(\Pi)$ και $\Pi'' = K_{ji}(\Pi')$ θα δείξουμε ότι $\Pi'' = \Pi$.

Έστω $i < j$. Με την κίνηση $K_{ij}(\Pi)$ παράγεται το πρόγραμμα Π', στο οποίο η εργασία a από τη θέση i μεταφέρεται στη θέση j. Το αποτέλεσμα της κίνησης αυτής είναι στο πρόγραμμα Π' οι εργασίες που βρίσκονταν στις θέσεις $i+1$ έως

και j του προγράμματος Π να ολισθήσουν κατά μία θέση, ενώ όλες οι άλλες εργασίες να παραμείνουν στις αρχικές θέσεις.

Εφαρμόζοντας την κίνηση K_{ji} στο πρόγραμμα Π' παράγεται το πρόγραμμα Π'', στο οποίο η εργασία a μεταφέρεται από τη θέση j στη θέση i. Στο πρόγραμμα Π'' οι εργασίες που βρίσκονταν στις θέσεις i έως και $j-1$ του προγράμματος Π' μετατοπίζονται στις θέσεις που είχαν στο πρόγραμμα Π, ενώ οι υπόλοιπες εργασίες διατηρούν τις αρχικές τους θέσεις. Βλέπουμε, λοιπόν, ότι η διαδοχική εφαρμογή των κινήσεων $K_{ij}(\Pi)$ και $K_{ji}(\Pi')$ είχε ως αποτέλεσμα το πρόγραμμα Π'', στο οποίο όλες οι εργασίες διατηρούν τελικά τις θέσεις που είχαν στο πρόγραμμα Π. Δηλαδή, $\Pi = \Pi''$.

Με όμοιο τρόπο αποδεικνύεται η πρόταση και για $i > j$. $\square$

Ιδιότητα 2 $K_{i,i+1} = K_{i+1,i}$

Η ιδιότητα αυτή είναι άμεση συνέπεια του ορισμού της κίνησης K.

Ιδιότητα 3 Αν $K_{ij}(\Pi_1) = \Pi_2$ και $K_{jk}(\Pi_2) = \Pi_3$ τότε $K_{ik}(\Pi_1) = \Pi_3$

Απόδειξη

Για λόγους απλότητας, θα αποδείξουμε την ιδιότητα για την περίπτωση που $i < j < k$. Η απόδειξη των υπόλοιπων περιπτώσεων γίνεται με όμοιο τρόπο.

Έστω η εργασία a η οποία βρίσκεται στη θέση i του προγράμματος Π_1.

Με την κίνηση $K_{ij}(\Pi_1)$ παράγεται το πρόγραμμα Π_2, στο οποίο η εργασία a από τη θέση i μεταφέρεται στη θέση j. Το αποτέλεσμα της κίνησης αυτής είναι στο πρόγραμμα Π_2 οι εργασίες που βρίσκονταν στις θέσεις $i+1$ έως και j του προγράμματος Π_1 να ολισθήσουν κατά μία θέση, ενώ όλες οι άλλες εργασίες να παραμείνουν στις αρχικές τους θέσεις.

Εφαρμόζοντας την κίνηση K_{jk} στο πρόγραμμα Π_2 παράγεται το πρόγραμμα Π_3, στο οποίο η εργασία a μεταφέρεται από τη θέση j στη θέση k. Στο πρόγραμμα Π_3 οι εργασίες που βρίσκονταν στις θέσεις $j+1$ έως και k του προγράμματος Π_2 ολισθαίνουν κατά μία θέση, ενώ υπόλοιπες εργασίες διατηρούν τις θέσεις που είχαν στο Π_2.

Βλέπουμε, λοιπόν, ότι η διαδοχική εφαρμογή των κινήσεων $K_{ij}(\Pi_1)$ και $K_{jk}(\Pi_2)$ είχε ως αποτέλεσμα το πρόγραμμα Π_3, στο οποίο η εργασία στη θέση i του Π_1 μεταφέρθηκε στη θέση k, οι εργασίες στις θέσεις $i+1$ έως j και $j+1$ έως k ολίσθησαν κατά μία θέση λόγω των κινήσεων K_{ij} και K_{jk} αντίστοιχα, ενώ όλες οι άλλες εργασίες διατήρησαν στο πρόγραμμα Π_3 τις θέσεις που είχαν στο Π_1.

Όμως, από τον ορισμό της κίνησης K (σχέση 4.6) βλέπουμε ότι το ίδιο πρόγραμμα Π_3 θα είχε παραχθεί με μία μόνο κίνηση, την $K_{ik}(\Pi_1)$. $\square$

Με τη χρήση της κίνησης K_{ij} σχηματίζεται ένα πλήθος γειτονιών, οι σημαντικότερες από τις οποίες, περιγράφονται στη συνέχεια :

- Η γειτονιά U^{API} (Adjacent Pairwise Interchange Neighbourhood)

  Η γειτονιά αυτή δημιουργείται με την αντιμετάθεση διαδοχικών εργασιών του προγράμματος Π και ορίζεται ως εξής :

  $$U^{API}(\Pi) \;=\; \{\, K_{i,i+1}(\Pi) \in \mathcal{S}_N \;\mid\; 1 \leq i \leq N-1 \,\}$$

  Από τον ορισμό της γειτονιάς είναι φανερό ότι η γειτονιά U^{API} περιέχει $N-1$ προγράμματα.

- Η γειτονιά U^{LS} (Left Shift Neighbourhood)

  Η γειτονιά αυτή σχηματίζεται με την τοποθέτηση μιας εργασίας του προγράμματος Π σε οποιαδήποτε από τις θέσεις που προηγούνται της δικής της και ορίζεται ως εξής :

  $$U^{LS}(\Pi) \;=\; \{\, K_{i,j}(\Pi) \in \mathcal{S}_N \;\mid\; 1 \leq j < i \leq N \,\}$$

  Η γειτονιά U^{LS} περίεχει $\frac{N(N-1)}{2}$ προγράμματα.

  Από τον ορισμό της γειτονιάς βλέπουμε ότι η εργασία που βρίσκεται στην πρώτη θέση του προγράμματος μπορεί να τοποθετηθεί σε $N-1$ διαφορετικές θέσεις, η εργασία που βρίσκεται στη δεύτερη θέση σε $N-2$ και γενικά η εργασία που βρίσκεται στη θέση i μπορεί να τοποθετηθεί σε $N-i$ διαφορετικές θέσεις. Επομένως, τα διαφορετικά προγράμματα που μπορούν να παραχθούν είναι $\sum_{i=1}^{N}(N-i)$. Όμως,

  $$\sum_{i=1}^{N}(N-i) \quad = \quad \sum_{i=1}^{N} N \;-\; \sum_{i=1}^{N} i \quad = \quad \frac{N(N-1)}{2}$$

- Η γειτονιά U^{RS} (Right Shift Neighbourhood)

  Η γειτονιά αυτή σχηματίζεται με την τοποθέτηση μιας εργασίας του προγράμματος Π σε οποιαδήποτε από τις θέσεις που έπονται της δικής της και ορίζεται κατά αναλογία με τη γειτονιά U^{LS} :

  $$U^{RS}(\Pi) \;=\; \{\, K_{i,j}(\Pi) \in \mathcal{S}_N \;\mid\; 1 \leq i < j \leq N \,\}$$

Η γειτονιά U^{RS} παράγεται με όμοιο τρόπο με τη γειτονιά U^{LS} και περιέχει $\frac{N(N-1)}{2}$ προγράμματα.

- Η γειτονιά U^S (Shift Neighbourhood)

  Η γειτονιά αυτή δημιουργείται με την τοποθέτηση μιας οποιασδήποτε εργασίας του προγράμματος Π σε οποιαδήποτε θέση και ορίζεται ως εξής :

$$U^S(\Pi) \ = \ \{ \ K_{i,j}(\Pi) \in \mathcal{S}_N \ \mid \ 1 \leq i, j \leq N \ , \ i \neq j \}$$

  Η γειτονιά U^S περιέχει $N^2 - 2N + 1$ προγράμματα.

  Πράγματι, από τον ορισμό της γειτονιάς βλέπουμε ότι η εργασία που βρίσκεται στη θέση i μπορεί να τοποθετηθεί σε οποιαδήποτε από τις $i-1$ θέσεις που προηγούνται της δικής της και σε οποιαδήποτε από τις $N-i$ που έπονται. Επομένως, καθεμία από τις N εργασίες μπορεί να τοποθετηθεί σε $N-1$ συνολικά θέσεις, παράγοντας $N(N-1)$ προγράμματα. Στο πλήθος των προγραμμάτων που υπολογίσαμε συμπεριλαμβάνονται και εκείνα που παράγονται από τις κινήσεις $K_{i,i+1}$ και $K_{i+1,i}$. Οι κινήσεις αυτές αντιμεταθέτουν τις εργασίες που βρίσκονται σε διαδοχικές θέσεις του προγράμματος Π και από τη δεύτερη ιδιότητα της κίνησης K γνωρίζουμε ότι $K_{i,i+1}(\Pi) = K_{i+1,i}(\Pi)$. Καθώς $2(N-1)$ τέτοιες αντιμεταθέσεις είναι δυνατές από τις δύο αυτές κινήσεις, στη γειτονιά U^S θα περιέχονται $N(N-1) - (N-1)$, δηλαδή $N^2 - 2N + 1$ διαφορετικά προγράμματα. □

4.5 Αναζήτηση με απαγόρευση κινήσεων

Το κρισιμότερο σημείο στην απόδοση των αλγορίθμων τοπικής αναζήτησης, όπως αυτού που παρουσιάσαμε στην ενότητα 4.2, είναι η παγίδευσή τους σε γειτονιές τοπικών ελαχίστων. 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 έως να εντοπισθεί κάποιο τοπικό ελάχιστο. Στο σημείο αυτό η αναζήτηση εγκλωβίζεται στη γειτονιά του τοπικού ελαχίστου, καθώς καμία κίνηση προς καμία κατεύθυνση δε βελτιώνει πλέον τη λύση. Στην περίπτωση αυτή η τοπική αναζήτηση συνήθως τερματίζεται και το τοπικό ελάχιστο αποτελεί την τελική λύση. Αφού το πρόγραμμα που αποτελεί τη βέλτιστη λύση, είναι ένα από τα τοπικά ελάχιστα του χώρου $\mathcal{S}_N$, η ποιότητα της τελικής λύσης που παράγουν οι αλγόριθμοι τοπικής αναζήτησης θα εξαρτάται από το πλήθος των τοπικών ελαχίστων. Καθώς η δομή

του χώρου $\mathcal{S}_N$ δεν είναι γνωστή, το πλήθος των τοπικών ελαχίστων που περιέχονται σε αυτόν δεν μπορεί να υπολογισθεί ακριβώς. Μπορεί, όμως, να θεωρηθεί ανάλογο του μεγέθους του χώρου αυτού [Eva87], με αποτέλεσμα σε μεγάλου μεγέθους χώρους να περιμένουμε ότι το πλήθος των τοπικών ελαχίστων είναι σημαντικό. Για το λόγο αυτό 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 [Glo89].

Ας υποθέσουμε ότι κατά την αναζήτηση εντοπίζεται κάποιο τοπικό ελάχιστο Π^*. Αν Π' είναι το ελάχιστο της γειτονιάς $U^S(\Pi^*) - \{\Pi^*\}$ τότε είναι φανερό ότι θα ισχύει $C_{max}(\Pi^*) \leq C_{max}(\Pi') \leq C_{max}(\Pi)$, για οποιοδήποτε πρόγραμμα $\Pi \in U^S(\Pi^*)$. Στο σημείο αυτό καμία κίνηση προς καμία κατεύθυνση δεν πρόκειται να βελτιώσει την τρέχουσα λύση. Η αναζήτηση, όμως, μπορεί να συνεχιστεί δεχόμενη λύση χειρότερη του Π^*, ελπίζοντας ότι η χειροτέρευση αυτή είναι προσωρινή και ότι μελλοντικά η λύση θα βελτιωθεί. Από τα προγράμματα της γειτονιάς $U^S(\Pi^*)$ το πρόγραμμα Π' φαίνεται να αποτελεί την καλύτερη επιλογή, καθώς προκαλεί τη μικρότερη αύξηση στην τιμή του C_{max}. Επιλέγοντας το Π' δύο περιπτώσεις είναι δυνατές :

- Να υπάρχει $\Pi_l \in U(\Pi') - U(\Pi^*)$, με την ιδιότητα $C_{max}(\Pi_l) < C_{max}(\Pi^*)$.

  Στην περίπτωση αυτή υπάρχει καλύτερη λύση στην ευρύτερη περιοχή του τοπικού ελαχίστου Π^*, ο εντοπισμός της οποίας δε θα ήταν δυνατός χωρίς την επιλογή του προγράμματος Π'. Συνήθως, τα τοπικά ελάχιστα εμφανίζονται απομονωμένα στο χώρο αναζήτησης, με αποτέλεσμα η περίπτωση αυτή να συναντάται σπάνια.

- Να μην υπάρχει τέτοιο πρόγραμμα Π_l.

  Καθώς δεν υπάρχει πρόγραμμα $\Pi_l \in U^S(\Pi')$, τέτοιο ώστε $C_{max}(\Pi_l) < C_{max}(\Pi^*)$ θα ισχύει ότι $C_{max}(\Pi^*) \leq C_{max}(\Pi)$ για κάθε πρόγραμμα $\Pi \in U^S(\Pi')$. Επίσης, από την πρώτη ιδιότητα της κίνησης K γνωρίζουμε ότι ισχύει $K^{-1}(K(\Pi)) = \Pi$, όπου K^{-1} η αντίστροφη της κίνησης K. Άρα το τοπικό ελάχιστο Π^* θα περιέχεται στη γειτονιά $U^S(\Pi')$, αφού το πρόγραμμα Π' ανήκει στη γειτονιά $U^S(\Pi^*)$. Επομένως, για να συνεχιστεί η αναζήτηση και να απομακρυνθεί από την περιοχή του τοπικού ελαχίστου, θα πρέπει να εξεταστούν τα προγράμματα της γειτονιάς $U^S(\Pi') - \{\Pi^*\}$. Διαφορετικά η αναζήτηση θα κινηθεί πίσω προς το Π^*.

Όσο η αναζήτηση απομακρύνεται από περιοχές τοπικών ελαχίστων πρέπει αυτά να εξαιρούνται των λύσεων που εξετάζονται. Αυτό μπορεί να επιτευχθεί απαγορεύοντας

τις κινήσεις εκείνες που οδηγούν στα τοπικά ελάχιστα [Glo89, Glo90a]. Θεωρείστε για παράδειγμα, το τοπικό ελάχιστο Π^*. Έστω ακόμη Π' το ελάχιστο της γειτονιάς $U^S(\Pi^*)$. Αυτό σημαίνει ότι υπάρχει κίνηση K_{ij} τέτοια ώστε $\Pi' = K_{ij}(\Pi^*)$. Επομένως, για να περιοριστεί η αναζήτηση στη γειτονιά $U^S(\Pi') - \{\Pi^*\}$, αρκεί να απαγορευθεί η αντίστροφη κίνηση $K_{ij}^{-1}(\Pi')$ αφού $K_{ij}^{-1}(\Pi') = \Pi^*$.

Η απαγόρευση μεμονωμένων κινήσεων δεν εξασφαλίζει ούτε τον απεγκλωβισμό από περιοχές τοπικών ελαχίστων ούτε αποτρέπει κύκλους κατά την αναζήτηση. Στο προηγούμενο παράδειγμα, θεωρείστε ότι Π'' είναι το ελάχιστο της γειτονιάς $U^S(\Pi')$. Είναι δυνατό να υπάρχει κίνηση K_{kl}, τέτοια ώστε $K_{kl}(\Pi'') = \Pi^*$. Καθώς η κίνηση K_{kl} δεν είναι απαγορευμένη η αναζήτηση θα επιστρέψει στο Π^*, αν στη γειτονιά $U^S(\Pi'')$ δεν υπάρχει καλύτερη λύση. Παρά την απαγόρευση της κίνησης K_{ij}^{-1} είναι δυνατός ο άσκοπος κύκλος $\Pi^* \rightarrow \Pi' \rightarrow \Pi'' \rightarrow \Pi^*$. Επίσης, αν το τοπικό ελάχιστο Π^* δεν είναι μοναδικό τότε στη γειτονιά του θα υπάρχει τουλάχιστον ένα ακόμη πρόγραμμα Π^{**}, για το οποίο θα ισχύει ότι $C_{max}(\Pi^*) = C_{max}(\Pi^{**})$. Στην περίπτωση αυτή είναι πιθανό να υπάρχει επιτρεπτή κίνηση K_{rs}, τέτοια ώστε $K_{rs}(\Pi') = \Pi^{**}$, με αποτέλεσμα η αναζήτηση να παγιδευθεί στην περιοχή του τοπικού ελαχίστου.

Για να αποτραπούν άσκοποι κύκλοι είναι απαραίτητη η απαγόρευση συνόλου κινήσεων. Θεωρείστε ότι κατά την αναζήτηση του προηγούμενου παραδείγματος δεν απαγορεύεται μόνο η κίνηση K_{ij}^{-1}, η οποία οδηγεί στο τοπικό ελάχιστο Π^*, αλλά όλες οι κινήσεις που οδηγούν σε αυτό. Με τον τρόπο αυτό σχηματίζεται ένα σύνολο $TL_K(\Pi^*)$ απαγορευμένων κινήσεων, το οποίο ορίζεται ως εξής :

$$TL_K(\Pi^*) \ = \ \{ \ K \ \mid \ K(\Pi) = \Pi^* \ \}$$

Με την απαγόρευση των κινήσεων του συνόλου $TL_K(\Pi^*)$ αποφεύγονται κύκλοι, όπως ο $\Pi^* \rightarrow \Pi' \rightarrow \Pi'' \rightarrow \Pi^*$, αφού $K_{kl} \in TL_K(\Pi^*)$. Δεν εξασφαλίζεται, όμως, η απομάκρυνση της αναζήτησης από την περιοχή του τοπικού ελαχίστου, καθώς η κίνηση K_{rs} για παράδειγμα δεν περιέχεται στο σύνολο $TL_K(\Pi^*)$. Είναι, λοιπόν, απαραίτητη η επέκταση του $TL_K(\Pi^*)$ και ο ορισμός ενός νέου συνόλου, ώστε να απαγορεύονται κινήσεις όπως η K_{rs}.

Για το λόγο αυτό ορίζουμε το σύνολο $TL_{KC}(\Pi^*)$, ώστε να περιέχει όλες τις κινήσεις που οδηγούν σε προγράμματα με χρόνους περάτωσης ίσους με αυτόν του τοπικού ελαχίστου Π^*. Το σύνολο $TL_{KC}(\Pi^*)$ ορίζεται με την ακόλουθη σχέση :

$$TL_{KC}(\Pi^*) \ = \ \{ \ K \ \mid \ C_{max}(K(\Pi)) = C_{max}(\Pi^*) \ \}$$

όπου Π^* τοπικό ελάχιστο. Εύκολα διαπιστώνουμε ότι το σύνολο $TL_{KC}(\Pi^*)$ ικανοποιεί

τις απαιτήσεις που θέσαμε, καθώς περιέχονται σε αυτό κινήσεις, όπως οι K_{kl} και K_{rs}. Η απαγόρευση των κινήσεων του συνόλου $TL_{KC}(\Pi^*)$ περιορίζει την αναζήτηση στη γειτονιά $U^S - TL_{\Pi}(\Pi^*)$, όπου $TL_{\Pi}(\Pi^*)$ το σύνολο των προγραμμάτων που παράγονται με απαγορευμένες κινήσεις. Το σύνολο $TL_{\Pi}(\Pi^*)$ μπορεί να οριστεί ως εξής :

$$TL_{\Pi}(\Pi^*) = \{\, K(\Pi) \in U^S \;\mid\; K \in TL_{KC}(\Pi^*) \,\}$$

Ο ορισμός του συνόλου $TL_{\Pi}(\Pi^*)$ μέσω του συνόλου $TL_{KC}(\Pi^*)$ με την παραπάνω σχέση είναι δύσχρηστος στην πράξη, καθώς οι κινήσεις που οδηγούν σε τοπικά ελάχιστα δεν είναι γνωστές εκ των προτέρων. Επίσης, το πλήθος των απαγορευμένων κινήσεων είναι σε πολλές περιπτώσεις εξαιρετικά μεγάλο, με αποτέλεσμα να είναι χρονοβόρος η διαδικασία εξακρίβωσης αν μια κίνηση είναι επιτρεπτή. Επειδή τα προγράμματα που παράγονται με απαγορευμένες κινήσεις του συνόλου $TL_{KC}(\Pi^*)$ χαρακτηρίζονται από την ιδιότητα να έχουν χρόνους περάτωσης C_{max}, οι οποίοι είναι τοπικά ελάχιστοι, μπορούμε να ορίσουμε το σύνολο $TL_{\Pi}(\Pi^*)$ με την ακόλουθη σχέση :

$$TL_{\Pi}(\Pi^*) \;=\; \{\, \Pi \in \mathcal{S}_N \;\mid\; C_{max}(\Pi) \;=\; C_{max}(\Pi^*) \,\} \tag{4.7}$$

4.6 Περιορισμός της αναζήτησης

Το μέγεθος της γειτονιάς U^S είναι καθοριστικό για τον υπολογιστικό χρόνο που απαιτεί η διαδικασία της αναζήτησης. Το πλήθος των προγραμμάτων που εξετάζονται σε κάθε επανάληψη του αλγορίθμου τοπικής αναζήτησης είναι τάξης $\mathcal{O}(N^2)$. Ο αριθμός αυτός είναι σημαντικός ακόμη και για μετρίου μεγέθους προβλήματα. Κρίνεται, επομένως, απαραίτητος ένας μηχανισμός αποκλεισμού προγραμμάτων από την αναζήτηση και περιορισμού του μεγέθους των γειτονιών που εξετάζονται.

Ας θεωρήσουμε την $k-$στη επανάληψη του αλγορίθμου τοπικής αναζήτησης που παρουσιάστηκε στην ενότητα 4.1 και το πρόγραμμα Π_k που αποτελεί την τρέχουσα λύση. Η αναζήτηση θα συνεχιστεί με την εύρεση του προγράμματος Π_{k+1}, το οποίο αποτελεί ελάχιστο της γειτονιάς $U^S(\Pi_k)$. Αυτό σημαίνει ότι υπάρχει μια εργασία a στη θέση i του προγράμματος Π_k, η οποία μετατιθέμενη στη θέση j παράγει το πρόγραμμα Π_{k+1}. Ισχύει, δηλαδή, $K_{ij}(\Pi_k) = \Pi_{k+1}$. Αφού το Π_{k+1} είναι ελάχιστο γειτονιάς, η αναζήτηση θα συνεχιστεί εξετάζοντας τα προγράμματα που περιέχονται στη γειτονιά $U^S(\Pi_{k+1})$. Ας συμβολίσουμε $U_a^S(\Pi_{k+1})$ το σύνολο των προγραμμάτων που παράγονται από το Π_{k+1} με τη μετάθεση της εργασίας a. Καθώς η εργασία a βρίσκεται στη θέση j του Π_{k+1}, το

σύνολο $U_a^S(\Pi_{k+1})$ μπορεί να οριστεί ως εξής :

$$U_a^S(\Pi_{k+1}) \;=\; U_a^{RS}(\Pi_{k+1}) \;\cup\; U_a^{LS}(\Pi_{k+1})$$

όπου :

$$U_a^{RS}(\Pi_{k+1}) \;=\; \{\,\Pi \in \mathcal{S}_N \;\mid\; \Pi = K_{jk}(\Pi_{k+1}) \quad 1 \le j < k \le N\}$$

και

$$U_a^{LS}(\Pi_{k+1}) \;=\; \{\,\Pi \in \mathcal{S}_N \;\mid\; \Pi = K_{kj}(\Pi_{k+1}) \quad 1 \le k < j \le N\}$$

Από την τρίτη ιδιότητα της κίνησης K εύκολα βλέπουμε ότι $U_a^S(\Pi_{k+1}) \subset U^S(\Pi_k)$. Επομένως, θα ισχύει ότι :

$$C_{max}(\Pi_{k+1}) \le C_{max}(\Pi) \quad \forall \Pi \in\; U_a^S(\Pi_{k+1}) \tag{4.8}$$

αφού το Π_{k+1} είναι ελάχιστο της γειτονιάς $U^S(\Pi_k)$.

Αν το πρόγραμμα Π_{k+1} δεν είναι τοπικό ελάχιστο (ελάχιστο της γειτονιάς $U^S(\Pi_{k+1})$), τότε το ελάχιστο της γειτονιάς $U^S(\Pi_{k+1}) - \{\Pi_{k+1}\}$, το οποίο ας συμβολίσουμε Π_l, αποτελεί την καλύτερη επιλογή για τη συνέχιση της αναζήτησης καθώς ισχύει ότι $C_{max}(\Pi_l) < C_{max}(\Pi_{k+1})$. Από τη σχέση 4.8 προκύπτει ότι το πρόγραμμα Π_l δεν μπορεί να περιέχεται στο σύνολο $U_a^S(\Pi_{k+1})$. Αντίθετα, αν το πρόγραμμα Π_{k+1} είναι τοπικό ελάχιστο τότε για το πρόγραμμα Π_l θα ισχύει ότι $C_{max}(\Pi_{k+1}) \le C_{max}(\Pi_l)$. Στην περίπτωση αυτή η καλύτερη κατεύθυνση για να συνεχιστεί η αναζήτηση, είναι εκείνη που την απομακρύνει από τη γειτονιά του τοπικού ελαχίστου προκαλώντας αύξηση στην τιμή του C_{max}. Όμως, καθώς $U_a^S(\Pi_{k+1}) \;\subset\; U^S(\Pi_k) \cap U^S(\Pi_{k+1})$ τα προγράμματα του συνόλου $U_a^S(\Pi_{k+1})$ δε φαίνεται να αποτελούν την καλύτερη επιλογή, καθώς δεν απεγκλωβίζουν την αναζήτηση από την περιοχή του τοπικού ελαχίστου. Επομένως, τα προγράμματα που παράγονται από την κίνηση της εργασίας a μπορούν να αποκλειστούν από την εξέταση της $U^S(\Pi_{k+1})$ περιορίζοντας την αναζήτηση στη γειτονιά $U^S(\Pi_{k+1}) - U_a^S(\Pi_{k+1})$ χωρίς να επηρεαστεί σημαντικά η ποιότητα της τελικής λύσης που παράγεται.

Η επέκταση του παραπάνω συλλογισμού ώστε σε κάθε επανάληψη του αλγορίθμου να απαγορεύεται η κίνηση P τον αριθμό εργασιών, παρουσιάζει ιδιαίτερο ενδιαφέρον για ένα ακόμα λόγο εκτός από τον περιορισμό του μεγέθους της γειτονιάς αναζήτησης. Εκτός από ειδικές και σπάνιες περιπτώσεις οι αρχικές λύσεις διαφέρουν σημαντικά από τις τελικές ως προς τη θέση των εργασιών στα αρχικά και τελικά προγράμματα. Ο περιορισμός της αναζήτησης σε κινήσεις ενός μικρού αριθμού εργασιών ελαττώνει την πιθανότητα παραγωγής καλής τελικής λύσης. Επομένως, η απαγόρευση των μεταθέσεων

P τον αριθμό εργασιών σε κάθε επανάληψη του αλγορίθμου αφενός εξασφαλίζει την εξέταση διαφορετικών προγραμμάτων αφετέρου μειώνει το μέγεθος των γειτονιών, καθώς η αναζήτηση περιορίζεται σε ένα υποσύνολο U^{S_P} της γειτονιάς U^S. Η επιλογή κατάλληλης τιμής για την παράμετρο P είναι κρίσιμη για την απόδοση του αλγορίθμου. Ας σημειωθεί ότι αν $P = 0$ τότε $U^{S_P} = U^S$, ενώ αν $P = N$ τότε $U^{S_P} = \emptyset$.

Για την αξιολόγηση του περιορισμού του μεγέθους των γειτονιών χρήσιμος είναι ο υπολογισμός του λόγου $\mathcal{R} = \frac{|U^{S_P}|}{|U^S|}$, όπου $|\ U^{S_P}\ |$ και $|\ U^S\ |$ τα μεγέθη των δύο γειτονιών. Ο ακριβής υπολογισμός του λόγου $\mathcal{R}$ δεν είναι δυνατός καθώς το μέγεθος της γειτονιάς U^{S_P} δεν εξαρτάται μόνο από την τιμή της παραμέτρου P αλλά και από τη θέση των εργασιών, των οποίων η κίνηση απαγορεύεται. Αυτό θα το δείξουμε με ένα παράδειγμα.

Έστω ένα πρόβλημα προγραμματισμού τεσσάρων εργασιών και το πρόγραμμα $abcd$. Όπως φαίνεται στο σχήμα 4.1, στη γειτονιά του $abcd$ περιέχονται εννέα προγράμματα, από τα οποία τα έξι παράγονται με μία μόνο κίνηση και τα υπόλοιπα τρία με δύο κινήσεις. Το πρόγραμμα $bcad$, για παράδειγμα, μπορεί να παραχθεί μόνο με την κίνηση $K_{13}(abcd)$, ενώ αντίθετα το πρόγραμμα $bacd$ μπορεί να παραχθεί με τις κινήσεις $K_{12}(abcd)$ και $K_{21}(abcd)$.

<table>
<tr><td colspan="4">$U^S(abcd)$</td></tr>
<tr><td>Κίνηση εργασίας a</td><td>Κίνηση εργασίας b</td><td>Κίνηση εργασίας c</td><td>Κίνηση εργασίας d</td></tr>
<tr><td colspan="2">$bacd$</td><td>$cabd$</td><td>$dabc$</td></tr>
<tr><td>$bcad$</td><td colspan="2">$acbd$</td><td>$adbc$</td></tr>
<tr><td>$bcda$</td><td>$acdb$</td><td colspan="2">$abdc$</td></tr>
</table>

Σχήμα 4.1: Η γειτονιά $U^S(abcd)$

Η απαγόρευση των τριών δυνατών μεταθέσεων της εργασίας a αποκλείει από τη γειτονιά $U^S(abcd)$ δύο μόνο προγράμματα, τα $bcad$ και $bcda$. Τα προγράμματα αυτά μπορούν να παραχθούν μόνο με μετάθεση της εργασίας a και συγκεκριμένα με τις κινήσεις $K_{13}(abcd)$ και $K_{14}(abcd)$ αντίστοιχα. Η απαγόρευση, όμως, της κίνησης $K_{12}(abcd)$ δεν αποκλείει το πρόγραμμα $bacd$, καθώς αυτό μπορεί να παραχθεί και με την κίνηση $K_{21}(abcd)$, δηλαδή με μετάθεση της εργασίας b. Αν συμβολίσουμε $U^S_{\bar{a}}(abcd)$ τη γειτονιά που σχηματίζεται με τον τρόπο αυτό τότε $U^S_{\bar{a}}(abcd) = U^S(abcd) - \{bcad, bcda\}$. Η περίπτωση αυτή απεικονίζεται στο σχήμα 4.2 .

$U^S_{\bar{a}}(abcd)$			
Κίνηση εργασίας a	Κίνηση εργασίας b	Κίνηση εργασίας c	Κίνηση εργασίας d
	$bacd$	$cabd$	$dabc$
	$acbd$		$adbc$
	$acdb$	$abdc$	

Σχήμα 4.2: Η γειτονιά $U^S_{\bar{a}}(abcd)$. Απαγορεύεται η κίνηση της a

Η απαγόρευση, όμως, των τριών δυνατών μεταθέσεων της εργασίας b αποκλείει από τη γειτονιά $U^S(abcd)$ ένα μόνο πρόγραμμα, το $acdb$, καθώς μόνο το πρόγραμμα αυτό παράγεται αποκλειστικά με μετάθεση της εργασίας b και συγκεκριμένα με την κίνηση $K_{24}(abcd)$. Οι δύο άλλες κινήσεις, $K_{21}(abcd)$ και $K_{23}(abcd)$ που απαγορεύτηκαν, δεν αποκλείουν τα προγράμματα $bacd$ και $acdb$, καθώς το πρώτο παράγεται και με την κίνηση $K_{12}(abcd)$, δηλαδή με μετάθεση της εργασίας a, ενώ το δεύτερο παράγεται και με την κίνηση $K_{32}(abcd)$, δηλαδή με μετάθεση της εργασίας c. Αν συμβολίσουμε $U^S_{\bar{b}}(abcd)$ τη γειτονιά που παράγεται με τον τρόπο αυτό τότε $U^S_{\bar{b}}(abcd) = U^S - \{acdb\}$. Η περίπτωση αυτή απεικονίζεται στο σχήμα 4.3 .

$U^S_{\bar{b}}(abcd)$			
Κίνηση εργασίας a	Κίνηση εργασίας b	Κίνηση εργασίας c	Κίνηση εργασίας d
$bacd$		$cabd$	$dabc$
$bcad$		$acbd$	$adbc$
$bcda$		$abdc$	

Σχήμα 4.3: Η γειτονιά $U^S_{\bar{b}}(abcd)$. Απαγορεύεται η κίνηση της b

Βλέπουμε, λοιπόν, ότι με την απαγόρευση των μεταθέσεων διαφορετικών εργασιών δημιουργούνται γειτονιές διαφορετικού μεγέθους. Η γειτονιά που προέκυψε με την απαγόρευση των μεταθέσεων της εργασίας a περιέχει επτά προγράμματα, ενώ η γειτονιά που σχηματίστηκε με την απαγόρευση των μεταθέσεων της εργασίας b περιέχει οκτώ προγράμματα.

Καθώς δεν μπορούμε να υπολογίσουμε το ακριβές μέγεθος της γειτονιάς U^{S_P} θα υπολογίσουμε δύο φράγματά του.

Πρόταση 4.1 $N^2-(P+2)N+P+1 \leq \mid U^{S_P} \mid \leq N^2-(P+2)N+3P+1$, όπου $\mid U^{S_P} \mid$ το πλήθος των προγραμμάτων N εργασιών, τα οποία παράγονται όταν απαγορεύονται οι μεταθέσεις P τον αριθμό εργασιών.

Απόδειξη

Έστω ένα πρόγραμμα N εργασιών. Από τον ορισμό της γειτονιάς U^S γνωρίζουμε ότι κάθε εργασία του προγράμματος μπορεί να τοποθετηθεί σε $N-1$ θέσεις παράγοντας $N-1$ προγράμματα. Επομένως, η απαγόρευση των μεταθέσεων P τον αριθμό εργασιών μπορεί να αποκλείσει από τη γειτονιά U^S το πολύ $P(N-1)$ προγράμματα. Άρα η γειτονιά U^{S_P} δεν μπορεί να περιέχει λιγότερα από $\mid U^S \mid -P(N-1)$ προγράμματα. Όμως, $\mid U^S \mid = N^2-2N+1$. Οπότε έχουμε $N^2-2N+1-P(N-1) \leq \mid U^{S_P} \mid$ και ισοδύναμα $N^2-(P+2)N+P+1 \leq \mid U^{S_P} \mid$.

Έστω η εργασία στη θέση i ενός προγράμματος N εργασιών. Από τα $N-1$ προγράμματα που παράγονται με μεταθέσεις της εργασίας αυτής τουλάχιστον δύο παράγονται με τις κινήσεις $K_{i,i+1}$ και $K_{i,i-1}$ (Αν η εργασία βρίσκεται στην πρώτη ή στην τελευταία θέση του προγράμματος τότε μόνο μία από τις δύο κινήσεις είναι εφικτή). Καθώς τα προγράμματα που παράγονται με τις κινήσεις αυτές μπορούν να παραχθούν και με τις αντίστροφές τους, $K_{i+1,i}$ και $K_{i-1,i}$ αντίστοιχα, η απαγόρευση της κίνησης της συγκεκριμένης εργασίας ενδέχεται να μην αποκλείσει τα προγράμματα αυτά από τη γειτονιά U^{S_P}. Επομένως, η απαγόρευση των μεταθέσεων P τον αριθμό εργασιών αποκλείει από τη γειτονιά U^S τουλάχιστον $P(N-3)$ προγράμματα. Άρα η γειτονιά U^{S_P} δεν μπορεί να περιέχει περισσότερα από $\mid U^S \mid -P(N-3)$ προγράμματα. Οπότε έχουμε $\mid U^{S_P} \mid \leq N^2-2N+1-P(N-3)$ και ισοδύναμα $\mid U^{S_P} \mid \leq N^2-(P+2)N+3P+1$.

Άρα $N^2-(P+2)N+P+1 \leq \mid U^{S_P} \mid \leq N^2-(P+2)N+3P+1$ □

Από την παραπάνω πρόταση προκύπτει ότι :

$$\frac{N^2-(P+2)N+P+1}{N^2-2N+1} \leq \mathcal{R} \leq \frac{N^2-(P+2)N+3P+1}{N^2-2N+1}$$

και ισοδύναμα :

$$1-\frac{P(N-1)}{N^2-2N+1} \leq \mathcal{R} \leq 1-\frac{P(N-3)}{N^2-2N+1}$$

4.7 Κριτήρια τερματισμού

Καθώς με την απαγόρευση κινήσεων η αναζήτηση δεν ολοκληρώνεται πλέον με την εύρεση τοπικού ελαχίστου, απαιτούνται νέες συνθήκες τερματισμού της εκτέλεσης του αλγορίθμου. Τρεις είναι οι περιπτώσεις που συνήθως επιβάλλουν τον τερματισμό της αναζήτησης :

1. Αν $U(\Pi) - TL_{\Pi} = \emptyset$, όπου TL_{Π} το σύνολο των προγραμμάτων που παράγονται με απαγορευμένες κινήσεις.

   Η συνθήκη αυτή ικανοποιείται όταν σε κάποια επανάληψη του αλγορίθμου όλες οι διαθέσιμες κινήσεις απαγορεύονται. Τότε η αναζήτηση έχει φτάσει σε αδιέξοδο και ο τερματισμός της είναι υποχρεωτικός. Αδιέξοδα σπάνια εμφανίζονται, καθώς το πλήθος των προγραμμάτων που περιέχονται στις γειτονιές αναζήτησης είναι συνήθως πολύ μεγαλύτερο από τον αριθμό των απαγορευμένων κινήσεων.

2. Αν εκτελεσθούν max_t τον αριθμό επαναλήψεις χωρίς να βελτιωθεί η λύση.

   Η εύρεση τοπικού ελαχίστου δεν τερματίζει την αναζήτηση, αντίθετα την στρέφει σε κατευθύνσεις που την απομακρύνουν από την περιοχή του τοπικού ελαχίστου με την προοπτική της εύρεσης καλύτερης λύσης. Όμως, αν το τοπικό ελάχιστο που εντοπίσθηκε είναι και ολικό τότε η συνέχιση της αναζήτησης είναι άσκοπη αφού καμία βελτίωση της λύσης δεν είναι πλέον δυνατή. Η περίπτωση αυτή είναι σπάνια, συναντάται όμως σε μικρού μεγέθους προβλήματα, όπου ο μικρός αριθμός προγραμμάτων που περιέχονται στο χώρο $\mathcal{S}_N$ επιτρέπει την εύρεση ολικού ελαχίστου.

   Το τεράστιο πλήθος προγραμμάτων του χώρου $\mathcal{S}_N$ ακόμα και για μετρίου μεγέθους προβλήματα, έχει ως αποτέλεσμα τη συχνή εμφάνιση απομονωμένων τοπικών ελαχίστων, τα οποία στη βιβλιογραφία ονομάζονται μικρές μαύρες τρύπες (mini black holes) [Ric83]. Ο απεγκλωβισμός της αναζήτησης από τέτοιες περιοχές απαιτεί υπερβολικά μεγάλο αριθμό επαναλήψεων ώστε να κρίνεται πρακτικά αδύνατος και να επιβάλεται ο τερματισμός της.

   Με την εξέταση προγραμμάτων γειτονιών δεν είναι δυνατό να διαπιστωθεί αν κάποιο τοπικό ελάχιστο είναι ολικό ή απομονωμένο ώστε να τερματισθεί αμέσως μια άσκοπη και υπολογιστικά χρονοβόρος αναζήτηση. Για το λόγο αυτό επιλέγεται ως κριτήριο τερματισμού ένα όριο max_t στον αριθμό των επαναλήψεων που εκτελούνται χωρίς να επιτευχθεί καμία βελτίωση της λύσης. Να σημειωθεί ότι η

συνθήκη αυτή αποτελεί το συνηθέστερο κριτήριο τερματισμού αλγορίθμων τοπικής αναζήτησης με απαγόρευση κινήσεων.

3. Αν εκτελεσθούν max_k τον αριθμό επαναλήψεις.

   Με τη χρήση των δύο κριτηρίων που παρουσιάστηκαν εξασφαλίζεται ο τερματισμός ενός αλγορίθμου τοπικής αναζήτησης με απαγόρευση κινήσεων, χωρίς όμως να προσδιορίζεται ο ακριβής αριθμός επαναλήψεων που θα εκτελεσθούν. Στη χειρότερη περίπτωση η αναζήτηση θα ολοκληρωθεί με την εξετάση όλων σχεδόν των προγραμμάτων του χώρου $\mathcal{S}_N$. Αυτό είναι δυνατό, αν για παράδειγμα η λύση βελτιώνεται κάθε max_t επαναλήψεις. Αν και στην πράξη οι αλγόριθμοι αυτοί δεν παρουσιάζουν τόσο άσχημη συμπεριφορά [WH89], συχνά υπάρχει ένα χρονικό όριο στην εκτέλεσή τους. Για το λόγο αυτό χρησιμοποιείται και ένα τρίτο κριτήριο τερματισμού, η εκτέλεση το πολύ max_k τον αριθμό επαναλήψεων.

4.8 Αλγόριθμος TS_C

Ο αλγόριθμος TS_C αποτελεί ευρηματική μέθοδο τοπικής αναζήτησης με απαγόρευση κινήσεων και αναπτύχθηκε στα πλαίσια της παρούσας εργασίας για τη λύση του προβλήματος $N/\,M/\,F,\ Seq-Ind,\ perm/\,C_{max}$, στο οποίο μοναδικό κριτήριο βελτίστου είναι ο χρόνος περάτωσης C_{max} προγράμματος εργασιών.

Αρχικά προγράμματα μπορούν να παραχθούν με μία από τις πέντε ευρηματικές μεθόδους που παρουσιάστηκαν στην ενότητα 4.3. Αποτελεί παράμετρο του αλγορίθμου ποια από τις μεθόδους αυτές χρησιμοποιείται κάθε φορά. Ως γειτονιά αναζήτησης επιλέχθηκε η γειτονιά U^S που παράγεται από την κίνηση K, η οποία ορίζεται από τη σχέση 4.6 της ενότητας 4.4. Η εκτέλεση του αλγορίθμου τερματίζεται όταν ικανοποιείται ένα τουλάχιστον από τα τρία κριτήρια που παρουσιάστηκαν στην ενότητα 4.7. Ο συνολικός αριθμός max_k επαναλήψεων και το πλήθος max_t των επαναλήψεων που είναι δυνατό να εκτελεσθούν δίχως να βελτιωθεί η λύση, αποτελούν παραμέτρους του αλγορίθμου.

Η απαγόρευση της κίνησης P τον αριθμό εργασιών έχει ως απότελεσμα τον περιορισμό της αναζήτησης στη γειτονιά $U^{S_P} \subset U^S$. Οι εργασίες, των οποίων η κίνηση απαγορεύεται, κρατώνται σε μια κυκλική λίστα, την οποία συμβολίζουμε με TL_P και περιέχει το πολύ P το πλήθος στοιχεία. Το μέγεθος P αποτελεί παράμετρο του αλγορίθμου.

Για την απαγόρευση κινήσεων που ενδεχομένως οδηγούν σε κύκλους ή παγιδεύουν την αναζήτηση σε περιοχές τοπικών ελαχίστων, είναι απαραίτητο το σύνολο TL_Π, το οποίο ορίζεται από τη σχέση 4.7 της ενότητας 4.5. Από τη σχέση 4.7 παρατηρούμε ότι δεν είναι απαραίτητο να αποθηκεύονται όλα τα προγράμματα που περιέχονται στο σύνολο TL_Π, το πλήθος των οποίων ενδέχεται να είναι μεγάλο. Αφού τα προγράμματα αυτά χαρακτηρίζονται από την ιδιότητα ότι οι χρόνοι C_{max} περάτωσής τους είναι τοπικά ελάχιστοι σε κάποιες περιοχές του χώρου αναζήτησης $\mathcal{S}_N$, είναι αρκετό να κρατηθούν σε μια λίστα TL_C οι χρόνοι αυτοί. Στην περίπτωση αυτή, αντί να ελέγχουμε αν ένα πρόγραμμα Π περιέχεται στο σύνολο TL_Π, αρκεί να εξετάσουμε αν ο χρόνος $C_{max}(\Pi)$ περάτωσής του περιέχεται στη λίστα TL_C. Το μέγεθος C του μέγιστου πλήθους στοιχείων της λίστας TL_C αποτελεί παράμετρο του αλγορίθμου.

Μπορούμε τώρα να παρουσιάσουμε τον αλγόριθμο TS_C :

Είσοδος Το πλήθος N των εργασιών, το πλήθος M των μηχανών, οι χρόνοι p επεξεργασίας των εργασιών, οι χρόνοι s εξάρμωσης των μηχανών και οι ομάδες των εργασιών.

Παράμετροι Η ευρηματική μέθοδος για την παραγωγή αρχικού προγράμματος, ο μέγιστος δυνατός αριθμός επαναλήψεων max_k, ο μέγιστος δυνατός αριθμός επαναλήψεων max_t από την τελευταία φορά που βελτιώθηκε η λύση και τα μεγέθη P και C.

Έξοδος Το πρόγραμμα $\Pi_{\beta\varepsilon\lambda\tau}$.

Βήμα 1 Δημιούργησε με ευρηματικό τρόπο ένα αρχικό πρόγραμμα Π_0.

Θέσε $\Pi_{\beta\varepsilon\lambda\tau} = \Pi_0$.

Βήμα 2 Θέσε

- $TL_C = \emptyset$.
- $TL_P = \emptyset$.
- $k = 0$.
- $t = 0$.

Βήμα 3 Αν $U^{S_P}(\Pi_k) - TL_\Pi$[1] $= \emptyset$ τότε τερμάτισε.

[1] Ας μας επιτραπεί μια μικρή διαφοροποίηση, τύπων και όχι ουσίας, στον ορισμό του συνόλου TL_Π από αυτόν της σχέσης 4.7. $TL_\Pi = \{ \Pi \in \mathcal{S}_N \mid C_{max}(\Pi) \in TL_C \}$.

Αλλιώς :

Βρες $\Pi_l \in U^{S_P}(\Pi_k) - TL_\Pi$, τέτοιο ώστε

$$C_{max}(\Pi_l) < C_{max}(\Pi)\ , \quad \forall \Pi \in U^{S_P}(\Pi_k) - TL_\Pi$$

Αν το πρόγραμμα Π_l δεν είναι μοναδικό τότε διάλεξε αυθαίρετα το πρώτο που θα βρεθεί.

Βήμα 4 Αν $C_{max}(\Pi_l) < C_{max}(\Pi_{\beta\varepsilon\lambda\tau})$ τότε

- Θέσε $\Pi_{\beta\varepsilon\lambda\tau} = \Pi_l$.
- Θέσε $t = k$.

Αλλιώς, αν $C_{max}(\Pi_l) > C_{max}(\Pi_k)$:

Τοποθέτησε στη θέση $k \, mod \, C$ της λίστας TL_C το στοιχείο $C_{max}(\Pi_l)$.

Βήμα 5 Αν $k \geq max\_k$ ή $k - t \geq max\_t$ τότε τερμάτισε.

Αλλιώς :

- Έστω $K_{ij}(\Pi_k) = \Pi_l$. Τοποθέτησε στη θέση $k \, mod \, P$ της λίστας TL_P το στοιχείο i.
- Θέσε $\Pi_{k+1} = \Pi_l$.
- Θέσε $k = k + 1$.
- Πήγαινε στο βήμα 3.

4.9 Αλγόριθμος TS_{TC}

Ο αλγόριθμος TS_{TC} αποτελεί ευρηματική μέθοδο τοπικής αναζήτησης με απαγόρευση κινήσεων και αναπτύχθηκε για τη λύση του προβλήματος $N/\ M/\ F,\ Seq - Ind,\ perm/T_{max},\ C_{max}$, στο οποίο αναζητούνται εφικτά προγράμματα ως προς τις προθεσμίες των εργασιών ή όποτε αυτό δεν είναι δυνατό, προγράμματα με μικρή καθυστέρηση των εργασιών που έχουν προθεσμίες. Το δικριτηριακό πρόβλημα προγραμματισμού εργασιών αποτελεί, όπως είδαμε στην ενότητα 4.1, περιορισμό του αντίστοιχου μονοκριτηριακού, με αποτέλεσμα τα συμπεράσματα της μελέτης μας εύκολα να επεκτείνονται και σε αυτό. Ο αλγόριθμος TS_{TC} είναι επέκταση του TS_C που παρουσιάσαμε στην προηγούμενη ενότητα για τη λύση του μονοκριτηριακού προβλήματος.

Δύο σημεία, όμως, στον αλγόριθμο TS_{TC} απαιτείται να σχολιαστούν ιδιαίτερα. Το πρώτο αφορά τη μέθοδο παραγωγής αρχικών προγραμμάτων. Στην ενότητα 4.3 παρουσιάστηκαν πέντε ευρηματικές μέθοδοι που χρησιμοποιήθηκαν για το σκοπό αυτό, καθώς η πολυπλοκότητα του μονοκριτηριακού προβλήματος ροϊκής παραγωγής καθιστά δύσκολη την αξιολόγηση των διάφορων μεθόδων. Στο δικριτηριακό πρόβλημα, όμως, η ανάθεση προθεσμιών παράδοσης στις εργασίες και η θεώρηση της μέγιστης καθυστέρησης T_{max} των εργασιών ως πρωτεύοντος κριτηρίου βελτίστου μας επιτρέπει να περιοριστούμε μόνο στη μεθόδο EDD (Earliest Due Date first) για την παραγωγή αρχικών προγαμμάτων. Η μέθοδος EDD σχηματίζει ένα πρόγραμμα εργασιών διατάσσοντας τις εργασίες σε αύξουσα σειρά ως προς τις προθεσμίες τους. Με τον τρόπο αυτό εργασίες με σφικτές προθεσμίες τοποθετούνται να εκτελεστούν πρώτες, ενώ εργασίες χωρίς ή με χαλαρές προθεσμίες εκτελούνται τελευταίες. Ειδικά σε προβλήματα προγραμματισμού εργασιών σε μία μηχανή το πρόγραμμα που παράγεται με τη μέθοδο αυτή είναι βέλτιστο [HC84]. Αν και στο γενικό πρόβλημα ροϊκής παραγωγής η μέθοδος EDD δεν παράγει πάντοτε το βέλτιστο πρόγραμμα ως προς το κριτήριο T_{max} , η διάταξη των εργασιών κατά αύξουσα σειρά αποτελεί μια πολύ καλή αρχική λύση [Κοπ93]. Για το λόγο αυτό στον αλγόριθμο TS_{TC} χρησιμοποιείται η μέθοδος EDD για την παραγωγή αρχικών προγραμμάτων.

Στο μονοκριτηριακό πρόβλημα $N/\,M/\,F,\ Seq-Ind,\ perm/\,C_{max}$ τα προγράμματα χαρακτηρίζονται μόνο από το χρόνο περάτωσής τους C_{max}. Στο δικριτηριακό, όμως, πρόβλημα $N/\,M/\,F,\ Seq-Ind,\ perm/\,T_{max},\ C_{max}$ τα προγράμματα χαρακτηρίζονται από το ζεύγος (T_{max}, C_{max}) της μέγιστης καθυστέρησης των εργασιών στο πρόγραμμα και του χρόνου περάτωσής τους. Επομένως, είναι απαραίτητη η ακόλουθη επέκταση του συνόλου TL_{Π}, το οποίο ορίζεται από τη σχέση 4.7 της ενότητας 4.5 :

$$TL_{\Pi} \;=\; \{\, \Pi \in \mathcal{S}_N \;\mid\; (T_{max}(\Pi),\, C_{max}(\Pi)) \;=\; (T_{max}(\Pi^*),\, C_{max}(\Pi^*)\,\}$$

όπου Π^* τοπικό ελάχιστο.

Στην προηγούμενη ενότητα είδαμε ότι είναι χρήσιμη η ύπαρξη μιας λίστας TL_C, με τη βοήθεια της οποίας είναι ευκολότερος ο έλεγχος αν ένα πρόγραμμα Π περιέχεται στο σύνολο TL_{Π}. Για τον ίδιο λόγο θα χρησιμοποιήσουμε στον αλγόριθμο TS_{TC} μια κυκλική λίστα TL_{TC}, όπου αποθηκεύονται ζεύγη $(T^*_{max},\ C^*_{max})$ που είναι τιμές τοπικών ελαχίστων κάποιων περιοχών του χώρου $\mathcal{S}_N$. Το μέγεθος TC του μέγιστου πλήθους στοιχείων της λίστας αυτής, αποτελεί παράμετρο του αλγορίθμου.

Μπορούμε τώρα να παρουσιάσουμε τον αλγόριθμο TS_{TC}.

Είσοδος Το πλήθος N των εργασιών, το πλήθος M των μηχανών, οι χρόνοι p επεξεργασίας των εργασιών, οι χρόνοι s εξάρμωσης των μηχανών, οι ομάδες και οι προθεσμίες των εργασιών.

Παράμετροι Ο μέγιστος δυνατός αριθμός επαναλήψεων max_k, ο μέγιστος δυνατός αριθμός επαναλήψεων max_t από την τελευταία φορά που βελτιώθηκε η λύση και τα μεγέθη P και TC.

Έξοδος Το πρόγραμμα $\Pi_{\beta\varepsilon\lambda\tau}$.

Βήμα 1 Δημιούργησε με τη μέθοδο EDD ένα αρχικό πρόγραμμα Π_0.

Θέσε $\Pi_{\beta\varepsilon\lambda\tau} = \Pi_0$.

Βήμα 2 Θέσε

- $TL_{TC} = \emptyset$.
- $TL_P = \emptyset$.
- $k = 0$.
- $t = 0$.

Βήμα 3 Αν $U^{S_P}(\Pi_k) - TL_\Pi = \emptyset$ τότε τερμάτισε.

Αλλιώς :

Βρες $\Pi_l \in U^{S_P}(\Pi_k) - TL_\Pi$, τέτοιο ώστε

$$T_{max}(\Pi_l) < T_{max}(\Pi)\,, \quad \forall \Pi \in U^{S_P}(\Pi_k) - TL_\Pi$$

Αν το πρόγραμμα Π_l δεν είναι μοναδικό τότε διάλεξε αυτό με τον ελάχιστο χρόνο περάτωσης C_{max}. Αν πάλι το πρόγραμμα αυτό δεν είναι μοναδικό τότε διάλεξε αυθαίρετα το πρώτο που θα βρεθεί.

Βήμα 4 Αν $T_{max}(\Pi_l) < T_{max}(\Pi_{\beta\varepsilon\lambda\tau})$ τότε

- Θέσε $\Pi_{\beta\varepsilon\lambda\tau} = \Pi_l$.
- Θέσε $t = k$.

Αλλιώς :

Αν $T_{max}(\Pi_l) = T_{max}(\Pi_{\beta\varepsilon\lambda\tau})$ και $C_{max}(\Pi_l) < C_{max}(\Pi_{\beta\varepsilon\lambda\tau})$ τότε

- Θέσε $\Pi_{\beta\varepsilon\lambda\tau} = \Pi_l$.
- Θέσε $t = k$.

Αλλιώς, αν $T_{max}(\Pi_l) > T_{max}(\Pi_k)$:

Τοποθέτησε στη θέση $k\, mod TC$ της λίστας TL_{TC} το ζεύγος $(T_{max}(\Pi_l), C_{max}(\Pi_l))$.

Βήμα 5 Αν $k \geq max\_k$ ή $k - t \geq max\_t$ τότε τερμάτισε.

Αλλιώς :

- Έστω $K_{ij}(\Pi_k) = \Pi_l$. Τοποθέτησε στη θέση $k\, mod\, P$ της λίστας TL_P το στοιχείο i.
- Θέσε $\Pi_{k+1} = \Pi_l$.
- Θέσε $k = k + 1$.
- Πήγαινε στο βήμα 3.

Κεφάλαιο 5

Παραλληλοποίηση της αναζήτησης

5.1 Εισαγωγή

Κρίσιμο σημείο στην απόδοση των αλγορίθμων που παρουσιάσαμε στο κεφάλαιο 4 αποτελεί το μέγεθος των γειτονιών αναζήτησης. Όσο μεγαλύτερος είναι ο αριθμός των προγραμμάτων που περιέχονται σε μια γειτονιά τόσο καλύτερη είναι η τελική λύση αλλά και πιο χρονοβόρος η διαδικασία της αναζήτησης. Συνήθη μεγέθη γειτονιών που χρησιμοποιούνται στην πράξη είναι τάξης N^2, με αποτέλεσμα ο διπλασιασμός του αριθμού των εργασιών ενός προβλήματος να συνεπάγεται τον τετραπλασιασμό του πλήθους των προγραμμάτων που περιέχονται σε μια γειτονιά. Για το λόγο αυτό προβλήματα προγραμματισμού μεγάλου αριθμού εργασιών απαιτούν υπερβολικά μεγάλο υπολογιστικό χρόνο, για να λυθούν. Μηχανισμοί περιορισμού του μεγέθους των γειτονιών, όπως αυτός που αναπτύχθηκε στο κεφάλαιο 4, δεν είναι πάντοτε επιτυχείς, καθώς η απόρριψη μερικών προγραμμάτων από μια γειτονιά ενδέχεται να χειροτερεύσει την τελική λύση του προβλήματος. Η παραλληλοποίηση της αναζήτησης μπορεί να μειώσει σημαντικά τον υπολογιστικό χρόνο που απαιτεί η λύση μεγάλων προβλημάτων με αλγορίθμους τοπικής αναζήτησης.

5.2 Παραλληλοποίηση της αναζήτησης

Έστω ένα πρόβλημα προγραμματισμού τεσσάρων εργασιών, τις οποίες συμβολίζουμε με τα τέσσερα πρώτα γράμματα του λατινικού αλφαβήτου. Έστω ακόμη το πρόγραμμα $abcd$. Τότε η γειτονιά $U^S(abcd)$ θα περιέχει εννέα προγράμματα. Οι κινήσεις των εργα-

σιών a, b, c, d παράγουν αντίστοιχα τα προγράμματα $\{bacd, bcad, bcda\}$, $\{acbd, acdb\}$, $\{cabd, abdc\}$ και $\{dabc, adbc\}$. Συμβολίζοντας με T_p τον υπολογιστικό χρόνο που απαιτείται για την παραγωγή καθενός από αυτά τα εννέα προγράμματα, η αναζήτηση στη γειτονιά $U^S(abcd)$ για την εύρεση του προγράμματος με τον ελάχιστο χρόνο περάτωσης C_{max} απαιτεί χρόνο ίσο με $9T_p$. Αν είχαμε δύο επεξεργαστές θα μπορούσαμε να χωρίσουμε σε δύο μέρη τη γειτονιά $U^S(abcd)$, αναθέτοντας στον πρώτο επεξεργαστή την αναζήτηση του ελαχίστου μεταξύ των πέντε προγραμμάτων που παράγονται από τις κινήσεις των εργασιών a και b και στο δεύτερο την αναζήτηση του ελαχίστου μεταξύ των τεσσάρων προγραμμάτων που παράγονται από τις κινήσεις των εργασιών c και d. Με την ολοκλήρωση της αναζήτησης στις δύο περιοχές της γειτονιάς $U^S(abcd)$ κάθε επεξεργαστής έχει εντοπίσει το ελάχιστο της περιοχής του και οι δύο επεξεργαστές απαιτείται να επικοινωνήσουν, για να διαπιστώσουν ποιο από τα δύο ελάχιστα είναι ελάχιστο της γειτονιάς. Αν συμβολίσουμε με T_c το χρόνο που απαιτείται για την επικοινωνία αυτή των δύο επεξεργαστών τότε η παράλληλη αναζήτηση απαιτεί χρόνο $5T_p + T_c$. Βλέπουμε, λοιπόν, ότι στο συγκεκριμένο παράδειγμα η διαίρεση της γειτονιάς σε δύο μέρη επιταχύνει την αναζήτηση μόνο αν $5T_p + T_c < 9T_p$ ή $T_c < 4T_p$. Αν, δηλαδή, ο χρόνος που απαιτείται για την παραγωγή και εξέταση τεσσάρων προγραμμάτων της γειτονιάς είναι μεγαλύτερος από το χρόνο που απαιτεί η επικοινωνία δύο επεξεργαστών τότε η παραλληλοποίηση της αναζήτησης είναι ωφέλιμη. Σε τόσο μικρά προβλήματα, όπως αυτό του παραδείγματός μας, ο διαμερισμός της γειτονιάς και η παραλληλοποίηση της αναζήτησης μάλλον δεν πρόκειται να την επιταχύνουν, καθώς ο χρόνος επικοινωνίας T_c είναι στην πράξη πολύ μεγαλύτερος από $4T_p$.

5.3 Απόδοση της παραλληλοποίησης

Όπως είδαμε στην προηγούμενη ενότητα, η παράλληλη αναζήτηση δεν ενδείκνυται για μικρά προβλήματα. Για το λόγο αυτό θα εξετάσουμε σε ποιές περιπτώσεις η παραλληλοποίηση της αναζήτησης είναι αποδοτική.

Στη μελέτη μας κάνουμε δύο υποθέσεις :

- Οι διαθέσιμοι επεξεργαστές είναι ίσης δυναμικότητας
- Το πλήθος των περιοχών, στις οποίες διαιρείται η γειτονιά αναζήτησης, είναι ίσο με τον αριθμό των διαθέσιμων επεξεργαστών.

  Αν το πλήθος των περιοχών είναι μικρότερο από τον αριθμό των διαθέσιμων

επεξεργαστών τότε ένα μέρος της υπολογιστικής ισχύος μένει ανεκμετάλλευτο. Αντίθετα, αν είναι μεγαλύτερο τότε η ανισοκατανομή φόρτου εργασίας που εμφανίζεται στους επεξεργαστές έχει ως αποτέλεσμα τη μείωση της απόδοσης της παραλληλοποίησης.

Ο χρόνος που απαιτείται για την εκτέλεση της σειριακής αναζήτησης στη γειτονιά U^S δίνεται από τη σχέση :

$$Y_\sigma = \mid U^S \mid T_p \tag{5.1}$$

όπου $\mid U^S \mid$ είναι το μέγεθος της γειτονιάς U^S.

Ο υπολογιστικός χρόνος Y_π που απαιτείται για την εκτέλεση της παράλληλης αναζήτησης διαιρώντας τη γειτονιά U^S σε E περιοχές δίνεται από τη σχέση :

$$Y_\pi = \lceil \frac{\mid U^S \mid}{E} \rceil T_p + \lceil \log_2 E \rceil T_c \tag{5.2}$$

όπου $\lceil \ \rceil$ η συνάρτηση οροφή (ceil).

Ζητώντας $Y_\pi < Y_\sigma$ και λύνοντας ως προς το λόγο $\frac{T_c}{T_p}$, ο οποίος είναι σταθερός για συγκεκριμένη παράλληλη μηχανή, από τις σχέσεις 5.1 και 5.2 παίρνουμε :

$$\frac{T_c}{T_p} < \frac{\mid U^S \mid - \lceil \frac{\mid U^S \mid}{E} \rceil}{\lceil \log_2 E \rceil} \tag{5.3}$$

Από τη σχέση 5.3 βλέπουμε ότι η επιτάχυνση που επιτυγχάνεται με την παραλληλοποίηση της αναζήτησης εξαρτάται από το μέγεθος της γειτονιάς U^S και από το μέγεθος του διαμερισμού της, δηλαδή τον αριθμό E των διαθέσιμων επεξεργαστών. Για συγκεκριμένο διαμερισμό, δηλαδή σταθερό E, το μέγεθος της γειτονιάς καθορίζει την απόδοση της παραλληλοποίησης. Σε μικρές γειτονιές είναι πιθανό η σχέση 5.3 να μην ικανοποιείται με αποτέλεσμα η σειριακή αναζήτηση να είναι ταχύτερη της παράλληλης, ακόμα και αν η γειτονιά διαιρεθεί σε δύο μόνο μέρη. Στους πίνακες 5.1 και 5.2 φαίνεται η σχέση της σειριακής με την παράλληλη αναζήτηση για διαφορετικά μεγέθη γειτονιών και διαφορετικές διαμερίσεις.

Αριθμός Εργασιών	Μέγεθος Γειτονιάς (προγράμματα)	Αναζήτηση		Σχέση 5.3
		Σειριακή	Παράλληλη	$E = 2$
5	16	$16\,T_p$	$8\,T_p + T_c$	$\frac{T_c}{T_p} < 8$
10	81	$81\,T_p$	$41\,T_p + T_c$	$\frac{T_c}{T_p} < 40$
15	196	$196\,T_p$	$98\,T_p + T_c$	$\frac{T_c}{T_p} < 98$
20	361	$361\,T_p$	$181\,T_p + T_c$	$\frac{T_c}{T_p} < 180$
25	576	$576\,T_p$	$288\,T_p + T_c$	$\frac{T_c}{T_p} < 288$
30	841	$841\,T_p$	$421\,T_p + T_c$	$\frac{T_c}{T_p} < 420$

Πίνακας 5.1: Σχέση σειριακής προς παράλληλη αναζήτηση (Δύο Επεξεργαστές)

Αριθμός Εργασιών	Μέγεθος Γειτονιάς (προγράμματα)	Αναζήτηση		Σχέση 5.3
		Σειριακή	Παράλληλη	$E = 4$
5	16	$16\,T_p$	$4\,T_p + 2\,T_c$	$\frac{T_c}{T_p} < 6$
10	81	$81\,T_p$	$21\,T_p + 2\,T_c$	$\frac{T_c}{T_p} < 30$
15	196	$196\,T_p$	$49\,T_p + 2\,T_c$	$\frac{T_c}{T_p} < 73.5$
20	361	$361\,T_p$	$91\,T_p + 2\,T_c$	$\frac{T_c}{T_p} < 135$
25	576	$576\,T_p$	$144\,T_p + 2\,T_c$	$\frac{T_c}{T_p} < 216$
30	841	$841\,T_p$	$211\,T_p + 2\,T_c$	$\frac{T_c}{T_p} < 315$

Πίνακας 5.2: Σχέση σειριακής προς παράλληλη αναζήτηση (Τέσσερις Επεξεργαστές)

5.4 Οι παράλληλοι αλγόριθμοι

Στην ενότητα αυτή αναπτύσσονται δύο παράλληλοι αλγόριθμοι : ο αλγόριθμος $ParTS_C$, ο οποίος αποτελεί την παράλληλη εκδοχή του αλγορίθμου TS_C που παρουσιάστηκε στην ενότητα 4.8 για τη λύση του προβλήματος $N/\,M/\,F,\ Seq-Ind,\ perm/\,C_{max}$ και ο αλγόριθμος $ParTS_{TC}$, ο οποίος αποτελεί την παράλληλη εκδοχή του αλγορίθμου TS_{TC} που παρουσιάστηκε στην ενότητα 4.9 για τη λύση του προβλήματος $N/\,M/\,F,\ Seq-Ind,\ perm/\,T_{max},\ C_{max}$.

Σε καθέναν από τους δύο παράλληλους αλγορίθμους διακρίνουμε δύο τμήματα : αυτό που εκτελείται σειριακά και εκείνο που εκτελείται παράλληλα. Χρησιμοποιώντας τη συνήθη ορολογία [CG90] ονομάζουμε **συντονιστή (master)** το σειριακό τμήμα και **εργάτη (worker)** το παράλληλο τμήμα. Ο συντονιστής αναλαμβάνει την έναρξη, το συντονισμό και τη λήξη της παράλληλης αναζήτησης, ενώ σε κάθε εργάτη ανατίθεται η εξέταση και ο εντοπισμός του ελαχίστου μιας περιοχής των γειτονιών.

Κατά την έναρξη της αναζήτησης ο συντονιστής παράγει με ευρηματικό τρόπο ένα αρχικό πρόγραμμα και στη συνέχεια συντονίζει την παράλληλη αναζήτηση με τη χρήση μηνυμάτων εργασίας. Δύο είναι τα μηνύματα εργασίας που ο συντονιστής στέλνει στους εργάτες :

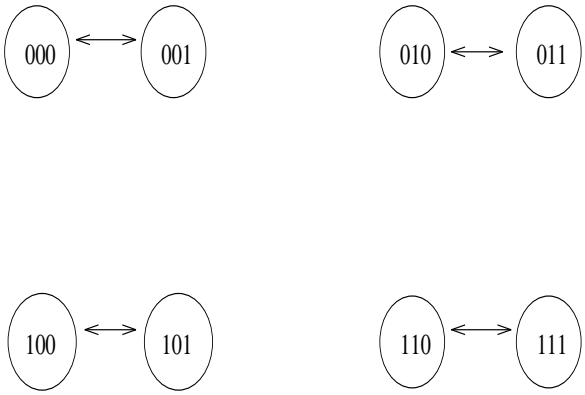
- Το μήνυμα εργασίας ΕΝΑΡΞΗ_ΕΡΓΑΣΙΑΣ.

  Το μήνυμα αυτό στέλνεται στους εργάτες, για να τους ενημερώσει ότι η αναζήτηση συνεχίζεται σε μια νέα γειτονιά. Έπειτα οι εργάτες λαμβάνουν το πρόγραμμα Π_k, η γειτονιά του οποίου πρέπει να εξεταστεί, και τις λίστες TL_C και TL_P, οι οποίες καθορίζουν τις απαγορευμένες κινήσεις. Αυτή είναι όλη η πληροφορία που απαιτείται, για να εξεταστούν τα προγράμματα της γειτονιάς $U^{S_P}(\Pi_k)$.

- Το μήνυμα εργασίας ΛΗΞΗ_ΕΡΓΑΣΙΑΣ.

  Με το μήνυμα αυτό ο συντονιστής πληροφορεί τους εργάτες ότι η αναζήτηση ολοκληρώθηκε, ώστε να τερματίσουν την εκτέλεσή τους.

Η επικοινωνία των εργατών, τόσο μεταξύ τους όσο και με το συντονιστή, διευκολύνεται αν θεωρήσουμε ότι οι εργάτες είναι διατεταγμένοι. Αυτό επιτυγχάνεται με την ανάθεση σε κάθε εργάτη ενός ακέραιου αύξοντα κωδικού ϵ, ο οποίος παίρνει τιμές μεταξύ 0 και $E-1$. Κάθε εργάτης εξετάζει τα προγράμματα που παράγονται από τις



Σχήμα 5.1: Πρώτο βήμα επικοινωνίας

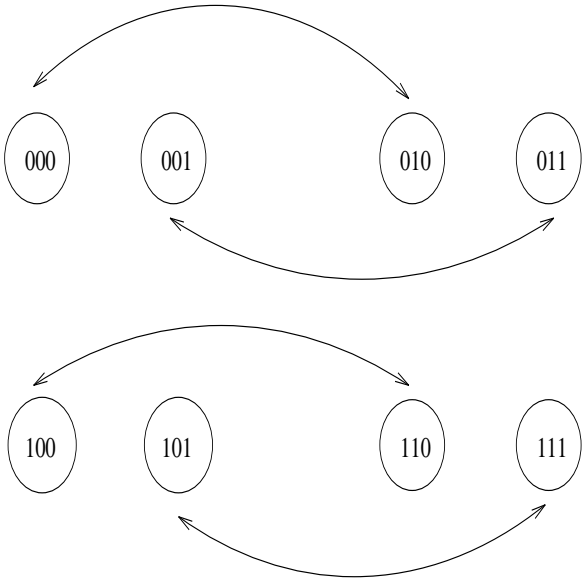
κινήσεις ορισμένων εργασιών. Με τον τρόπο αυτό η γειτονιά U^{S_P} χωρίζεται σε περιοχές $U_\epsilon^{S_P}$, οι οποίες εξετάζονται παράλληλα. (Ισχύει βέβαια ότι $\cup_{\epsilon=0}^{E-1} U_\epsilon^{S_P} = U^{S_P}$).

Όταν η αναζήτηση ολοκληρωθεί σε καθεμιά περιοχή $U_\epsilon^{S_P}$, οι εργάτες απαιτείται να επικοινωνήσουν μεταξύ τους, για να διαπιστωθεί ποιο από τα ελάχιστα των περιοχών είναι το ελάχιστο της γειτονιάς. Η επικοινωνία E το πλήθος εργατών απαιτεί $\log_2 E$ βήματα. Σε κάθε βήμα ένας εργάτης επικοινωνεί με κάποιον άλλο, αν η δυαδική αναπαράσταση των κωδικών τους ϵ διαφέρει κατά ένα ψηφίο. Αυτό θα φανεί καλύτερα με ένα παράδειγμα. Θεωρείστε την επικοινωνία των οκτώ εργατών του σχήματος 5.1, η οποία απαιτεί τρία βήματα. Θα συμβολίσουμε με $(\epsilon)_2$ τη δυαδική αναπαράσταση του κωδικού ϵ. Ο εργάτης 0 $(000)_2$ θα επικοινωνήσει αρχικά με τον εργάτη 1 $(001)_2$. Στο δεύτερο βήμα με τον εργάτη 2 $(010)_2$ (σχήμα 5.2) και στο τρίτο βήμα με τον εργάτη 4 $(100)_2$ (σχήμα 5.3). Με την ολοκλήρωση των τριων βημάτων ο πρώτος εργάτης ($\epsilon = 0$) θα γνωρίζει το αποτέλεσμα της αναζήτησης και θα ενημερώσει το συντονιστή.

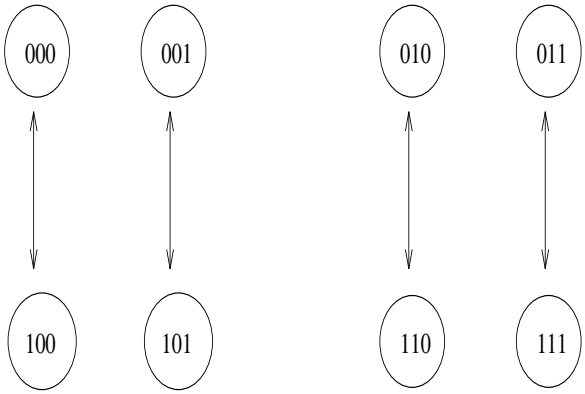
Κατά την αναζήτηση σε μια περιοχή $U_\epsilon^{S_P}$ δύο περιπτώσεις είναι δυνατές : είτε η εύρεση ελαχίστου της περιοχής είτε η εμφάνιση αδιεξόδου. Ο συντονιστής ενημερώνεται για το αποτέλεσμα της αναζήτησης με τη βοήθεια δύο μηνυμάτων εργασίας :

- Το μήνυμα εργασίας ΑΔΙΕΞΟΔΟ_ΑΝΑΖΗΤΗΣΗΣ.

  Το μήνυμα αυτό στέλνεται στο συντονιστή αν κατά την επικοινωνία των εργατών διαπιστωθεί ότι όλοι οι εργάτες έχουν οδηγηθεί σε αδιέξοδο. Στην περίπτωση αυτή ο συντονιστής ολοκληρώνει την αναζήτηση, καθώς ικανοποιείται το πρώτο κριτήριο τερματισμού.



Σχήμα 5.2: Δεύτερο βήμα επικοινωνίας



Σχήμα 5.3: Τρίτο βήμα επικοινωνίας

- Το μήνυμα εργασίας ΕΛΑΧΙΣΤΟ_ΑΝΑΖΗΤΗΣΗΣ.

  Αν ένας τουλάχιστον εργάτης δεν οδηγήθηκε σε αδιέξοδο τότε ο συντονιστής πληροφορείται τον εντοπισμό ελαχίστου γειτονιάς με αυτό το μήνυμα εργασίας. Στη συνέχεια λαμβάνει το ελάχιστο Π_l της γειτονιάς U^{S_P} που εξετάστηκε και την κίνηση K που το παράγει.

Μπορούμε τώρα να παρουσιάσουμε τους δύο αλγορίθμους.

5.4.1 Ο αλγόριθμος $ParTS_C$

Είσοδος Το πλήθος N των εργασιών, το πλήθος M των μηχανών, οι χρόνοι p επεξεργασίας των εργασιών, οι χρόνοι s εξάρμωσης των μηχανών και οι ομάδες των εργασιών.

Παράμετροι Η ευρηματική μέθοδος για την παραγωγή αρχικού προγράμματος, ο μέγιστος δυνατός αριθμός επαναλήψεων max_k, ο μέγιστος δυνατός αριθμός επαναλήψεων max_t από την τελευταία φορά που βελτιώθηκε η λύση, το μέγιστο πλήθος C και P στοιχείων που μπορούν να έχουν οι λίστες TL_C και TL_P αντίστοιχα, και το πλήθος E των διαθέσιμων επεξεργαστών.

Έξοδος Το πρόγραμμα $\Pi_{\beta\varepsilon\lambda\tau}$.

Σειριακό Τμήμα Αλγορίθμου - Συντονιστής (Master)

Βήμα 1 Δημιούργησε με ευρηματικό τρόπο ένα αρχικό πρόγραμμα Π_0.

Θέσε $\Pi_{\beta\varepsilon\lambda\tau} = \Pi_0$.

Βήμα 2 Θέσε

- $TL_C = \emptyset$.
- $TL_P = \emptyset$.
- $k = 0$.
- $t = 0$.

Βήμα 3 Δημιούργησε E το πλήθος εργάτες και στείλε σε καθέναν τον αύξοντα κωδικό ϵ ($0 \leq \epsilon < E$).

Βήμα 4 Στείλε σε κάθε εργάτη :

- Το μήνυμα εργασίας ΕΝΑΡΞΗ_ΕΡΓΑΣΙΑΣ.
- Το πρόγραμμα Π_k.
- Τη λίστα TL_C.
- Τη λίστα TL_P.

Βήμα 5 Λάβε από τον πρώτο εργάτη το μήνυμα εργασίας.

Βήμα 6 Αν το μήνυμα εργασίας είναι το ΑΔΙΕΞΟΔΟ_ΑΝΑΖΗΤΗΣΗΣ τότε πήγαινε στο βήμα 9.

Αλλιώς, λάβε από τον πρώτο εργάτη :

- Το πρόγραμμα Π_l, ελάχιστο της γειτονιάς $U^{S_P}(\Pi_k)$.
- Την κίνηση K_{ij}, με την οποία παράγεται το πρόγραμμα Π_l.

Βήμα 7 Αν $C_{max}(\Pi_l) < C_{max}(\Pi_{\text{βελτ}})$ τότε

- Θέσε $\Pi_{\text{βελτ}} = \Pi_l$.
- Θέσε $t = k$.

Αλλιώς, αν $C_{max}(\Pi_l) > C_{max}(\Pi_k)$:

Τοποθέτησε στη θέση $k \, mod \, C$ της λίστας TL_C το στοιχείο $C_{max}(\Pi_l)$.

Βήμα 8 Αν $k \geq max\_k$ ή $k - t \geq max\_t$ τότε πήγαινε στο βήμα 9.

Αλλιώς :

- Έστω $K_{ij}(\Pi_k) = \Pi_l$. Τοποθέτησε στη θέση $k \, mod \, P$ της λίστας TL_P το στοιχείο i.
- Θέσε $\Pi_{k+1} = \Pi_l$.
- Θέσε $k = k + 1$.
- Πήγαινε στο βήμα 4.

Βήμα 9 Στείλε σε κάθε εργάτη το μήνυμα εργασίας ΛΗΞΗ_ΕΡΓΑΣΙΑΣ και τερμάτισε.

Παράλληλο Τμήμα Αλγορίθμου - Εργάτης (Worker)

Βήμα 1 Λάβε από το συντονιστή τον αύξοντα κωδικό ϵ.

Βήμα 2 Λάβε από το συντονιστή το μήνυμα εργασίας.

Βήμα 3 Αν το μήνυμα εργασίας είναι το ΛΗΞΗ_ΕΡΓΑΣΙΑΣ τότε τερμάτισε.

Αλλιώς, λάβε από το συντονιστή :

- Το πρόγραμμα Π_k.
- Τη λίστα TL_C.
- Τη λίστα TL_P.

Βήμα 4 Αν $U_\epsilon^{S_P}(\Pi_k) - TL_\Pi = \emptyset$ τότε πήγαινε στο βήμα 5.

Αλλιώς :

Βρες $\Pi_l^\epsilon \in U_\epsilon^{S_P}(\Pi_k) - TL_\Pi$, τέτοιο ώστε

$$C_{max}(\Pi_l^\epsilon) < C_{max}(\Pi)\ , \quad \forall \Pi \in U_\epsilon^{S_P}(\Pi_k) - TL_\Pi$$

Αν το πρόγραμμα Π_l^ϵ δεν είναι μοναδικό τότε διάλεξε αυθαίρετα το πρώτο που θα βρεθεί.

Βήμα 5 Επικοινώνησε με τους άλλους εργάτες, για να διαπιστωθεί ποιος έχει το ελάχιστο της γειτονιας U^{S_P} ή αν η αναζήτηση έχει φτάσει σε αδιέξοδο (πρώτο κριτήριο τερματισμού).

Βήμα 6 Αν είσαι ο πρώτος εργάτης τότε :

Αν ικανοποιείται το πρώτο κριτήριο τερματισμού, δηλαδή ισχύει ότι :

$$\cup_{\epsilon=0}^{E-1} U_\epsilon^{S_P} \ - \ TL_\Pi \ = \ \emptyset$$

τότε στείλε στο συντονιστή το μήνυμα εργασίας ΑΔΙΕΞΟΔΟ_ΑΝΑΖΗΤΗΣΗΣ.

Αλλιώς, στείλε στο συντονιστή :

- Το μήνυμα εργασίας ΕΛΑΧΙΣΤΟ_ΑΝΑΖΗΤΗΣΗΣ.
- Το πρόγραμμα Π_l, όπου

$$C_{max}(\Pi_l) \ = \ \min_{0 \le \epsilon < E} C_{max}(\Pi_l^\epsilon)$$

- Την κίνηση K_{ij}, με την οποία παράγεται το πρόγραμμα Π_l.

Βήμα 7 Πήγαινε στο βήμα 2.

5.4.2 Ο αλγόριθμος $ParTS_{TC}$

Είσοδος Το πλήθος N των εργασιών, το πλήθος M των μηχανών, οι χρόνοι p επεξεργασίας των εργασιών, οι χρόνοι s εξάρμωσης των μηχανών, οι ομάδες και οι προθεσμίες των εργασιών.

Παράμετροι Ο μέγιστος δυνατός αριθμός επαναλήψεων max_k, ο μέγιστος δυνατός αριθμός επαναλήψεων max_t από την τελευταία φορά που βελτιώθηκε η λύση, το μέγιστο πλήθος TC και P που μπορούν να έχουν οι λίστες TL_{TC} και TL_P αντίστοιχα, και το πλήθος E των διαθέσιμων επεξεργαστών.

Έξοδος Το πρόγραμμα $\Pi_{\beta\varepsilon\lambda\tau}$.

Σειριακό Τμήμα Αλγορίθμου - Συντονιστής (Master)

Βήμα 1 Δημιούργησε με τη μέθοδο EDD ένα αρχικό πρόγραμμα Π_0.

Θέσε $\Pi_{\beta\varepsilon\lambda\tau} = \Pi_0$.

Βήμα 2 Θέσε

- $TL_{TC} = \emptyset$.
- $TL_P = \emptyset$.
- $k = 0$.
- $t = 0$.

Βήμα 3 Δημιούργησε E το πλήθος εργάτες και στείλε σε καθέναν τον αύξοντα κωδικό ϵ ($0 \leq \epsilon < E$).

Βήμα 4 Στείλε σε κάθε εργάτη :

- Το μήνυμα εργασίας ΕΝΑΡΞΗ_ΕΡΓΑΣΙΑΣ.
- Το πρόγραμμα Π_k.
- Τη λίστα TL_{TC}.
- Τη λίστα TL_P.

Βήμα 5 Λάβε από τον πρώτο εργάτη το μήνυμα εργασίας.

Βήμα 6 Αν το μήνυμα εργασίας είναι το ΑΔΙΕΞΟΔΟ_ΑΝΑΖΗΤΗΣΗΣ τότε πήγαινε στο βήμα 9.

Αλλιώς, λάβε από τον πρώτο εργάτη :

- Το πρόγραμμα Π_l, ελάχιστο της γειτονιάς $U^{S_P}(\Pi_k)$.
- Την κίνηση K_{ij}, με την οποία παράγεται το πρόγραμμα Π_l.

Βήμα 7 Αν $T_{max}(\Pi_l) < T_{max}(\Pi_{\text{βελτ}})$ τότε

- Θέσε $\Pi_{\text{βελτ}} = \Pi_l$.
- Θέσε $t = k$.

Αλλιώς, αν $T_{max}(\Pi_l) = T_{max}(\Pi_{\text{βελτ}})$ και $C_{max}(\Pi_l) < C_{max}(\Pi_{\text{βελτ}})$ τότε

- Θέσε $\Pi_{\text{βελτ}} = \Pi_l$.
- Θέσε $t = k$.

Αλλιώς, αν $T_{max}(\Pi_l) > T_{max}(\Pi_k)$:

Τοποθέτησε στη θέση $k\, mod TC$ της λίστας TL_{TC} το ζεύγος $(T_{max}(\Pi_l), C_{max}(\Pi_l))$.

Βήμα 8 Αν $k \geq max\_k$ ή $k - t \geq max\_t$ τότε πήγαινε στο βήμα 9.

Αλλιώς :

- Έστω $K_{ij}(\Pi_k) = \Pi_l$. Τοποθέτησε στη θέση $k\, mod\, P$ της λίστας TL_P το στοιχείο i.
- Θέσε $\Pi_{k+1} = \Pi_l$.
- Θέσε $k = k + 1$.
- Πήγαινε στο βήμα 4.

Βήμα 9 Στείλε σε κάθε εργάτη το μήνυμα εργασίας ΛΗΞΗ_ΕΡΓΑΣΙΑΣ και τερμάτισε.

Παράλληλο Τμήμα Αλγορίθμου - Εργάτης (Worker)

Βήμα 1 Λάβε από το συντονιστή τον αύξοντα κωδικό ϵ.

Βήμα 2 Λάβε από το συντονιστή το μήνυμα εργασίας.

Βήμα 3 Αν το μήνυμα εργασίας είναι το ΛΗΞΗ_ΕΡΓΑΣΙΑΣ τότε τερμάτισε.

Αλλιώς, λάβε από το συντονιστή :

- Το πρόγραμμα Π_k.
- Τη λίστα TL_{TC}.
- Τη λίστα TL_P.

Βήμα 4 Αν $U_{\epsilon}^{S_P}(\Pi_k) - TL_{\Pi} = \emptyset$ τότε πήγαινε στο βήμα 5.

Αλλιώς :

Βρες $\Pi_l^{\epsilon} \in U_{\epsilon}^{S_P}(\Pi_k) - TL_{\Pi}$, τέτοιο ώστε

$$T_{max}(\Pi_l^{\epsilon}) < T_{max}(\Pi)\ , \quad \forall \Pi \in U_{\epsilon}^{S_P}(\Pi_k) - TL_{\Pi}$$

Αν το πρόγραμμα Π_l^{ϵ} δεν είναι μοναδικό τότε διάλεξε αυτό με τον ελάχιστο χρόνο περάτωσης C_{max}. Αν πάλι το πρόγραμμα αυτό δεν είναι μοναδικό τότε διάλεξε αυθαίρετα το πρώτο που θα βρεθεί.

Βήμα 5 Επικοινώνησε με τους άλλους εργάτες, για να διαπιστωθεί ποιος έχει το ελάχιστο της γειτονιας U^{S_P} ή αν η αναζήτηση έχει φτάσει σε αδιέξοδο (πρώτο κριτήριο τερματισμού).

Βήμα 6 Αν είσαι ο πρώτος εργάτης τότε :

Αν ικανοποιείται το πρώτο κριτήριο τερματισμού, δηλαδή ισχύει ότι :

$$\cup_{\epsilon=0}^{E-1} U_{\epsilon}^{S_P} \ - \ TL_{\Pi} \ = \ \emptyset$$

τότε στείλε στο συντονιστή το μήνυμα εργασίας ΑΔΙΕΞΟΔΟ_ΑΝΑΖΗΤΗΣΗΣ.

Αλλιώς, στείλε στο συντονιστή :

- Το μήνυμα εργασίας ΕΛΑΧΙΣΤΟ_ΑΝΑΖΗΤΗΣΗΣ.
- Το πρόγραμμα Π_l, όπου

$$(T_{max}(\Pi_l), C_{max}(\Pi_l)) \ = \ \min_{0 \leq \epsilon < E} (T_{max}(\Pi_l^{\epsilon}), C_{max}(\Pi_l^{\epsilon}))$$

- Την κίνηση K_{ij}, με την οποία παράγεται το πρόγραμμα Π_l.

Βήμα 7 Πήγαινε στο βήμα 2.

Κεφάλαιο 6

Δενδροειδής Αναζήτηση

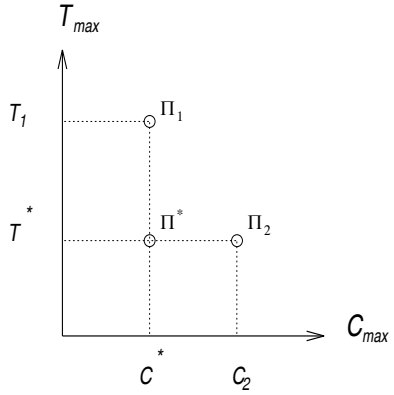
6.1 Πολλαπλά τοπικά ελάχιστα

Η απόδοση των αλγορίθμων που παρουσιάστηκαν και ειδικά η ποιότητα των λύσεων που παράγονται εξαρτάται από το πλήθος των τοπικών ελαχίστων που υπάρχουν στις γειτονιές αναζήτησης. Αυτό θα φανεί καλύτερα με ένα παράδειγμα. Έστω ένα πρόγραμμα Π. Αν στη γειτονιά του προγράμματος Π υπάρχει ένα μοναδικό πρόγραμμα Π_1, του οποίου ο χρόνος περάτωσης $C_{max}(\Pi_1)$ είναι ο ελάχιστος μεταξύ των χρόνων των άλλων προγραμμάτων της γειτονιάς τότε η αναζήτηση θα συνεχιστεί στη γειτονιά του Π_1. Αν, όμως, το τοπικό ελάχιστο της γειτονιάς δεν είναι μοναδικό τότε θα υπάρχει τουλάχιστον ένα ακόμη πρόγραμμα Π_2, για το οποίο θα ισχύει ότι $C_{max}(\Pi_1) = C_{max}(\Pi_2)$. Στην περίπτωση αυτή υπάρχουν δύο ισοδύναμες κατευθύνσεις για τη συνέχιση της αναζήτησης, η εξέταση των προγραμμάτων της γειτονιάς του Π_1 και της γειτονιάς του Π_2. Καθώς μόνο με τη σύγκριση των χρόνων περάτωσης των δύο προγραμμάτων δεν είναι δυνατό να διαπιστωθεί ποια από τις δύο γειτονιές είναι καλύτερη, μία από τις δύο θα επιλεχθεί με τυχαίο τρόπο. Ο τυχαίος τρόπος επιλογής δεν εξασφαλίζει, όμως, ότι η αναζήτηση συνεχίζεται προς την καλύτερη κατεύθυνση. Πράγματι, έστω ότι τυχαία επιλέγεται η γειτονιά του προγράμματος Π_1 και έστω Π_1' το τοπικό ελάχιστο της γειτονιάς αυτής. Αν Π_2' το τοπικό ελάχιστο της γειτονιάς του Π_2 τότε ενδέχεται $C_{max}(\Pi_2') < C_{max}(\Pi_1')$.

Η απροσδιοριστία στην επιλογή κατεύθυνσης λόγω της ύπαρξης πολλαπλών τοπικών ελαχίστων στις γειτονιές αναζήτησης εμφανίζεται ιδιαίτερα έντονη στο μονοκριτηριακό πρόβλημα $N/\ M/F,\ Seq - Ind,\ perm/C_{max}$. Η ύπαρξη προθεσμιών παράδοσης και

η εισαγωγή της μέγιστης καθυστέρησης T_{max} των εργασιών ως δεύτερου κριτηρίου βελτίστου έχει ως αποτέλεσμα μεγαλύτερη διασπορά των προγραμμάτων στο χώρο έρευνας S_N και καλύτερη διακριτοποίηση του χώρου αυτού.

Ο χώρος έρευνας του δικριτηριακού προβλήματος εμφανίζεται περισσότερο δομημένος από τον αντίστοιχο χώρο του μονοκριτηριακού προβλήματος. Ένα πρόγραμμα Π^* είναι τοπικό ελάχιστο μιας γειτονιάς του δικριτηριακού χώρου αν ικανοποιεί τις συνθήκες : $T_{max}(\Pi^*) \leq T_{max}(\Pi)$ και $C_{max}(\Pi^*) \leq C_{max}(\Pi)$ για οποιοδήποτε πρόγραμμα Π της γειτονιάς. Έστω ένα πρόγραμμα Π_1, όπως αυτό του σχήματος 6.1, για το οποίο ισχύει ότι $C_{max}(\Pi^*) = C_{max}(\Pi_1)$. Αφού τα δύο προγράμματα έχουν τον ίδιο χρόνο περάτωσης και ο χρόνος αυτός είναι ελάχιστος στη γειτονιά στην οποία περιέχονται, στη γειτονιά αυτή του μονοκριτηριακού χώρου υπάρχει διπλό τοπικό ελάχιστο. Όμως, στο δικριτηριακό χώρο το πρόγραμμα Π^* είναι το μοναδικό τοπικό ελάχιστο της γειτονιάς, αφού $T_{max}(\Pi^*) < T_{max}(\Pi_1)$.

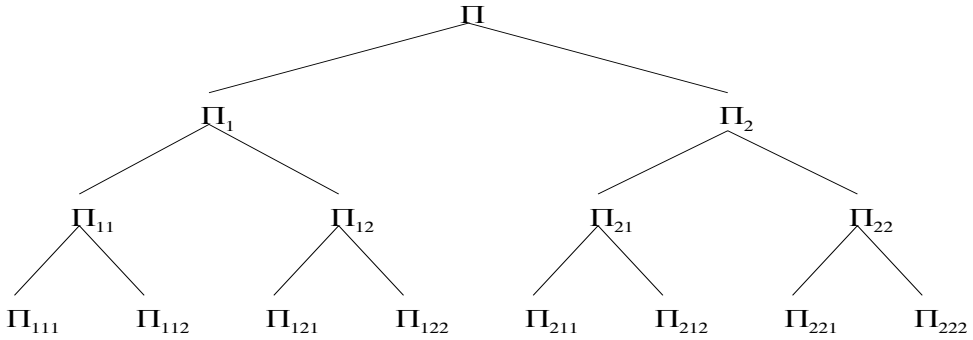


Σχήμα 6.1: Τοπικό ελάχιστο στο δικριτηριακό χώρο

6.2 Δενδροειδής αναζήτηση

Έστω ένα πρόγραμμα Π, στη γειτονιά $U^S(\Pi)$ του οποίου υπάρχουν δύο προγράμματα Π_1 και Π_2, των οποίων ο χρόνος περάτωσης είναι τοπικά ελάχιστος. Για να συνεχιστεί η αναζήτηση προς την καλύτερη κατεύθυνση απαιτείται να αξιολογηθούν οι γειτονιές που παράγονται από τα προγραμμάτα Π_1 και Π_2. Επομένως, είναι απαραίτητη η εξέταση των προγραμμάτων που περίεχονται και στις δύο γειτονιές. Αν Π_1' και Π_2' είναι αντίστοιχα τα τοπικά ελάχιστα στις γειτονιές των Π_1 και Π_2 τότε η σύγκριση των χρόνων περάτωσης των δύο τοπικών ελαχίστων είναι συνήθως αρκετή, για να αποφασιστεί σε ποια γειτονιά θα στραφεί η αναζήτηση. Για παράδειγμα, αν $C_{max}(\Pi_1') < C_{max}(\Pi_2')$ τότε θα επιλεχθεί

η γειτονιά του Π'_1. Επειδή με τον τρόπο αυτό εξετάζεται ένα σύνολο γειτονιών, το οποίο σχηματίζει δένδρο (σχήμα 6.2), η αναζήτηση ονομάζεται δενδροειδής.



Σχήμα 6.2: Δενδροειδής Αναζήτηση

Η απόδοση της δενδροειδούς αναζήτησης εξαρτάται από δύο παραμέτρους :

- Το βάθος της αναζήτησης

  Θεωρείστε το παράδειγμα του σχήματος 6.2, όπου στη γειτονιά του Π υπάρχουν δύο τοπικά ελάχιστα, τα προγράμματα Π_1 και Π_2 , και Π_{11}, Π_{21} είναι αντίστοιχα τα τοπικά ελάχιστα στις γειτονιές των Π_1 και Π_2. Αν $C_{max}(\Pi_{11}) = C_{max}(\Pi_{21})$ τότε η εξέταση των προγραμμάτων στις γειτονιές των Π_1 και Π_2 αποδεικνύεται ανεπαρκής, για να διαπιστωθεί η καλύτερη κατεύθυνση αναζήτησης. Στην περίπτωση αυτή είναι απαραίτητη η εξέταση των γειτονιών $U^S(\Pi_{11})$ και $U^S(\Pi_{21})$. Αν η αναζήτηση στις γειτονιές $U^S(\Pi_{11})$ και $U^S(\Pi_{21})$ οδηγηθεί πάλι σε πολλαπλά τοπικά ελάχιστα τότε η δενδροειδής αναζήτηση θα συνεχιστεί στις νέες γειτονιές που προκύπτουν μέχρι η εύρεση μοναδικού τοπικού ελαχίστου να κρίνει προς ποια κατεύθυνση θα στραφεί η αναζήτηση. Επειδή η διαδικασία αυτή είναι χρονοβόρος δεν μπορεί να επαναλαμβάνεται πολλές φορές. Ο μέγιστος δυνατός αριθμός επαναλήψεων καθορίζει το βάθος της αναζήτησης.

  Το βάθος είναι η πρώτη κρίσιμη παράμετρος της δενδροειδούς αναζήτησης. Μεγάλο βάθος εξασφαλίζει καλύτερη αξιολόγηση των δυνατών κατευθύνσεων απαιτεί, όμως, ιδιαίτερα μεγάλο υπολογιστικό χρόνο. Αντίθετα μικρό βάθος ενδέχεται να αποδειχθεί ανεπαρκές για να διαπιστωθεί η καλύτερη κατεύθυνση συνέχιση της αναζήτησης.

- Το πλάτος της αναζήτησης

  Έστω $TE = \{ \Pi^* \in U^S(\Pi) \mid C_{max}(\Pi^*) \leq C_{max}(\Pi'), \quad \forall \Pi' \in U^S(\Pi) \}$, το σύνολο όλων των τοπικών ελαχίστων της γειτονιάς του προγράμματος Π. Η δενδροειδής

αναζήτηση θα εξετάσει όλες τις γειτονιές των προγραμμάτων του TE. Επειδή, όμως, η αναζήτηση σε μια γειτονιά είναι εξαιρετικά χρονοβόρος δεν είναι πάντοτε δυνατό να εξετάζονται όλες οι γειτονιές των τοπικών ελαχίστων, ιδιαίτερα όταν το πλήθος τους είναι μεγάλο. Στην περίπτωση αυτή επιλέγεται ένα υποσύνολο TE_b του TE και η αναζήτηση συνεχίζεται στις γειτονιές των προγραμμάτων του υποσυνόλου αυτού. Το μέγεθος του TE_b καθορίζει το πλάτος της δενδροειδούς αναζήτησης.

Το πλάτος αποτελεί τη δεύτερη κρίσιμη παράμετρο της δενδροειδούς αναζήτησης. Μεγάλο πλάτος εξασφαλίζει καλύτερες τελικές λύσεις απαιτώντας, όμως, υπερβολικά μεγάλο υπολογιστικό χρόνο, αφού εξετάζεται μεγάλος αριθμός γειτονιών. Αντίθετα η επιλογή μικρού πλάτους έχει ως αποτέλεσμα τον αποκλεισμό κάποιων τοπικών ελαχίστων από την αναζήτηση με πιθανό κίνδυνο να μην εξετάζονται πάντοτε οι καλύτερες κατευθύνσεις αναζήτησης.

6.3 Παράλληλη δενδροειδής αναζήτηση

Έστω η δενδροειδής αναζήτηση του σχήματος 6.2 με πλάτος δύο και βάθος τέσσερα. Το δένδρο που σχηματίζεται αποτελείται από δεκαπέντε κόμβους, με αποτέλεσμα με τη δενδροειδή αναζήτηση να εξετάζονται δεκαπέντε γειτονιές. Αν η εξέταση των γειτονιών αυτών εκτελεσθεί σειριακά τότε ο υπολογιστικός χρόνος που θα απαιτηθεί για την αναζήτηση ισούται με $15 \mid U^S \mid T_p$, όπου T_p είναι ο χρόνος που απαιτείται για την παραγωγή ενός προγράμματος της γειτονιάς. Επειδή αυτός ο υπολογιστικός χρόνος είναι εξαιρετικά μεγάλος, θα εξετάσουμε τη δυνατότητα παραλληλοποίησης της δενδροειδούς αναζήτησης.

Από το σχήμα 6.2 παρατηρούμε ότι δεν είναι δυνατή η παράλληλη εξέταση και των δεκαπέντε γειτονιών. Έστω, για παράδειγμα, η γειτονιά $U^S(\Pi)$, στην οποία υπάρχουν δύο ελάχιστα, τα προγράμματα Π_1 και Π_2. Στην περίπτωση αυτή η δενδροειδής αναζήτηση θα συνεχιστεί εξετάζοντας τα προγράμματα που περιέχονται στις γειτονιές $U^S(\Pi_1)$ και $U^S(\Pi_2)$. Είναι φανερό ότι ο σχηματισμός και η αναζήτηση στις δύο αυτές γειτονιές απαιτεί την ολοκλήρωση της εξέτασης της γειτονιάς $U^S(\Pi)$, καθώς μόνο τότε είναι γνωστά τα προγράμματα Π_1 και Π_2, τα οποία παράγουν τις γειτονιές $U^S(\Pi_1)$ και $U^S(\Pi_2)$. Βλέπουμε, λοιπόν, ότι η αναζήτηση δεν είναι δυνατό να εκτελεσθεί παράλληλα στις γειτονιές $U^S(\Pi)$, $U^S(\Pi_1)$ και $U^S(\Pi_2)$, καθώς τα προγράμματα των δύο τελευταίων εξαρτώνται από τα αποτελέσματα της αναζήτησης στην πρώτη. Δεν ισχύει, όμως, το

ίδιο και για τις γειτονιές $U^S(\Pi_1)$, $U^S(\Pi_2)$, οι οποίες είναι ανεξάρτητες, με αποτέλεσμα η αναζήτηση σε αυτές να μπορεί να εκτελεσθεί παράλληλα.

Τα συμπεράσματα του παραδείγματος αυτού γενικεύονται εύκολα :

Συμπέρασμα 6.1 Η αναζήτηση σε γειτονιές που βρίσκονται σε κάποιο επίπεδο απαιτεί την ολοκλήρωση της αναζήτησης γειτονιών του αμέσως προηγούμενου επιπέδου.

Συμπέρασμα 6.2 Η αναζήτηση σε γειτονιές που βρίσκονται στο ίδιο επίπεδο μπορεί να εκτελεσθεί παράλληλα.

6.4 Απόδοση της παράλληλης δενδροειδούς αναζήτησης

Αν b το πλάτος και d το βάθος δενδροειδούς αναζήτησης τότε στο πρώτο επίπεδο εξετάζεται μια γειτονιά, στο δεύτερο επίπεδο εξετάζονται b τον αριθμό γειτονιές, στο τρίτο b^2 και γενικά στο επίπεδο d εξετάζονται b^{d-1} τον αριθμό γειτονιές. Συμβολίζοντας $E(b,d)$ το συνολικό αριθμό γειτονιών που εξετάζονται κατά την αναζήτηση αυτή τότε

$$E(b,d) \;=\; \sum_{k=1}^{d} b^{k-1} \;=\; \frac{1-b^d}{1-b} \tag{6.1}$$

Επομένως, ο χρόνος Y_σ που απαιτεί η σειριακή αναζήτηση δίνεται από την ακόλουθη σχέση :

$$Y_\sigma \;=\; E(b,d) \mid U^S(\Pi) \mid T_p \tag{6.2}$$

Αν ο αριθμός των διαθέσιμων επεξεργαστών είναι ίσος με το πλήθος των γειτονιών, τότε η παραλληλοποίηση της δενδροειδούς αναζήτησης είναι άμεση, καθώς σε κάθε επεξεργαστή ανατίθεται η εξέταση μιας γειτονιάς. Καθώς η αναζήτηση μπορεί να εκτελεσθεί παράλληλα μόνο σε γειτονιές που βρίσκονται στο ίδιο επίπεδο, ο χρόνος που απαιτείται γαι την εξέταση όλων των γειτονιών θα είναι ίσος με $d \mid U^S(\Pi) \mid T_p$. Οι επεξεργαστές που έχουν αναλάβει την εξέταση γειτονιών διαδοχικών επιπέδων απαιτείται να επικοινωνήσουν μεταξύ τους στην αρχή και στο τέλος της αναζήτησης. Την πρώτη φορά για να αναθέσουν στους επεξεργαστές του επόμενου επιπέδου τη γειτονιά που θα εξετάσουν, ενώ τη δεύτερη για να λάβουν το αποτέλεσμα της εξέτασης αυτής. Μπορούμε

να θεωρήσουμε ότι οι επεξεργαστές αναθέτουν παράλληλα τις προς εξέταση γειτονιές στους επεξεργαστές του αμέσως επόμενου επιπέδου. Όμως, καθένας επεξεργαστής αναθέτει σειριακά τις b τον αριθμό γειτονιές που παράγει στους b επεξεργαστές του επόμενου επιπέδου, οι οποίοι του αναλογούν. Με τον ίδιο τρόπο θεωρούμε ότι πραγματοποιείται η επικοινωνία και στην περίπτωση που λαμβάνονται τα αποτελέσματα της αναζήτησης. Επομένως, σε κάθε επίπεδο απαιτείται επικοινωνιακός χρόνος $2bT_c$ και συνολικά $2b(d-1)T_c$. Στην περίπτωση αυτή ο υπολογιστικός χρόνος Y_π που απαιτεί η εκτέλεση της παράλληλης αναζήτησης ισούται με :

$$Y_\pi = d \mid U^S \mid T_p + 2b(d-1)T_c \tag{6.3}$$

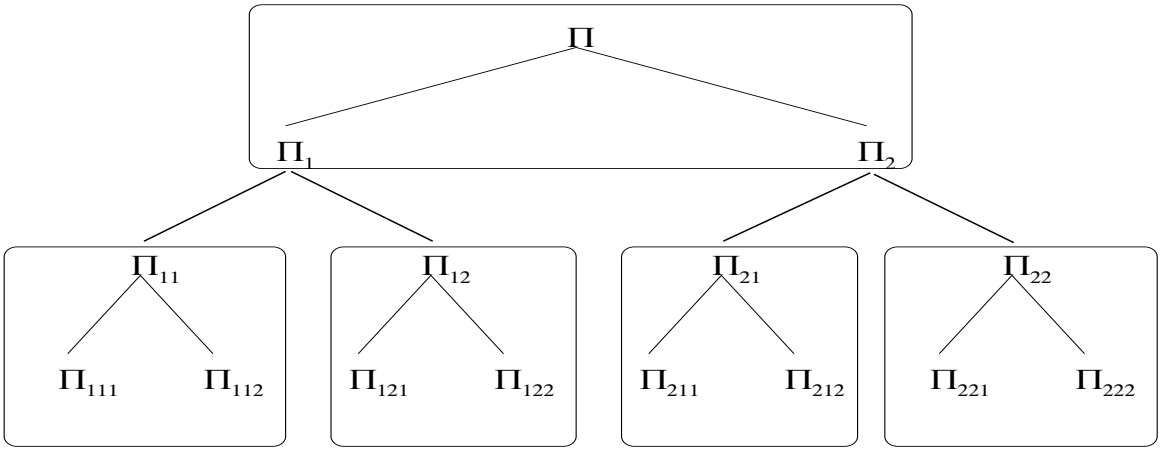
Από τις σχέσεις 6.2 και 6.3 προκύπτει ότι $Y_\pi < Y_\sigma$ αν

$$\frac{T_c}{T_p} < \frac{E(b,d)-d}{2b(d-1)} \mid U^S \mid \tag{6.4}$$

Η σχέση 6.4 ισχύει αν το πλήθος των διαθέσιμων επεξεργαστών είναι ίσο με το πλήθος των γειτονιών που εξετάζονται με τη δενδροειδή αναζήτηση. Το πλήθος αυτό, όμως, είναι αρκετά μεγάλο ακόμα και σε συνδυασμούς μικρών τιμών των παραμέτρων d και b της αναζήτησης. Για παράδειγμα, στη δενδροειδή αναζήτηση του σχήματος 6.2, όπου το βάθος είναι τέσσερα και το πλάτος δύο, το πλήθος των γειτονιών που εξετάζονται είναι ίσο με δεκαπέντε. Αν και στις μέρες μας υπάρχουν παράλληλες μηχανές μερικών εκατοντάδων επεξεργαστών, σε αρκετές περιπτώσεις ενδέχεται οι διαθέσιμοι επεξεργαστές να είναι λιγότεροι από δεκαπέντε. Στην περίπτωση αυτή σε κάθε επεξεργαστή ανατίθεται η εξέταση μιας ομάδας γειτονιών αντί μίας μόνο γειτονιάς. Δυστυχώς όμως, ο σχηματισμός τέτοιων ομάδων εμφανίζεται ιδιαίτερα δύσκολος με αποτέλεσμα να μην υπάρχει σαφής μέθοδος για το σκοπό αυτό. Γενικά επιδιώκεται ο σχηματισμός ομάδων κατά τέτοιον τρόπο ώστε να ισοκατανέμεται ο φόρτος εργασίας στους επεξεργαστές, ενώ ταυτόχρονα να ελαχιστοποιείται ο χρόνος που απαιτείται για την επικοινωνία τους.

Για παράδειγμα, θεωρείστε ότι για την αναζήτηση του σχήματος 6.2 διαθέτουμε πέντε μόνο επεξεργαστές. Η απαίτηση για ισοκατανομή του φόρτου εργασίας στους επεξεργαστές μας οδηγεί στο σχηματισμό ομάδων των τριων γειτονιών. Αφού επικοινωνία απαιτείται μόνο μεταξύ επεξεργαστών που εξετάζουν γειτονιές που βρίσκονται σε

διαφορετικά επίπεδα, είναι λογικό να επιδιώξουμε την ομαδοποίηση τέτοιων γειτονιών, όπως φαίνεται στο σχήμα 6.3.



Σχήμα 6.3: Παράλληλη Δενδροειδής Αναζήτηση (Πέντε Επεξεργαστές)

Ο υπολογιστικός χρόνος που απαιτεί η δενδροειδής αναζήτηση του σχήματος 6.3 ισούται με $Y_\pi(5) = 6 \mid U^S \mid T_p + 8T_c$. Αν διαθέταμε δεκαπέντε επεξεργαστές τότε η αναζήτηση αυτή θα απαιτούσε χρόνο ίσο με $Y_\pi(15) = 4 \mid U^S \mid T_p + 12T_c$.

6.5 Αλγόριθμος παράλληλης δενδροειδούς αναζήτησης

Μπορούμε τώρα να παρουσιάσουμε τον αλγόριθμο $ParTrTS_C$:

Είσοδος Το πλήθος N των εργασιών, το πλήθος M των μηχανών, οι χρόνοι επεξεργασίας των εργασιών και εξάρμωσης των μηχανών και οι ομάδες των εργασιών.

Παράμετροι Η ευρηματική μέθοδος για την παραγωγή αρχικού προγράμματος, το βάθος d και το πλάτος b της δενδροειδούς αναζήτησης, ο μέγιστος δυνατός αριθμός επαναλήψεων max_k, ο μέγιστος δυνατός αριθμός επαναλήψεων max_t από την τελευταία φορά που βελτιώθηκε η λύση και τα μεγέθη C και P των TL_C και TL_P.

Έξοδος Το πρόγραμμα $\Pi_{\beta\varepsilon\lambda\tau}$.

Συντονιστής (Master)

Βήμα 1 Δημιούργησε με ευρηματικό τρόπο ένα αρχικό πρόγραμμα Π_0.

Θέσε $\Pi_{\beta\epsilon\lambda\tau} = \Pi_0$.

Βήμα 2 Θέσε

- $TL_C = \emptyset$.
- $TL_P = \emptyset$.
- $k = 0$.
- $t = 0$.
- $L = 1$.

Βήμα 3 Δημιούργησε b το πλήθος εργάτες-παιδιά και στείλε σε καθέναν

- Τον αύξοντα κωδικό ϵ ($0 \leq \epsilon < b$).
- Το βάθος d της δενροειδούς αναζήτησης.
- Το βάθος L ($0 \leq L \leq d$) της δενδροειδούς αναζήτησης, στο οποίο τοποθετούνται.

Βήμα 4 Αν $U^{S_P}(\Pi_k) - TL_\Pi = \emptyset$ τότε πήγαινε στο βήμα 12.

Αλλιώς :

Βρες b το πλήθος $\Pi_l^\epsilon \in U^{S_P}(\Pi_k) - TL_\Pi$, τέτοια ώστε

$$C_{max}(\Pi_l^\epsilon) < C_{max}(\Pi)\ , \quad \forall \Pi \in U^{S_P}(\Pi_k) - TL_\Pi$$

Αν υπάρχουν περισσότερα από b το πλήθος ελάχιστα Π_l^ϵ γειτονιάς τότε διάλεξε αυθαίρετα τα πρώτα b που θα βρεθούν.

Βήμα 5 Τοποθέτησε στη θέση $k \, mod \, C$ της λίστας TL_C^ϵ το στοιχείο $C_{max}(\Pi_l^\epsilon)$.

Βήμα 6 Έστω $K_{ij}(\Pi_k) = \Pi_l^\epsilon$. Τοποθέτησε στη θέση $k \, mod \, P$ της λίστας TL_P^ϵ το στοιχείο i.

Βήμα 7 Στείλε σε κάθε εργάτη-παιδί

- Το μήνυμα εργασίας ΕΝΑΡΞΗ_ΕΡΓΑΣΙΑΣ.
- Το πρόγραμμα Π_l^ϵ.
- Τη λίστα TL_C^ϵ.
- Τη λίστα TL_P^ϵ.

Βήμα 8 Λάβε από τον πρώτο εργάτη-παιδί το μήνυμα εργασίας.

Βήμα 9 Αν το μήνυμα εργασίας είναι το ΑΔΙΕΞΟΔΟ_ΑΝΑΖΗΤΗΣΗΣ τότε πήγαινε στο βήμα 12.

Αλλιώς λάβε από τον πρώτο εργάτη

- Το πρόγραμμα Π_l.
- Την επανάληψη t_l, όπου εντοπίστηκε το πρόγραμμα Π_l.
- Το πρόγραμμα Π_{k+d}.
- Τη λίστα TL_C.
- Τη λίστα TL_P.

Βήμα 10 Αν $C_{max}(\Pi_l) < C_{max}(\Pi_{\text{βελτ}})$ τότε

- Θέσε $\Pi_{\text{βελτ}} = \Pi_l$
- Θέσε $t = t_l$.

Βήμα 11 Αν $k \geq max\_k$ ή $k - t \geq max\_t$ τότε πήγαινε στο βήμα 12.

Αλλιώς

- Θέσε $k = k + d$.
- Πήγαινε στο βήμα 4.

Βήμα 12 Στείλε σε κάθε εργάτη-παιδί το μήνυμα εργασίας ΛΗΞΗ_ΕΡΓΑΣΙΑΣ και τερμάτισε.

Εργάτης (Worker)

Βήμα 1 Λάβε από τον εργάτη-πατέρα

- Τον αύξοντα κωδικό ϵ .
- Το βάθος d της δενδροειδούς αναζήτησης.
- Το βάθος L της δενδροειδούς αναζήτησης, στο οποίο έχει τοποθετηθεί.

Βήμα 2 Αν $L < d$ τότε δημιούργησε b το πλήθος εργάτες-παιδιά και στείλε σε καθέναν

- Τον αύξοντα κωδικό ϵ^L $(0 \leq \epsilon^L < b)$.
- Το βάθος d της δενδροειδούς αναζήτησης.
- Το βάθος $L + 1$ της δενδροειδούς αναζήτησης, στο οποίο τοποθετούνται.

Βήμα 3 Λάβε από τον εργάτη-πατέρα το μήνυμα εργασίας.

Βήμα 4 Αν το μήνυμα εργασίας είναι το ΛΗΞΗ_ΕΡΓΑΣΙΑΣ τότε πήγαινε στο βήμα 12.

Αλλιώς λάβε από τον εργάτη-πατέρα

- Το πρόγραμμα Π_k^ϵ.
- Τη λίστα TL_C^ϵ.
- Τη λίστα TL_P^ϵ.

Βήμα 5 Αν $U^{S_P}(\Pi_k^\epsilon) - TL_\Pi = \emptyset$ τότε πήγαινε στο βήμα 9.

Αλλιώς :

Βρες b το πλήθος $\Pi_l^{\epsilon^L} \in U^{S_P}(\Pi_k^\epsilon) - TL_\Pi$, τέτοια ώστε

$$C_{max}(\Pi_l^{\epsilon^L}) < C_{max}(\Pi)\,, \quad \forall \Pi \in U^{S_P}(\Pi_k^\epsilon) - TL_\Pi$$

Αν υπάρχουν περισσότερα από b το πλήθος ελάχιστα $\Pi_l^{\epsilon^L}$ γειτονιάς τότε διάλεξε αυθαίρετα τα πρώτα b που θα βρεθούν.

Βήμα 6 Τοποθέτησε στη θέση $(k+L)\,mod\,C$ της λίστας $TL_C^{\epsilon^L}$ το στοιχείο $C_{max}(\Pi_l^{\epsilon^L})$.

Βήμα 7 Έστω $K_{ij}(\Pi_k^\epsilon) = \Pi_l^{\epsilon^L}$. Τοποθέτησε στη θέση $(k+L)\,mod\,P$ της λίστας $TL_P^{\epsilon^L}$ το στοιχείο i.

Βήμα 8 Αν $L < d$ τότε :

Βήμα 8.1 Στείλε σε κάθε εργάτη-παιδί :

- Το μήνυμα εργασίας ΕΝΑΡΞΗ_ΕΡΓΑΣΙΑΣ.
- Το πρόγραμμα $\Pi_l^{\epsilon^L}$.
- Τη λίστα $TL_C^{\epsilon^L}$.
- Τη λίστα $TL_P^{\epsilon^L}$.

Βήμα 8.2 Λάβε από τον πρώτο εργάτη-παιδί το μήνυμα εργασίας. Αν το μήνυμα εργασίας είναι το ΕΛΑΧΙΣΤΟ_ΑΝΑΖΗΤΗΣΗΣ τότε λάβε από τον πρώτο εργάτη παιδί :

- Το πρόγραμμα Π_l.
- Την επανάληψη t_l, όπου εντοπίστηκε το πρόγραμμα Π_l.
- Το πρόγραμμα Π_d.
- Τη λίστα TL_C.

- Τη λίστα TL_P.

Βήμα 9 Επικοινώνησε με τους άλλους εργάτες-παιδιά, οι οποίοι έχουν δημιουργηθεί από τον ίδιο εργάτη-πατέρα, για να διαπιστωθεί σε ποια από τις γειτονιές U^{S_P} εντοπίστηκε ελάχιστο ή αν η αναζήτηση έφτασε σε αδιέξοδο.

Βήμα 10 Αν είσαι ο πρώτος ΕΡΓΑΤΗΣ τότε :

Αν ικανοποιείται το πρώτο κριτήριο τερματισμού τότε στείλε στον εργάτη-πατέρα το μήνυμα εργασίας ΑΔΙΕΞΟΔΟ_ΑΝΑΖΗΤΗΣΗΣ.

Αλλιώς, στείλε στον εργάτη-πατέρα (ή συντονιστη αν $L = 1$) :

- Το μήνυμα εργασίας ΕΛΑΧΙΣΤΟ_ΑΝΑΖΗΤΗΣΗΣ.
- Το πρόγραμμα Π_l.
- Την επανάληψη t_l, όπου εντοπίστηκε το πρόγραμμα Π_l.
- Το πρόγραμμα Π_d.
- Τη λίστα TL_C.
- Τη λίστα TL_P.

Βήμα 11 Πήγαινε στο βήμα 3.

Βήμα 12 Αν $L < d$ τότε στείλε σε κάθε εργάτη-παιδί το μήνυμα εργασίας ΛΗΞΗ_ΕΡΓΑΣΙΑΣ.

Βήμα 13 Τερμάτισε.

Κεφάλαιο 7

Δοκιμές Επιδόσεων

7.1 Εισαγωγή

Οι αλγόριθμοι που αναπτύχθηκαν στα πλαίσια της παρούσας εργασίας ανήκουν στην κατηγορία των ευρηματικών μεθόδων για το πρόβλημα της ροϊκής παραγωγής. Καθώς δεν προέκυψαν από την αναλυτική μελέτη ιδιοτήτων των βέλτιστων λύσεων, ο μοναδικός τρόπος για την αξιολόγηση των επιδόσεών τους είναι η οργάνωση συστηματικών δοκιμών πάνω σε μεγάλο αριθμό τυχαίων προβλημάτων. Οι πειραματικές δοκιμές που παρουσιάζονται στο κεφάλαιο αυτό διακρίνονται σε αυτές που αξιολογούν την ποιότητα των παραγόμενων λύσεων και σε εκείνες που εκτιμούν την επιτάχυνση της εκτέλεσης των αλγορίθμων. Αφορούν δε και τα δύο προβλήματα που εξετάστηκαν στην εργασία αυτή : το πρόβλημα $N/\,M/\,F,\,Seq-Ind,\,perm/\,C_{max}$, όπου επιδιώκεται η ελαχιστοποίηση του χρόνου περάτωσης προγράμματος και το πρόβλημα $N/M/F,\,Seq-Ind,perm/T_{max},\,C_{max}$, όπου αναζητούνται εφικτά προγράμματα ως προς τις προθεσμίες των εργασιών ή, όποτε αυτό δεν είναι δυνατό, προγράμματα με μικρή καθυστέρηση των εργασιών που έχουν προθεσμίες.

Στη συνέχεια του κεφαλαίου, θα ορίσουμε καταρχήν τα κριτήρια επιδόσεων των αλγορίθμων, θα αιτιολογήσουμε τις επιλογές που αφορούν τις τιμές των παραμέτρων τους και τα διάφορα μεγέθη προβλημάτων και θα μελετήσουμε την απόδοση των αλγορίθμων χωριστά για κάθε πρόβλημα. Τέλος, θα παρακολουθήσουμε τη μεταβολή του υπολογιστικού χρόνου που απαιτείται για την εκτέλεσή τους σε σχέση με τις τιμές των παραμέτρων τους και την επιτάχυνση που επιτυγχάνεται με την παραλληλοποίησή τους.

7.2 Κριτήρια επιδόσεων

Για την καλύτερη αξιολόγηση των επιδόσεων των αλγορίθμων που αναπτύχθηκαν στα πλαίσια της παρούσας εργασίας, είναι απαραίτητη η ύπαρξη ενός μέτρου σύγκρισης των παραγόμενων λύσεων. Για το σκοπό αυτό θεωρήσαμε σε κάθε περίπτωση ως αλγόριθμο αναφοράς τον αλγόριθμο του Κοπιδάκη [Κοπ93], του οποίου το γενικό αλγοριθμικό σχήμα παρουσιάσαμε στην ενότητα 3.6. Οδηγηθήκαμε στην επιλογή αυτή επειδή στο μονοκριτηριακό πρόβλημα $N/\ M/\ F,\ Seq-Ind,\ perm/\ C_{max}$ ο αλγόριθμος του Κοπιδάκη εμφανίζεται με τις καλύτερες επιδόσεις μεταξύ άλλων [Naw83, WH89, Lei90, Κοπ93], ενώ παράλληλα αποτελεί το μόνο γνωστό αλγόριθμο επίλυσης του δικριτηριακού προβλήματος $N/M/F,\ Seq-Ind,\ perm/T_{max},\ C_{max}$ της σχετικής με τη ροϊκή παραγωγή βιβλιογραφίας.

Στο πρόβλημα $N/\ M/\ F,\ Seq-Ind,\ perm/\ C_{max}$ μοναδικό κριτήριο βελτίστου είναι ο χρόνος περάτωσης C_{max} προγράμματος. Αν συμβολίσουμε με $C_{max}(A)$ το χρόνο περάτωσης προγράμματος που παράγει ως λύση ο αλγόριθμος A και $C(K)$ τον αντίστοιχο χρόνο περάτωσης που παράγει ως λύση ο αλγόριθμος του Κοπιδάκη τότε μπορούμε να ορίσουμε τα παρακάτω κριτήρια επιδόσεων :

- Απόλυτη απόκλιση αλγορίθμου A :

$$error_A\ =\ C_{max}(A)\ -\ C(K)$$

- Σχετική απόκλιση αλγορίθμου A :

$$relerror_A\ =\ \frac{C_{max}(A)\ -\ C(K)}{C(K)}$$

- Σχετική βελτίωση αλγορίθμου A ως προς αλγόριθμο B :

$$imprv_{A,B}\ =\ \frac{error_B - error_A}{error_B}$$

Στο πρόβλημα $N/M/F,\ Seq-Ind,\ perm/T_{max},\ C_{max}$ πραγματοποιούμε δικριτηριακό προγραμματισμό ως προς τη μέγιστη καθυστέρηση T_{max} των εργασιών με προθεσμίες και το χρόνο περάτωσης C_{max} προγράμματος. Θεωρώντας ως πρωτεύον κριτήριο το πρώτο θα αξιολογήσουμε έναν αλγόριθμο A με βάση τις παρακάτω μετρήσεις :

- Αριθμός περιπτώσεων που ο αλγόριθμος A εντόπισε εφικτές λύσεις.

- Αριθμός περιπτώσεων που ο αλγόριθμος A μείωσε τη μέγιστη καθυστέρηση που παρουσίασε η λύση του αλγορίθμου του Κοπιδάκη.

- Η σχετική μείωση της καθυστέρησης των εργασιών :

$$Trelerror_A = \frac{T_{max}(A) - T(K)}{T(K)}$$

όπου $T(K)$ η μέγιστη καθυστέρηση που παράγει ως λύση ο αλγόριθμος του Κοπιδάκη.

- Σχετική απόκλιση του αλγορίθμου A, $relerror_A$.

Τα κριτήρια επιδόσεων που παρουσιάστηκαν πιο πάνω αξιολογούν τους αλγορίθμους ως προς την ποιότητα των λύσεων που παράγουν. Για να εκτιμηθεί η επιτάχυνση που συνεπάγεται η παραλληλοποίηση της αναζήτησης και για να μελετηθεί πώς ο μηχανισμός περιορισμού του μεγέθους των γειτονιών επηρεάζει τον υπολογιστικό χρόνο που απαιτείται για την εκτέλεση των αλγορίθμων, ορίζουμε τα παρακάτω κριτήρια επιδόσεων :

- Επιτάχυνση αλγορίθμου A ως προς αλγόριθμο B

$$speedup_{A,B} = \frac{Y_B}{Y_A}$$

όπου Y_A, Y_B ο υπολογιστικός χρόνος που απαιτεί η εκτέλεση των αλγορίθμων A, B αντίστοιχα.

- Σχετική βελτίωση (υπολογιστικού χρόνου) αλγορίθμου A ως προς αλγόριθμο B :

$$Yimprv_{A,B} = \frac{Y_B - Y_A}{Y_B}$$

7.3 Πειραματικές δοκιμές για το $N/\,M/\,F,\,Seq-Ind,\,perm/\,C_{max}$

Για το προκείμενο πρόβλημα σκοπός των πειραμάτων ήταν η αξιολόγηση της απόδοσης της τοπικής αναζήτησης με απαγόρευση κινήσεων και η εκτίμηση της βελτίωσης που επιτυγχάνεται με την τεχνική της δενδροειδούς αναζήτησης. Για το σκοπό αυτό υλοποιήσαμε και συγκρίναμε τους εξής αλγορίθμους :

- Τον αλγόριθμο τοπικής αναζήτησης με απαγόρευση κινήσεων και αποκλεισμό προγραμμάτων από τις γειτονιές αναζήτησης, ο οποίος παρουσιάστηκε στην ενότητα 4.8 και στο εξής θα αναφέρεται ως TS_C.

- Τον αλγόριθμο δενδροειδούς αναζήτησης, ο οποίος παρουσιάστηκε στην ενότητα 6.5 και στο εξής θα αναφέρεται ως $ParTrTS_C$.

Οι πειραματικές δοκιμές οργανώθηκαν με βασικές επιδιώξεις τη μελέτη της επίδρασης διάφορων μεθόδων παραγωγής αρχικών προγραμμάτων στην ποιότητα των παραγόμενων τελικών λύσεων και τη διερεύνηση της συμπεριφοράς των αλγορίθμων για διάφορα μεγέθη προβλημάτων και διάφορες τιμές των παραμέτρων τους. Σε κάθε περίπτωση οι τελικές λύσεις που παράγονται από τους αλγορίθμους αυτούς, συγκρίνονται με τις αντίστοιχες λύσεις που δίνει ο αλγόριθμος του Κοπιδάκη.

Τα μεταβαλλόμενα μεγέθη των δοκιμών ήταν συνολικά τρία :

N : Ο αριθμός των εργασιών προς προγραμματισμό.

Στις πειραματικές δοκιμές το N πήρε τιμές στο σύνολο $\{6,\ 8,\ 10,\ 20,\ 30,\ 40,\ 50\}$. Προβλήματα με έξι ή οκτώ εργασίες θεωρούνται μικρού μεγέθους, ενώ αυτά με δέκα ή είκοσι θεωρούνται μετρίου. Προβλήματα με περισσότερες εργασίες θεωρούνται μεγάλου μεγέθους. Στη σχετική βιβλιογραφία ο μέγιστος αριθμός εργασιών των προβλημάτων που λύνονται συνήθως δεν υπερβαίνει το σαράντα [Lei90, CB92, Κοπ93].

M : Ο αριθμός των μηχανών.

Ο αριθμός M των μηχανών καθορίζει το πλήθος των σταδίων παραγωγής των συστημάτων ροϊκής παραγωγής και στην πράξη δεν ξεπερνά συνήθως το δέκα. Συστήματα με μεγαλύτερο αριθμό μηχανών θεωρούνται πολύπλοκα. Στις πειραματικές δοκιμές το M πήρε τιμές στο σύνολο $\{4,\ 8,\ 12,\ 16\}$.

T : Το μέγεθος του συνόλου TL_C.

Το μέγεθος T του συνόλου TL_C ορίζει το πλήθος των τοπικών ελαχίστων, προς τα οποία απαγορεύεται να κινηθεί η τοπική αναζήτηση (ενότητα 4.5). Στις πειραματικές δοκιμές το T ορίστηκε να παίρνει τιμές στο σύνολο $\{3, 8, 13, 18, 23, 28\}$. Σύνολα που περιέχουν τρία μόνο στοιχεία θεωρούνται μικρού μεγέθους, ενώ αυτά με οκτώ ή δεκατρία στοιχεία θεωρούνται μετρίου. Στην πράξη το σύνολο TL_C δεν περιέχει περισσότερα από δέκα στοιχεία [BL91, LBG89, WH89].

Η εξάρτηση των επιδόσεων από τα αριθμητικά δεδομένα κάθε προβλήματος είχε ως συνέπεια τη μεγάλη διακύμανση στα αποτελέσματα των αλγορίθμων. Για να εξομαλυνθούν ενδεχόμενες αποκλίσεις και να ενισχυθεί η αξιοπιστία των πειραματικών

μετρήσεων, κρίθηκε απαραίτητη η επίλυση μεγάλου αριθμού διαφορετικών τυχαίων προβλημάτων. Για το λόγο αυτό δημιουργήθηκαν και λύθηκαν 30 τυχαία προβλήματα για κάθε ζεύγος τιμών (N, M). Τα κριτήρια επιδόσεων που αξιολογήθηκαν αφορούσαν πάντα μέσους όρους τιμών για τα 30 διαφορετικά προβλήματα. Οι χρόνοι επεξεργασίας των εργασιών και εξάρμωσης των μηχανών ήταν τυχαίοι ακέραιοι ομοιόμορφα κατανεμημένοι στο διάστημα $[0, 199]$ και $[0, 49]$ αντίστοιχα. Η επιλογή μεγάλου διαστήματος τιμών έγινε με σκοπό τη συχνή εμφάνιση ανομοιόμορφων χρόνων επεξεργασίας, οι οποίοι συνεπάγονται νεκρούς χρόνους στις μηχανές και καθυστέρηση στην εκτέλεση του προγράμματος. Η σχέση 4:1 στο εύρος τιμών των χρόνων επεξεργασίας και εξάρμωσης ορίζει μια γενική αναλογία. Είναι φανερό, όμως, ότι υπήρξαν και περιπτώσεις όπου οι χρόνοι εξάρμωσης ξεπερνούσαν τους αντίστοιχους επεξεργασίας.

Στην ενότητα 7.3.1 συγκρίνονται οι μέθοδοι παραγωγής αρχικών προγραμμάτων που παρουσιάστηκαν στην ενότητα 4.3 και στη συνέχεια διερευνάται η συμπεριφορά του αλγορίθμου TS_C ως προς την παράμετρο T (ενότητα 7.3.2). Η επίδραση του μηχανισμού περιορισμού μεγέθους γειτονιάς στην ποιότητα των παραγόμενων λύσεων εξετάζεται στην ενότητα 7.3.3, ενώ αμέσως μετά μελετάται ο αλγόριθμος $ParTrTS_C$ και εκτιμάται η βελτίωση που επιτυγχάνεται με τη δενδροειδή αναζήτηση (ενότητα 7.3.4). Στην ενότητα 7.3.5 εξετάζεται η περίπτωση ύπαρξης ομοειδών εργασιών και τέλος, στην ενότητα 7.3.6 παρουσιάζονται τα αποτελέσματα της επίλυσης προβλημάτων με διαφορετικές κατανομές στους χρόνους επεξεργασίας και εξάρμωσης.

7.3.1 Σύγκριση ευρηματικών μεθόδων παραγωγής αρχικών προγραμμάτων

Η πρώτη ομάδα δοκιμών οργανώθηκε με κύρια επιδίωξη την αξιολόγηση ευρηματικών μεθόδων παραγωγής αρχικών προγραμμάτων και τη μελέτη της επίδρασής τους στην ποιότητα των τελικών λύσεων που παράγονται από τον αλγόριθμο TS_C. Για το σκοπό αυτό εξετάστηκαν η μέθοδος του Dannenbring (στο εξής θα την αναφέρουμε εν συντομία **DANN**), των νεκρών χρόνων (**IDLE**), της ελάχιστης παρεμβολής (**MINS**), του Palmer (**PALM**) και των Widmer και Hertz (**WIHE**). Οι μέθοδοι αυτές παρουσιάστηκαν αναλυτικά στις ενότητες 3.3 και 4.3.

Καθένα από τα τυχαία προβλήματα λύθηκε συνολικά τριάντα φορές, για καθεμιά από τις πέντε μεθόδους και για τις έξι διαφορετικές τιμές της παράμετρου T , η οποία καθορίζει το πλήθος των απαγορευμένων τοπικών ελαχίστων ($T = 3, 8, 13, 18, 23, 28$). Για την καλύτερη σύγκριση των πέντε μεθόδων δε χρησιμοποιήθηκε ο μηχανισμός

περιορισμού του μεγέθους της γειτονιάς αναζήτησης, έτσι ώστε να εξετάζεται ολόκληρο το σύνολο των προγραμμάτων που περιέχονται σε αυτές. Κατά τη διάρκεια αυτών των πειραματικών δοκιμών κάθε εργασία ανήκε σε διαφορετική ομάδα, έτσι ώστε να μην υπάρχουν ομοειδείς εργασίες και η εξάρμωση των μηχανών να εκτελείται πάντα. Η επιλογή αυτή υπαγορεύθηκε από τη φύση [1] των ευρηματικών μεθόδων που συγκρίνονται.

NxM	DANN	IDLE	MINS	PALM	WIHE
6x8	0,01	0,00	0,07	0,00	0,27
8x8	3,52	2,60	2,12	2,71	3,00
10x8	4,13	3,13	2,58	4,42	3,89
20x8	-2,91	3,56	0,28	-0,75	3,12
30x8	-8,56	-10,57	-4,57	-8,67	5,62
40x8	-43,46	-38,88	-41,89	-42,82	-24,82
50x8	-4,63	-8,67	2,17	-3,69	-2,24
10x4	3,79	0,58	1,11	1,68	-0,06
10x8	4,13	3,13	2,58	4,42	3,89
10x12	2,54	1,32	0,98	-0,21	4,80
10x16	4,22	1,87	5,72	4,71	-0,09

Πίνακας 7.1: Μεταβολή απόλυτης απόκλισης. Μέσος όρος ως προς T. $\mathcal{R} = 100\%$.

Στον πίνακα 7.1 παραθέτουμε τους μέσους όρους απόκλισης για τα τριάντα προβλήματα που λύθηκαν για κάθε μέγεθος και για τις τιμές της παραμέτρου T. Στην πρώτη στήλη παρακολουθούμε το μέγεθος του προβλήματος $(N\text{x}M)$. Αρχικά μεταβάλλεται ο αριθμός των εργασιών για οκτώ μηχανές και στη συνέχεια μεταβάλλεται ο αριθμός των μηχανών για δέκα εργασίες. Έτσι, τα μεγέθη των προβλημάτων που λύθηκαν είναι 6x8, 8x8, 10x8, 20x8, 30x8, 40x8, 50x8, 10x4, 10x12, και 10x16. Για την ευκολότερη παρακολούθηση της διακύμανσης της απόκλισης με τα μεγέθη M και N, τα αποτελέσματα για την κατηγορία προβλημάτων 10x8 επαναλαμβάνονται στην τέταρτη και στην ενδέκατη γραμμή. Στις στήλες δύο έως και έξι δίνεται η απόκλιση του αλγορίθμου TS_C για καθεμία από τις πέντε ευρηματικές μεθόδους που χρησιμοποιήθηκαν για την παραγωγή αρχικών προγραμμάτων. Στην ένατη γραμμή παρατίθεται ο μέσος όρος (Μ.Ο.) της απόκλισης για την περίπτωση που μεταβάλλεται ο αριθμός των εργασιών, ενώ στην τελευταία γραμμή

[1] Από τις πέντε ευρηματικές μεθόδους που χρησιμοποιήθηκαν, οι μέθοδοι του Dannenbring, του Palmer και των Widmer και Hertz, θεωρούν ότι η εξάρμωση των μηχανών εκτελείται πάντοτε (ενότητες 3.3.1-2, 4.3).

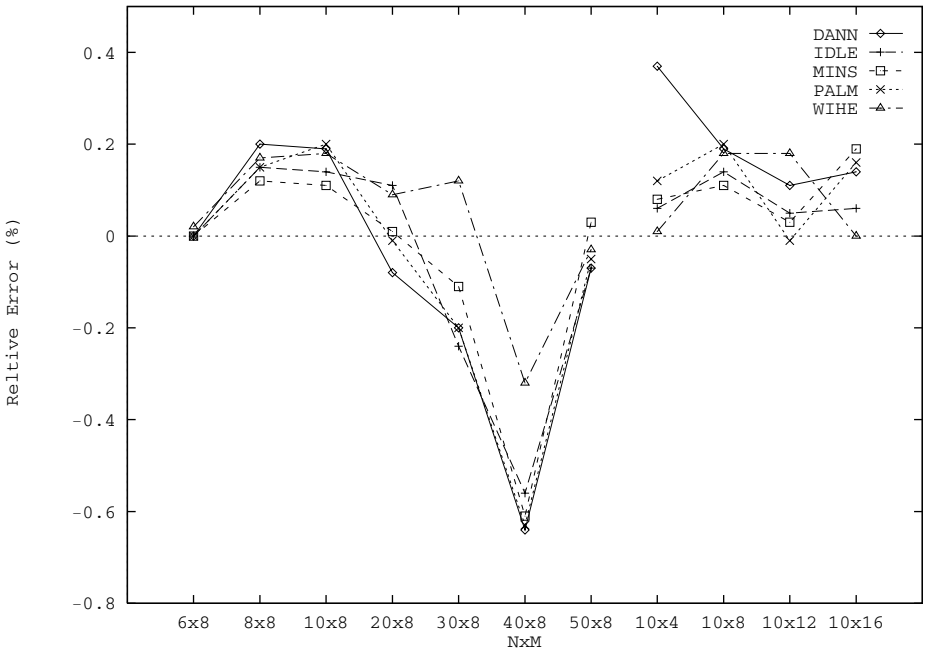
του πίνακα υπολογίζεται ο μέσος όρος στην περίπτωση του μεταβλητού αριθμού μηχανών. Παρόμοια έκθεση της ποσοστιαίας απόκλισης για τις πέντε ευρηματικές μεθόδους γίνεται στον πίνακα 7.2. Ας σημειωθεί ότι σε προβλήματα προγραμματισμού εργασιών η ποσοστιαία απόκλιση θεωρείται πάντα το χαρακτηριστικότερο κριτήριο επίδοσης.

NxM	DANN	IDLE	MINS	PALM	WIHE
6x8	0,00	0,00	0,00	0,00	0,02
8x8	0,20	0,15	0,12	0,15	0,17
10x8	0,19	0,14	0,11	0,20	0,18
20x8	-0,08	0,11	0,01	-0,01	0,09
30x8	-0,20	-0,24	-0,11	-0,20	0,12
40x8	-0,64	-0,56	-0,61	-0,63	-0,32
50x8	-0,07	-0,12	0,03	-0,05	-0,03
10x4	0,37	0,06	0,08	0,12	0,01
10x8	0,19	0,14	0,11	0,20	0,18
10x12	0,11	0,05	0,03	-0,01	0,18
10x16	0,14	0,06	0,19	0,16	0,00

Πίνακας 7.2: Μεταβολή ποσοστιαίας απόκλισης. Μέσος όρος ως προς T. $\mathcal{R} = 100\%$.

Για κάθε πρόβλημα μετρήθηκε η απόκλιση της λύσης του αλγορίθμου TS_C από τη λύση που έδωσε ο αλγόριθμος του Κοπιδάκη. Η απόκλιση είναι θετική όταν οι λύσεις του αλγορίθμου TS_C υστερούν, ενώ είναι αρνητική όταν υπερέχουν, των λύσεων του αλγορίθμου του Κοπιδάκη. Για παράδειγμα, στη δεύτερη στήλη και τρίτη γραμμή του πίνακα 7.2 βλέπουμε την ποσοστιαία απόκλιση για τα προβλήματα προγραμματισμού οκτώ εργασιών σε οκτώ μηχανές, όταν για τα αρχικά προγράμματα έχει χρησιμοποιηθεί η μέθοδος του Dannenbring. Στην περίπτωση αυτή οι χρόνοι περάτωσης που δόθηκαν ως λύση από τον αλγόριθμο TS_C είναι μεγαλύτεροι από τους χρόνους που έδωσε ως λύση ο αλγόριθμος του Κοπιδάκη κατά 0,2%. Στην ίδια στήλη αλλά στην έκτη γραμμή βλέπουμε ότι στα προβλήματα προγραμματισμού τριάντα εργασιών σε οκτώ μηχανές ο αλγόριθμος TS_C , βελτιώνει κατά 0,2% τις τιμές των λύσεων που δίνει ο αλγόριθμος του Κοπιδάκη.

Από τον πίνακα 7.2 παρατηρούμε επίσης ότι η μέθοδος που χρησιμοποιείται για την παραγωγή αρχικών προγραμμάτων δεν επηρεάζει σημαντικά την ποιότητα των λύσεων



Σχήμα 7.1: Μεταβολή ποσοστιαίας απόκλισης. Μέσος όρος ως προς T. $\mathcal{R} = 100\%$.

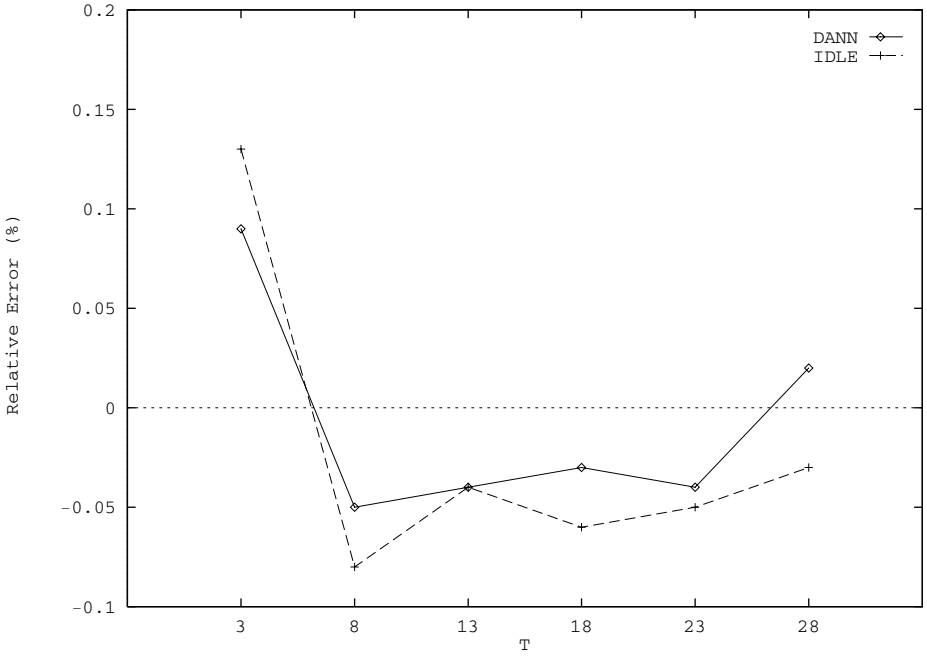
που παράγονται από τον αλγόριθμο TS_C. Η παρατήρηση αυτή, η οποία επιβεβαιώνεται και από τα αποτελέσματα των υπόλοιπων πειραματικών δοκιμών, είναι ιδιαίτερα σημαντική αφού η απόδοση των τεχνικών τοπικής αναζήτησης εξαρτάται συνήθως σημαντικά από την ποιότητα των αρχικών τους λύσεων [Ric83, WH89]. Όπως φαίνεται και στο σχήμα 7.1, όπου παρουσιάζεται η διακύμανση της ποσοστιαίας απόκλισης ως προς το μέγεθος του προβλήματος, στην περίπτωση που ο αριθμός των μηχανών διατηρείται σταθερός η απόκλιση μεταβάλλεται ομοιόμορφα, χωρίς οι τιμές της να εμφανίζουν σημαντικές διαφορές μεταξύ των μεθόδων. Εξαίρεση αποτελεί η μέθοδος των Widmer και Hertz για την οποία ο αλγόριθμος TS_C εμφανίζεται με τη χειρότερη απόδοση, χωρίς όμως στην περίπτωση αυτή η διαφορά της απόκλισης των παραγόμενων λύσεων από των άλλων μεθόδων να είναι μεγαλύτερη από μισή ποσοστιαία μονάδα.

Στην περίπτωση που ο αριθμός των μηχανών μεταβάλλεται η απόκλιση ποικίλει ανάλογα με τη μέθοδο και το πλήθος των σταδίων επεξεργασίας. Παρατηρούμε ότι ο αλγόριθμος TS_C επιτυγχάνει την καλύτερη απόδοση με τη μέθοδο των Widmer και Hertz στα προβλήματα με τέσσερεις και δεκαέξι μηχανές, με τη μέθοδο της ελάχιστης παρεμβολής όταν τα στάδια επεξεργασίας είναι οκτώ και με τη μέθοδο του Palmer όταν είναι δώδεκα. Να σημειωθεί ότι με τις μεθόδους αυτές ο αλγόριθμος TS_C παρουσιάζει σημαντικές διακυμάνσεις στην απόδοσή του, ενώ αντίθετα η μέθοδος των νεκρών χρόνων

παρουσιάζει τη μεγαλύτερη σταθερότητα σε σχέση με τις υπόλοιπες. Οι διαφορές, όμως, που οι μέθοδοι συνεπάγονται στην ποιότητα των παραγόμενων λύσεων είναι και πάλι πολύ μικρές αφού δεν υπερβαίνουν τη μισή ποσοστιαία μονάδα.

7.3.2 Αξιολόγηση μεγέθους T

Στην ενότητα αυτή μελετάται η μεταβολή της απόδοσης του αλγορίθμου TS_C ως προς τις τιμές της παραμέτρου T. Κατά τη διάρκεια των δοκιμών κάθε πρόβλημα λύθηκε για τις έξι διαφορετικές τιμές της παραμέτρου T. Για την παραγωγή των αρχικών προγραμμάτων χρησιμοποιήθηκαν οι πέντε μέθοδοι που εξετάστηκαν στην προηγούμενη ενότητα. Για λόγους απλότητας θα περιοριστούμε στην παρουσίαση των αποτελεσμάτων των δοκιμών στις περιπτώσεις που για τα αρχικά προγράμματα χρησιμοποιήθηκαν οι μέθοδοι του Dannenbring και των νεκρών χρόνων. Οι πίνακες με τις απόλυτες και ποσοστιαίες αποκλίσεις του αλγορίθμου για τα διάφορα μεγέθη προβλημάτων, όλες τις μεθόδους παραγωγής αρχικών προγραμμάτων και τις τιμές της παραμέτρου T παρατίθενται στο παράρτημα Α.



Σχήμα 7.2: Μεταβολή ποσοστιαίας απόκλισης ως προς T.

Από το σχήμα 7.2, όπου παρουσιάζεται η διακύμανση της ποσοστιαίας απόκλισης ως προς τις διάφορες τιμές του T, παρατηρούμε καταρχήν ότι η απόδοση του αλγορίθμου μεταβάλλεται ομοιόμορφα για τις δύο μεθόδους. Η παρατήρηση αυτή επιβεβαιώνει τα

συμπεράσματα της προηγούμενης ενότητας, δηλαδή ότι η συμπεριφορά του αλγορίθμου εμφανίζεται σχετικά ανεξάρτητη της μεθόδου που χρησιμοποιείται για την παραγωγή αρχικών προγραμμάτων. Η καλύτερη απόδοση επιτυγχάνεται για $T = 8$, ενώ τα χειρότερα αποτελέσματα εμφανίζονται για $T = 3$ και $T = 28$ που είναι αντίστοιχα η μικρότερη και η μεγαλύτερη τιμή που λαμβάνει η παράμετρος.

Η αιτιολόγηση των αποτελεσμάτων αυτών πρέπει να αναζητηθεί στην ικανότητα του αλγορίθμου να μην παγιδεύεται σε περιοχές τοπικών ελαχίστων που έχουν εντοπισθεί, απαγορεύοντας την επιστροφή του σε αυτά. Στο σημείο αυτό χρήσιμο είναι να υπενθυμίσουμε ότι η παράμετρος T καθορίζει το μέγιστο πλήθος των τοπικών ελαχίστων προς τα οποία η αναζήτηση δεν επιτρέπεται να κινηθεί [2] (ενότητα 4.5). Όταν λαμβάνει μικρές τιμές, όπως για παράδειγμα στην περίπτωση που $T = 3$, τότε ανάλογα μικρή είναι η διάρκεια που ένα τοπικό ελάχιστο θεωρείται απαγορευμένο, με αποτέλεσμα να είναι εύκολο η αναζήτηση να παγιδευτεί στην περιοχή του. Όταν η παράμετρος T λαμβάνει μεγάλες τιμές, τότε ανάλογα μεγάλη είναι η διάρκεια που ένα τοπικό ελάχιστο και όλα τα σημεία που έχουν τιμή ίση με την τιμή του θεωρούνται απαγορευμένα. Αυτό οδηγεί σε άσχημα αποτελέσματα, καθώς ακόμα και όταν η αναζήτηση έχει απομακρυνθεί από την περιοχή του τοπικού ελαχίστου, του οποίου ο εντοπισμός είχε ως αποτέλεσμα την απαγόρευση όλων των κινήσεων που οδηγούν σε αυτό, όλα τα σημεία του χώρου που έχουν τιμές ίση με την τιμή εκείνου του τοπικού ελαχίστου παραμένουν απρόσιτα, αφού και έχουν απαγορευθεί και οι κινήσεις που οδηγούν σε αυτά.

7.3.3 Αξιολόγηση μεγέθους $\mathcal{R}$

Η απαίτηση για τον περιορισμό του υπολογιστικού χρόνου εκτέλεσης του αλγορίθμου οδήγησε στην ανάπτυξη ενός μηχανισμού περιορισμού του πλήθους των προγραμμάτων που περιέχονται στις γειτονιές αναζήτησης. Η παράμετρος $\mathcal{R}$ καθορίζει το ποσοστό των προγραμμάτων της γειτονιάς που εξετάζονται (ενότητα 4.6). Επειδή η εξέταση μέρους αντί του συνόλου μιας γειτονιάς ενδέχεται να επιδράσει αρνητικά στην ποιότητα των παραγόμενων λύσεων, οργανώθηκαν πειραματικές δοκιμές με βασική επιδίωξη να μελετηθεί πώς επηρεάζει ο μηχανισμός αποκλεισμού προγραμμάτων την ποιότητα των λύσεων, το οποίο αποτελεί αντικείμενο της ενότητας αυτής, και πώς μεταβάλλει τον υπολογιστικό χρόνο, το οποίο εξετάζεται στην ενότητα 7.5.2.

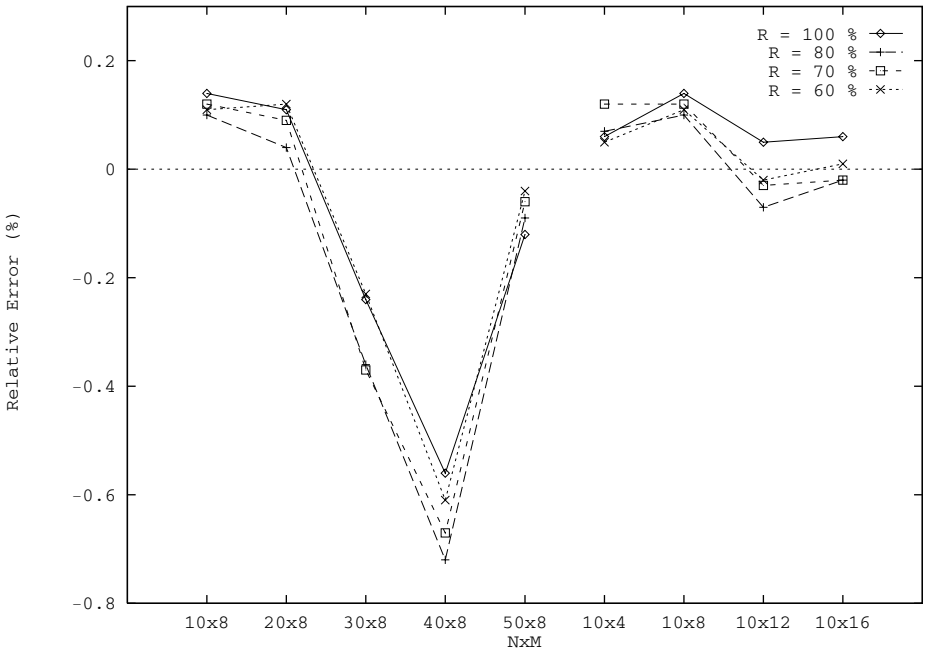
[2] Ο χαρακτηρισμός ενός τοπικού ελαχίστου ως απαγορευμένου δεν περιορίζεται μόνο στο συγκεκριμένο σημείο αλλά επεκτείνεται σε όλα τα σημεία του χώρου αναζήτησης που έχουν τιμή ίση με την τιμή του τοπικού ελαχίστου

Κατά τη διάρκεια των δοκιμών αυτών λύθηκαν προβλήματα μεγέθους 10x8, 20x8, 30x8, 40x8, 50x8, 10x4, 10x12 και 10x16 (τριάντα προβλήματα ανά μέγεθος). Ο υπολογιστικός χρόνος που απαιτείται για τη λύση προβλημάτων μεγέθους 6x8 και 8x8 ήταν πολύ μικρός ώστε οι συγκρίσεις που πραγματοποιούνται να μπορούν να θεωρηθούν αξιόπιστες. Κάθε πρόβλημα λύθηκε τέσσερεις φορές με τον αλγόριθμο TS_C, εξετάζοντας το 100%, 80%, 70% και 60% των προγραμμάτων που περιέχονται στις γειτονιές αναζήτησης.

Για την παραγωγή αρχικών προγραμμάτων χρησιμοποιήθηκαν οι μέθοδοι του Dannenbring και των νεκρών χρόνων. Περιοριστήκαμε σε αυτές τις μεθόδους για δύο λόγους : αφενός στις πειραματικές δοκιμές η απόδοση του αλγορίθμου φάνηκε να είναι ανεξάρτητη από τη μέθοδο που χρησιμοποιείται, αφετέρου μεταξύ των πέντε μεθόδων που συγκρίθηκαν αυτές παρουσίασαν τα καλύτερα αποτελέσματα. Συγκεκριμένα η μέθοδος του Dannenbring εμφανίζεται με την καλύτερη απόδοση όταν ο αριθμός των μηχανών είναι σταθερός, ενώ η μέθοδος των νεκρών χρόνων παρουσιάζει τις μικρότερες διακυμάνσεις όταν μεταβάλλονται τα στάδια επεξεργασίας.

Στην ενότητα αυτή θα παρουσιάσουμε τα αποτελέσματα στην περίπτωση που χρησιμοποιήθηκε η μέθοδος των νεκρών χρόνων για την παραγωγή των αρχικών προγραμμάτων. Οι πίνακες με τις απόλυτες και ποσοτιαίες αποκλίσεις των λύσεων του αλγορίθμου για τα διάφορα μεγέθη προβλημάτων και τις τιμές των παραμέτρων παρατίθενται στο παράρτημα Α.

Στο σχήμα 7.3 παρουσιάζεται η διακύμανση των ποσοστιαίων αποκλίσεων των λύσεων του αλγορίθμου TS_C ως προς το μέγεθος του προβλήματος, για τις τιμές του $\mathcal{R}$. Παρατηρούμε ότι η επίδοση του αλγορίθμου είναι ανώτερη για $\mathcal{R} = 80\%, 70\%$. Στην περίπτωση που κατά την αναζήτηση εξετάζεται το μικρότερο μέρος της γειτονιάς ($\mathcal{R} = 60\%$) η διαφορά στην ποιότητα των παραγόμενων λύσεων από την περίπτωση κατά την οποία η γειτονιά εξετάζεται στο σύνολό της ($\mathcal{R} = 100\%$) είναι ανεπαίσθητη. Τέλος, ας σημειωθεί ότι η βελτίωση που επιτυγχάνεται παρουσιάζεται ανεξάρτητη του μεγέθους του προβλήματος, αφού είτε μεταβάλλεται ο αριθμός των εργασιών είτε μεταβάλλεται ο αριθμός των μηχανών η υπεροχή των λύσεων στην περίπτωση που $\mathcal{R} = 80\%$ είναι σαφής. Από τα παραπάνω αποτελέσματα επιβεβαιώνεται ότι ο μηχανισμός περιορισμού του μεγέθους της γειτονιάς λειτουργεί αποδοτικά αποκλείοντας προγράμματα, τα οποία δεν αποτελούν αξιόλογες κατευθύνσεις για τη συνέχιση της αναζήτησης.

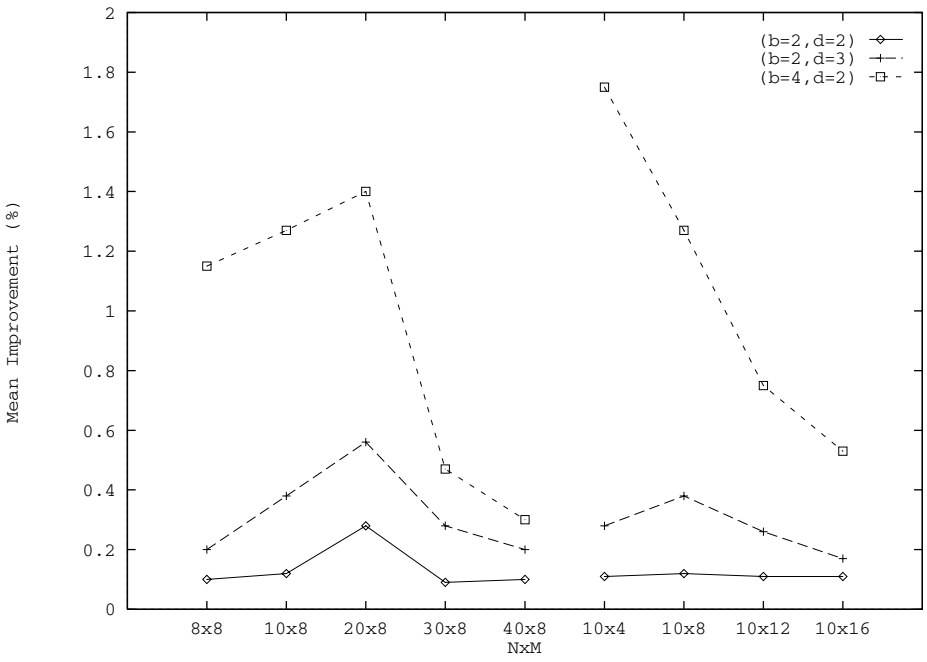


Σχήμα 7.3: Μεταβολή ποσοστιαίας απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο των νεκρών χρόνων.

7.3.4 Απόδοση δενδροειδούς αναζήτησης

Οι πειραματικές δοκιμές της ενότητας αυτής οργανώθηκαν με σκοπό τη μελέτη της απόδοσης του μηχανισμού της δενδροειδούς αναζήτησης. Διερευνήθηκε η συμπεριφορά του αλγορίθμου $ParTrTS_C$ και οι επιδόσεις του συγκρίθηκαν με αυτές του αλγορίθμου TS_C. Τα προβλήματα που λύθηκαν ήταν μεγέθους 8x8, 10x8, 20x8, 30x8, 40x8, 10x4, 10x12 και 10x16 (τριάντα προβλήματα ανά μέγεθος). Οι παράμετροι που καθορίζουν τη λειτουργία του αλγορίθμου $ParTrTS_C$ είναι το βάθος d και το πλάτος b της δενδροειδούς αναζήτησης. Το πλήθος των γειτονιών που εξετάζονται σε κάθε βήμα του $ParTrTS_C$ υπολογίζεται από τη σχέση 6.1 της ενότητας 6.2. Κατά τη διάρκεια των δοκιμών κάθε πρόβλημα λύθηκε για τρία ζεύγη τιμών (b, d) : (2,2), (4,2) και (2,3), με αποτέλεσμα να εξετάζονται αντίστοιχα 3, 5 και 7 γειτονιές προγραμμάτων ανά βήμα.

Στο σχήμα 7.4 παρουσιάζεται η ποσοστιαία βελτίωση του αλγορίθμου $ParTrTS_C$ ως προς τον TS_C, όταν για την παραγωγή των αρχικών προγραμμάτων χρησιμοποιήθηκε η μέθοδος των νεκρών χρόνων. Καταρχήν παρατηρούμε ότι η βελτίωση που επιτυγχάνεται και για τους τρεις συνδυασμούς τιμών των παραμέτρων b και d μεταβάλλεται σχετικά ομοιόμορφα ως προς το μέγεθος του προβλήματος. Η μεγαλύτερη βελτίωση παρατηρείται



Σχήμα 7.4: Ποσοστιαία βελτίωση δενδροειδούς αναζήτησης

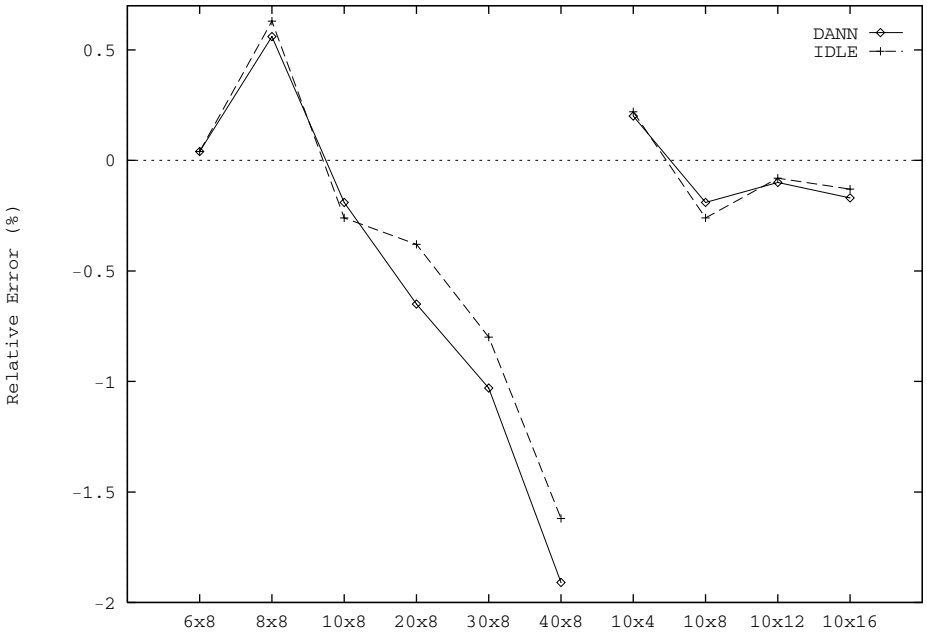
σε προβλήματα όπου είτε ο αριθμός των εργασιών είναι μικρός (8x8, 10x8) είτε ο αριθμός των μηχανών (10x4, 10x8).

Το πλάτος b εμφανίζεται ως η κρισιμότερη παράμετρος της δενδροειδούς αναζήτησης. Να υπενθυμίσουμε ότι το πλάτος καθορίζει τον αριθμό των ισοδύναμων κατευθύνσεων που εξετάζονται και αξιολογούνται. Στην περίπτωση που το πλήθος των κατευθύνσεων αυτών είναι μικρότερο από το πλάτος, τότε εξετάζονται οι αμέσως επόμενες καλύτερες επιλογές. Με τον τρόπο αυτό αίρεται ως ενα βαθμό η μυωπική λειτουργία του αλγορίθμου, με αποτέλεσμα τη βελτίωση της απόδοσής του.

Όταν το πλάτος b της αναζήτησης διατηρείται σταθερό τότε το βάθος d είναι καθοριστικό για την απόδοση του $ParTrTS_C$. Μεγαλύτερο βάθος συνεπάγεται μεγαλύτερο πλήθος γειτονιών που εξετάζονται ανά βήμα (τρεις για το ζεύγος (2,2) και επτά για το (2,3)). Παρατηρούμε ότι για d=2 η απόδοση του $ParTrTS_C$ προσεγγίζει την απόδοση του TS_C και η βελτίωση που επιτυγχάνεται είναι σταθερά μικρότερη του 0,2%, με εξαίρεση το μέγεθος προβλήματος 20x8.

7.3.5 Παρτιδοποίηση εργασιών

Στην ενότητα αυτή μελετάται η συμπεριφορά του αλγορίθμου TS_C όταν υπάρχουν ομάδες εργασιών, οπότε η εξάρμωση των μηχανών δεν εκτελείται μεταξύ ομοειδών εργασιών (ενότητα 2.3). Δημιουργήθηκαν και λύθηκαν προβλήματα μεγέθους 6x8, 8x8, 10x8, 20x8, 30x8, 40x8, 10x4, 10x12 και 10x16 (τριάντα προβλήματα ανά μέγεθος). Σε κάθε πρόβλημα οι εργασίες χωρίσθηκαν με τυχαίο τρόπο σε ομάδες. Κάθε ομάδα περιείχε το πολύ $N/4$ εργασίες. Κατά τη διάρκεια των δοκιμών αυτών στην παράμετρο T δόθηκαν οι τιμές 3, 8, 13, 18, 23 και 28, ενώ εξετάζεται πάντα το σύνολο της γειτονιάς αναζήτησης. Για την παραγωγή των αρχικών προγραμμάτων χρησιμοποιήθηκαν και οι πέντε ευρηματικές μέθοδοι που εξετάστηκαν στην ενότητα 7.3.1. Στο παράρτημα Α παρατίθενται οι πίνακες με τις απόλυτες και ποσοστιαίες αποκλίσεις για τα διάφορα μεγέθη προβλημάτων και τις τιμές των παραμέτρων του αλγορίθμου.



Σχήμα 7.5: Παρτιδοποίηση εργασιών.

Στο σχήμα 7.5 παρουσιάζεται η διακύμανση της ποσοστιαίας απόκλισης ως προς το μέγεθος του προβλήματος όταν υπάρχουν ομάδες εργασιών. Παρατηρούμε ότι η συμπεριφορά του αλγορίθμου δε φαίνεται να διαφοροποιείται σε σχέση με την περίπτωση κατά την οποία κάθε εργασία ανήκε σε διαφορετική ομάδα. Η απόδοσή του παραμένει ανεξάρτητη από τη μέθοδο που χρησιμοποιείται για την παραγωγή αρχικών προγραμμάτων. Συγκρίνοντας τα αποτελέσματα του σχήματος 7.5 με αυτά του σχήματος 7.1 για

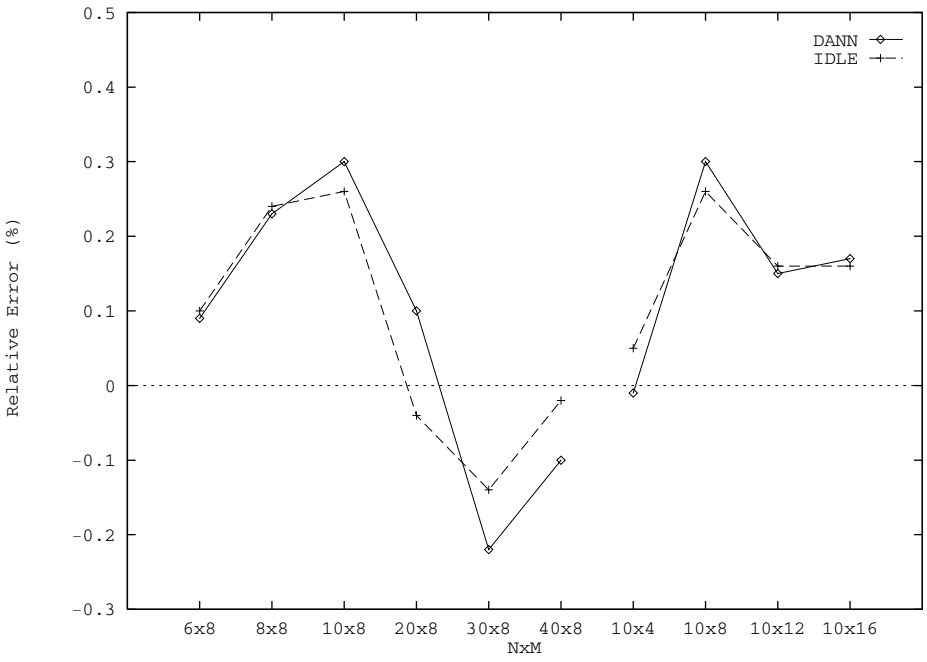
τις πέντε μεθόδους , διαπιστώνουμε ότι εξομαλύνονται οι διαφορές των επιδόσεών τους. Αυτό φαίνεται καλύτερα στην περίπτωση του μεταβλητού αριθμού των μηχανών, όπου οι υπάρχουσες διαφορές είναι ανεπαίσθητες.

Ο αλγόριθμος εξακολουθεί να επιτυγχάνει καλύτερες επιδόσεις στα προβλήματα με μεγάλο αριθμό εργασιών, ενώ η μεταβολή του αριθμού των μηχανών δεν προκαλεί ιδιαίτερες διακυμάνσεις στην ποιότητα των λύσεων. Τη χειρότερη επίδοση την εμφανίζει στο πρόβλημα 8x8, όπου η ποσοστιαία απόκλιση από τις αντίστοιχες λύσεις του αλγορίθμου του Κοπιδάκη είναι υστέρηση περίπου μισής ποσοστιαίας μονάδας, ενώ η καλύτερη επιτυγχάνεται στο πρόβλημα 40x8 και είναι υπεροχή δύο περίπου ποσοστιαίων μονάδων.

7.3.6 Μεταβολή κατανομών

Στην ενότητα αυτή εξετάζεται η συμπεριφορά του αλγορίθμου TS_C όταν οι χρόνοι επεξεργασίας και εξάρμωσης είναι τυχαίοι ακέραιοι ομοιόμορφα κατανεμημένοι στο διάστημα [0,99] και [0,24] αντίστοιχα. Η γενική αναλογία 4:1 διατηρείται, με αλλαγή όμως των ορίων των διαστημάτων. Για κάθε μέγεθος προβλήματος (6x8, 8x8, 10x8, 20x8, 30x8, 40x8, 10x4, 10x12, 10x16) λύθηκαν τριάντα προβλήματα με τον αλγόριθμο TS_C και την παράμετρο T να παίρνει τιμές 3, 8, 13, 18, 23, 28. Για την παραγωγή αρχικών προγραμμάτων χρησιμοποιήθηκαν οι μέθοδοι του Dannenbring και των νεκρών χρόνων. Στο παράρτημα Α παρατίθενται οι πίνακες με τις απόλυτες και ποσοστιαίες αποκλίσεις για τα διάφορα μεγέθη προβλημάτων και τις τιμές των παραμέτρων του αλγορίθμου.

Στο σχήμα 7.6 παρουσιάζεται γραφικά η διακύμανση της ποσοστιαίας απόκλισης με το μέγεθος του προβλήματος. Τα συμπεράσματα των προηγούμενων ενοτήτων επιβεβαιώνονται από τα αποτελέσματα αυτών των πειραματικών δοκιμών. Η μεταβολή της διακύμανσης είναι ομοιόμορφη για τις δύο μεθόδους παραγωγής αρχικών προγραμμάτων. Το πλήθος των εργασιών παραμένει το κρισιμότερο μέγεθος της απόδοσης του αλγορίθμου. Έτσι, στα προβλήματα με μικρό αριθμό εργασιών (6x8, 8x8, 10x8) παρατηρείται υστέρηση με μέγιστη τιμή περίπου 0,3% για το πρόβλημα 10x8, ενώ στα προβλήματα με μεγάλο αριθμό εργασιών έχουμε υπεροχή, η οποία αποκτά μέγιστη τιμή περίπου 0,25% στο πρόβλημα 30x8.



Σχήμα 7.6: Μεταβολή κατανομών.

7.4 Πειραματικές δοκιμές για το πρόβλημα $N/\ M/\ F,\ Seq - Ind,\ perm/\ T_{max},\ C_{max}$

Στο δικριτηριακό πρόβλημα προγραμματισμού εργασιών προέχει η ελαχιστοποίηση της μέγιστης καθυστέρησης T_{max} των εργασιών έναντι της ελαχιστοποίησης του χρόνου περάτωσης C_{max} του προγράμματος. Σκοπός των πειραμάτων ήταν η αξιολόγηση της απόδοσης της τοπικής αναζήτησης με απαγόρευση κινήσεων.

Τα μεταβαλλόμενα μεγέθη των δοκιμών ήταν συνολικά τέσσερα :

N : Ο αριθμός των εργασιών προς προγραμματισμό.

M : Ο αριθμός των μηχανών.

T : Το μέγεθος του συνόλου TL_C.

$ddper$: Το ποσοστό των εργασιών με προθεσμίες επί του συνόλου των εργασιών.

Η παράμετρος $ddper$ καθορίζει την αναλογία των εργασιών με προθεσμίες προς αυτές που δεν έχουν περιορισμούς στο χρόνο περάτωσής τους. Οι τιμές που δόθηκαν στην παράμετρο $ddper$ κατά τις δοκιμές ήταν 70% και 90%. Για καθεμιά ομάδα

δοκιμών που δημιουργήθηκε, λύθηκαν προβλήματα μεγέθους NxM : 6x8, 8x8, 10x8, 20x8, 30x8, 40x8, 10x4, 10x12 και 10x16 (τριάντα προβλήματα ανά μέγεθος). Οι χρόνοι επεξεργασίας και εξάρμωσης ήταν τυχαίοι ακέραιοι ομοιόμορφα κατανεμημένοι στο διάστημα [0,199] και [0,49] αντίστοιχα. Σε κάθε εργασία δόθηκε προθεσμία παράδοσης με πιθανότητα ίση με $ddper$. Η προθεσμία μιας εργασίας i ήταν τυχαίος ακέραιος στο διάστημα $[Topt_i, Cmax(\Pi)]$, όπου $Topt_i = \sum_{j=1}^{M}(p_{ij} + s_{ij})$ εκφράζει το συντομότερο χρόνο περάτωσης της εργασίας i και $C_{max}(\Pi)$ ο χρόνος περάτωσης του προγράμματος Π, στο οποίο καμία εργασία δεν είχε προθεσμία. Κάθε πρόβλημα λύθηκε με τον αλγόριθμο TS_{TC}, ο οποίος παρουσιάστηκε στην ενότητα 4.9. Σε κάθε περίπτωση οι τελικές λύσεις που παράγονται από τον TS_{TC} συγκρίνονται με τις αντίστοιχες λύσεις που δίνει ο αλγόριθμος του Κοπιδάκη.

Στην ενότητα αυτή θα ασχοληθούμε με την ομάδα δοκιμών $ddper$ =70%. Οι πίνακες με τα αποτελέσματα για $ddper$ =90% παρατίθενται στο παράρτημα Β. Στην πρώτη στήλη του πίνακα 7.3 παρακολουθούμε το μέγεθος του προβλήματος. Στις επόμενες τρεις στήλες παρουσιάζονται διαδοχικά ο αριθμός ΕΛ των προβλημάτων για τα οποία εντοπίσθηκε εφικτή λύση, ο αριθμός ΒΛ των προβλημάτων στα οποία ο αλγόριθμος TS_{TC} εντόπισε μη εφικτη λύση με μικρότερη καθυστέρηση από αυτήν της λύσης του αλγορίθμου του Κοπιδάκη για το ίδιο πρόβλημα και οι ποσοστιαίες αποκλίσεις Trel και Crel ως προς τη μέγιστη καθυστέρηση T_{max} και χρόνο περάτωσης C_{max} αντίστοιχα. Για παράδειγμα στη γραμμή 20x8 παρατηρούμε ότι ο αλγόριθμος TS_{TC} εντόπισε εφικτή λύση σε δεκαοκτώ από τα τριάντα προβλήματα που λύθηκαν. Από τις υπόλοιπες δώδεκα περιπτώσεις σε έντεκα βελτίωσε το T_{max} που έδωσε ως λύση ο αλγόριθμος του Κοπιδάκη. Οι παραγόμενες λύσεις ήταν βελτιωμένες κατά 19,62% ως προς το T_{max} και κατά 1,98% ως προς το C_{max} στο σύνολο των τριάντα προβλημάτων.

Από τα αποτελέσματα που παρουσιάζονται στον πίνακα 7.3 είναι φανερή η υπεροχή του αλγορίθμου TS_{TC}. Κατά μέσο όρο εντοπίζει εφικτή λύση σε δεκαοκτώ και εμφανίζεται με καλύτερη λύση σε έντεκα από τα τριάντα προβλήματα που λύθηκαν. Οι παραγόμενες λύσεις υπερέχουν κατά 13,16% ως προς το πρωτεύον και κατά 2,32% ως προς το δευτερεύον κριτήριο του προγραμματισμού. Η υπεροχή αυτή είναι σημαντική, ειδικά αν συγκριθεί με τις επιδόσεις του αλγορίθμου της τοπικής αναζήτησης για το μονοκριτηριακό πρόβλημα όπου η απόκλιση δεν ξεπέρασε τη μισή ποσοστιαία μονάδα. Επιβεβαιώνεται, λοιπόν, η υπόθεση ότι η αναζήτηση στο δικριτηριακό χώρο είναι αποδοτικότερη από ότι στο μονοκριτηριακό.

NxM	ΕΛ	ΒΛ	Trel	Crel
6x8	7	23	-26,68	-4,53
8x8	20	9	-6,96	-1,00
10x8	19	11	-12,98	-1,00
20x8	18	11	-19,62	-1,98
30x8	23	7	-7,23	-2,35
40x8	22	5	-5,49	-3,08
Μ.Ο.	18,16	11	-13,16	-2,32
10x4	23	6	-5,60	-0,40
10x8	19	11	-12,98	-1,00
10x12	17	13	-17,08	-1,97
10x16	14	16	-21,65	-1,76
Μ.Ο.	18,25	11,5	-14,32	-1,28

Πίνακας 7.3: Αποτελέσματα για $ddper$ = 70%

7.4.1 Αξιολόγηση μεγέθους T

Στην ενότητα αυτή μελετάται η μεταβολή της απόδοσης του αλγορίθμου ως προς τις τιμές της παραμέτρου T. Καθένα πρόβλημα λύθηκε για τις έξι τιμές της παραμέτρου T. Για λόγους απλότητας θα περιοριστούμε στην παρουσιάση των αποτελεσμάτων των δοκιμών στην περίπτωση που $ddper$ = 70%. Οι πίνακες με τις ποσοστιαίες αποκλίσεις του αλγορίθμου για $ddper$ = 90% παρατίθενται στο παράρτημα Β.

T	ΕΛ	ΒΛ	Trel	Crel
3	18,11	11,22	-13,64	-1,92
8	18,11	11,22	-13,72	-2,02
13	18,11	11,22	-13,73	-2,05
18	18,11	11,22	-13,72	-2,01
23	18,11	11,22	-13,67	-2,02
28	18,11	11,22	-13,68	-2,00
Μ.Ο.	18,11	11,22	-13,69	-2,00

Πίνακας 7.4: Μεταβολή της απόδοσης ως προς T. $ddper$ = 70%

Στον πίνακα 7.4 παρουσιάζεται η μεταβολή της απόδοσης του αλγορίθμου για τις έξι τιμές της παραμέτρου T. Παρατηρούμε ότι η μεταβολή του μεγέθους T δεν επηρεάζει το πλήθος των εφικτών λύσεων που εντοπίζονται ούτε τον αριθμό των περιπτώσεων, όπου βελτιώνεται η μέγιστη καθυστέρηση των εργασιών με προθεσμίες. Η καλύτερη επίδοση έπιτυγχάνεται για T = 13(η αντίστοιχη για το μονοκριτηριακό πρόβλημα ήταν T = 8). Η διαφορές που εμφανίζονται στις τιμές των Trel και Crel είναι μικρές και δεν υπερβαίνουν το 0,3% και 0,1% αντίστοιχα.

7.4.2 Αξιολόγηση μεγέθους $\mathcal{R}$

Στην ενότητα αυτή εξετάζεται η επίδραση του μεγέθους $\mathcal{R}$ στην ποιότητα των λύσεων που παράγονται από τον αλγόριθμο TS_{TC}. Η μεταβολή που συνεπάγεται στον υπολογιστικό χρόνο μελετάται στην ενότητα 7.5.2. Κατά τη διάρκεια των δοκιμών λύθηκαν προβλήματα μεγέθους 10x8, 20x8, 30x8, 40x8, 10x4, 10x12 και 10x16. Επειδή ο υπολογιστικός χρόνος που απαιτείται για τη λύση προβλημάτων μικρότερου μεγέθους (6x8, 8x8)

είναι πολύ μικρός, ο μηχανισμός περιορισμού μεγέθους γειτονιάς που ελέγχεται από την παράμετρο $\mathcal{R}$ δε χρησιμοποιήθηκε για τη λύση τους.

Κάθε πρόβλημα λύθηκε τέσσερις φορές με τον αλγόριθμο TS_{TC} εξετάζοντας κάθε φορά το 100%, 80%, 70% και 60% των προγραμμάτων που περιέχονται στις γειτονιές αναζήτησης. Στην ενότητα αυτή θα παρουσιάσουμε τα αποτελέσματα για την περίπτωση που $ddper$ = 70% και για $\mathcal{R}$ = 60%. Οι πίνακες με τις ποσοστιαίες αποκλίσεις των λύσεων για τις άλλες τιμές των παραμέτρων του αλγορίθμου παρατίθενται στο παράρτημα Β.

NxM	$\mathcal{R}$ = 60 %			
	ΕΛ	ΒΛ	Trel	Crel
10x8	19	11	-12,99	-1,07
20x8	18	11	-19,62	-2,06
30x8	23	7	-7,23	-2,34
40x8	22	5	-5,49	-2,94
Μ.Ο.	20,5	8,5	-11,33	-2,10
10x4	23	6	-5,60	-0,27
10x8	19	11	-12,99	-1,07
10x12	17	13	-17,27	-2,11
10x16	14	16	-21,67	-1,76
Μ.Ο.	18,25	11,5	-14,38	-1,30

Πίνακας 7.5: Αποτελέσματα για $ddper$ = 70%, $\mathcal{R}$ = 60%

Συγκρίνοντας τα αποτελέσματα του πίνακα 7.5 με αυτά που παρουσιάζονται στον πίνακα 7.3 βλέπουμε ότι ο περιορισμός στην εξέταση του 60% των προγραμμάτων που περιέχονται στη γειτονιά αναζήτησης δεν επηρεάζει σημαντικά τις επιδόσεις του αλγορίθμου. Καταρχήν παρατηρούμε ότι ο αριθμός των περιπτώσεων που εντοπίζεται εφικτή λύση δε μεταβάλλεται για κανένα μέγεθος προβλήματος. Όσον αφορά την απόκλιση Trel δεν παρατηρείται καμία διαφορά στις τιμές της, ενώ μικρή είναι η διακύμανση της Crel, η οποία στο πρόβλημα 20x8 βελτιώνεται 0,08 ποσοστιαίες μονάδες και στο πρόβλημα 40x8 ελαττώνεται 0,14. Βλέπουμε, λοιπόν, ότι ο μηχανισμός περιορισμού του μεγέθους γειτονιάς επιδρά με όμοιο τρόπο στην απόδοση των αλγορίθμων τοπικής αναζήτησης τόσο στο μονοκριτηριακό όσο και στο δικριτηριακό πρόβλημα, προκαλώντας

ανεπαίσθητες μεταβολές στην επίδοση των αλγορίθμων. Αυτό ισχυροποιεί την υπόθεσή μας ότι τα προγράμματα που αποκλείονται από τις γειτονιές αναζήτησης δεν αποτελούν αξιόλογες κατευθύνσεις.

7.4.3 Παρτιδοποίηση εργασιών

Στην ενότητα αυτή μελετάται η συμπεριφορά του αλγορίθμου TS_{TC} όταν υπάρχουν ομάδες εργασιών, οπότε η εξάρμωση των μηχανών δεν εκτελείται μεταξύ ομοειδών εργασιών. Σε κάθε μέγεθος προβλήματος που λύθηκε οι εργασίες χωρίσθηκαν με τυχαίο τρόπο σε ομάδες. Σε κάθε ομάδα ανήκαν το πολύ $N/4$ εργασίες. Ενδεικτικά θα παρουσιάσουμε τα αποτελέσματα για την περίπτωση που $ddper$ = 70%. Στο παράρτημα Β παρατίθενται οι πίνακες με τις αποκλίσεις για $ddper = 90\%$.

NxM	ΕΛ	ΒΛ	Trel	Crel
6x8	12	17	-29,32	-5,09
8x8	21	9	-7,53	-2,17
10x8	17	10	-13,64	-1,86
20x8	21	9	-22,62	-2,32
30x8	23	7	-9,18	-2,34
40x8	18	10	-6,17	-3,74
Μ.Ο.	18,66	10,3	-14,74	-2,92
10x4	20	8	-6,75	-0,12
10x8	17	10	-13,64	-1,86
10x12	21	9	-20,19	-1,27
10x16	18	12	-22,14	-1,63
Μ.Ο.	19	9,75	-15,68	-1,22

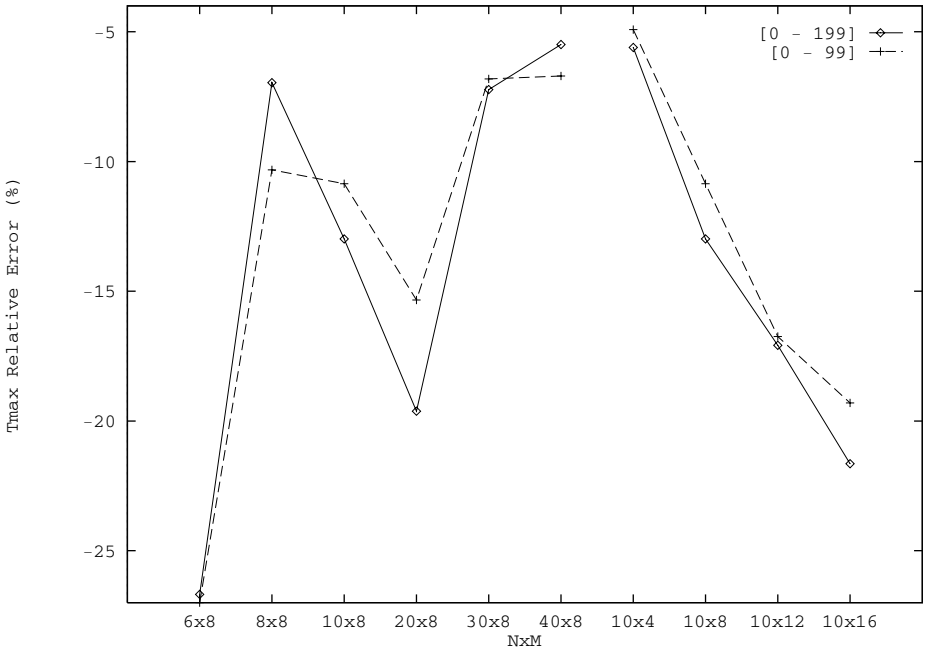
Πίνακας 7.6: Παρτιδοποίηση εργασιών για $ddper = 70\%$.

Από τον πίνακα 7.6 παρατηρούμε ότι η ύπαρξη ομοειδών εργασιών δεν επηρεάζει τη συμπεριφορά του αλγορίθμου TS_{TC}. Κατά μέσο όρο εντοπίσθηκε εφικτή λύση σε 18,66 προβλήματα ενώ η μέγιστη καθυστέρηση των εργασιών βελτιώθηκε σε 10,3. Οι αντίστοιχες τιμές στην περίπτωση που κάθε εργασία ανήκε σε διαφορετική ομάδα ήταν

18,16 στην πρώτη και 11 στη δεύτερη περίπτωση. Παρατηρείται δηλαδή μια μικρή αύξηση των περιπτώσεων που αλγόριθμος εντόπισε εφικτές λύσεις. Ανάλογα μικρή είναι και η μεταβολή του Crel από -2,32% σε -2,92%, ενώ το Trel βελτιώνεται περίπου κατα μία ποσοστιαία μονάδα. Βλέπουμε, λοιπόν, ότι η συμπεριφορά του αλγορίθμου TS_{TC} δεν επηρεάζεται σημαντικά από την ύπαρξη ομοειδών εργασιών.

7.4.4 Μεταβολή κατανομών

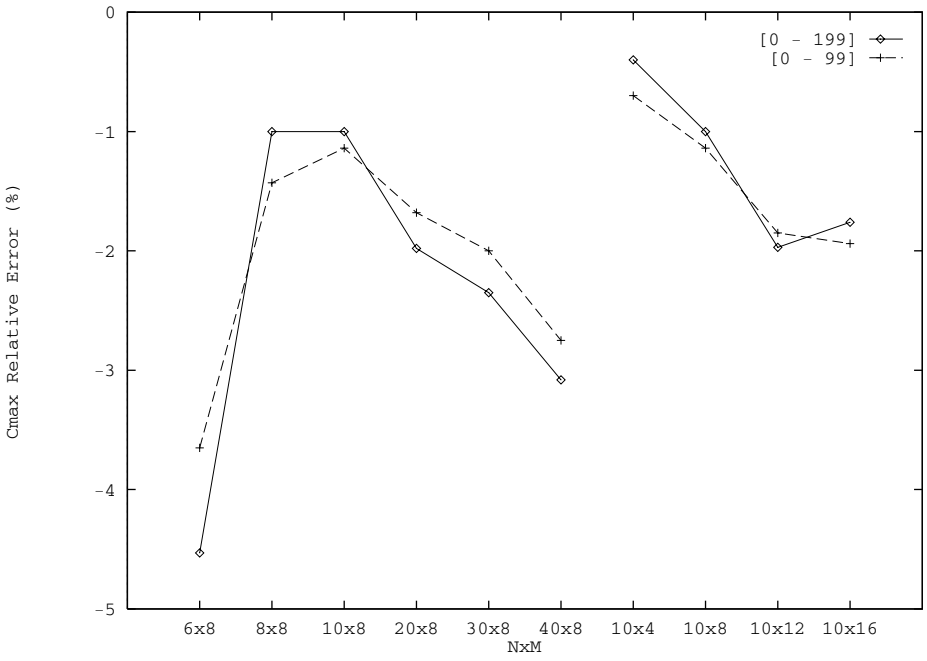
Στην ενότητα αυτή εξετάζεται η συμπεριφορά του αλγορίθμου TS_{TC} όταν οι χρόνοι επεξεργασίας και εξάρμωσης είναι τυχαίοι ακέραιοι ομοιόμορφα κατανεμημένοι στο διάστημα [0,99] και [0,49] αντίστοιχα. Η γενική αναλογία 4:1 διατηρείται, με αλλαγή όμως των ορίων των διαστημάτων. Για κάθε μέγεθος προβλήματος λύθηκαν τριάντα προβλήματα με τον αλγόριθμο TS_{TC} για $ddper$ = 70% και 90%. Στα σχήματα 7.7 και 7.8 παρουσιάζονται γραφικά οι διακυμάνσεις των ποσοστιαίων αποκλίσεων Trel και Crel αντίστοιχα, με το μέγεθος του προβλήματος και για $ddper$ = 70% . Στο παράρτημα Β παρατίθενται αναλυτικά οι πίνακες με τις ποσοστιαίες αποκλίσεις για τα διάφορα μεγέθη προβλημάτων και τις τιμές των παραμέτρων.



Σχήμα 7.7: Μεταβολή κατανομών για Trel και $ddper$ = 70%

Τα συμπεράσματα των προηγουμένων ενοτήτων επιβεβαιώνονται από τα αποτελέσματα αυτών των πειραματικών δοκιμών. Οι λύσεις που παράγονται από τον αλγόριθμο

TS_{TC} εξακολουθούν να υπερέχουν και ως προς τα δύο κριτήρια. Η μεγαλύτερη βελτίωση για το T_{max} επιτυγχάνεται πάλι για το μέγεθος προβλήματος 6x8 και είναι περίπου 27 ποσοστιαίες μονάδες, ενώ η μικρότερη παρατηρείται στο 10x4 και πέντε ποσοστιαίες μονάδες. Η μεταβολή της απόκλισης Trel παραμένει σχετικά ανεξάρτητη του αριθμού των εργασιών, ενώ αντίθετα διατηρείται η εξάρτησή της από το πλήθος των μηχανών. Τα ίδια συμπεράσματα βλέπουμε ότι ισχύουν και για τη διακύμανση του Crel, όπως φαίνεται στο σχήμα 7.8.



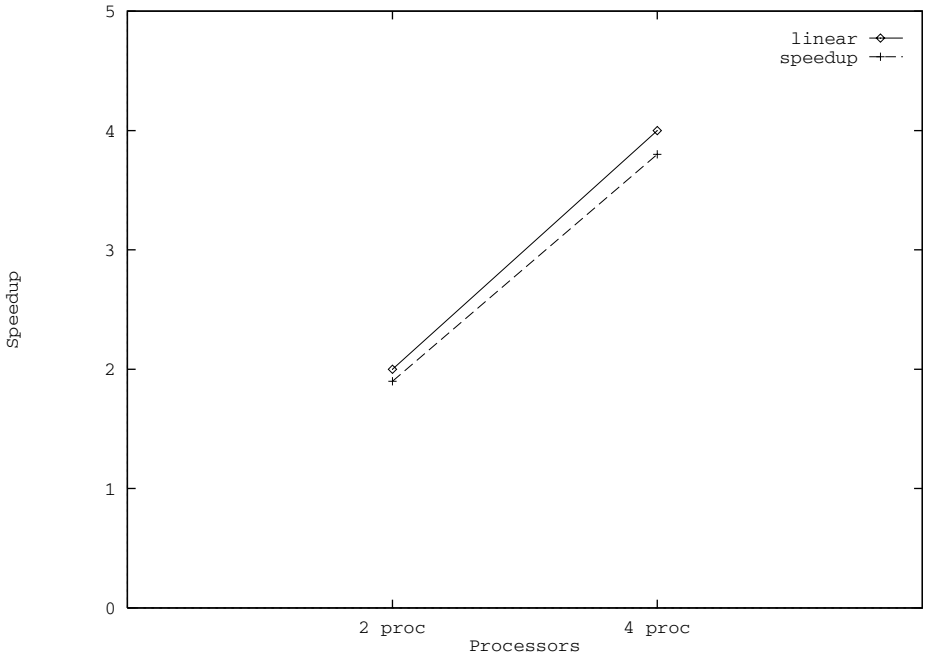
Σχήμα 7.8: Μεταβολή κατανομών για Crel και $ddper$ = 70%

7.5 Υπολογιστικός χρόνος

Οι αλγόριθμοι τοπικής αναζήτησης με απαγόρευση κινήσεων απαιτούν συνήθως σημαντικό υπολογιστικό χρόνο για την εκτέλεσή τους. Η επιτάχυνση της εκτέλεσής τους αποτέλεσε έναν από τους στόχους της παρούσας εργασίας. Στην ενότητα αυτή αξιολογούνται οι δύο μέθοδοι που χρησιμοποιήθηκαν για το σκοπό αυτό : η παραλληλοποίηση της αναζήτησης και ο μηχανισμός περιορισμού του μεγέθους της γειτονιάς.

7.5.1 Απόδοση παράλληλης αναζήτησης

Οι πειραματικές δοκιμές που οργανώθηκαν σκοπό είχαν τη διερεύνηση της απόδοσης των παράλληλων αλγορίθμων που αναπτύχθηκαν ως προς το πλήθος E των διαθέσιμων επεξεργαστών. Υλοποιήθηκαν ο αλγόριθμος $ParTs_C$, ο οποίος παρουσιάστηκε στην ενότητα 5.4.1 και αποτελεί την παράλληλη εκδοχή του αλγορίθμου TS_C και ο αλγόριθμος $ParTS_{TC}$, ο οποίος παρουσιάστηκε στην ενότητα 5.4.2 και αποτελεί την παράλληλη εκδοχή του αλγορίθμου TS_{TC}. Περιοριζόμενοι από τις δυνατότητες της παράλληλης μηχανής που χρησιμοποιήθηκε για την εκτέλεση των αλγορίθμων, στην παράμετρο E δόθηκαν οι τιμές 2 και 4. Κατά τη διάρκεια των πειραματικών δοκιμών λύθηκαν προβλήματα μεγέθους 20x8, 30x8 και 40x8 (τριάντα προβλήματα ανά μέγεθος), με την παράμετρο T να παίρνει τις τιμές 3, 13 και 23.

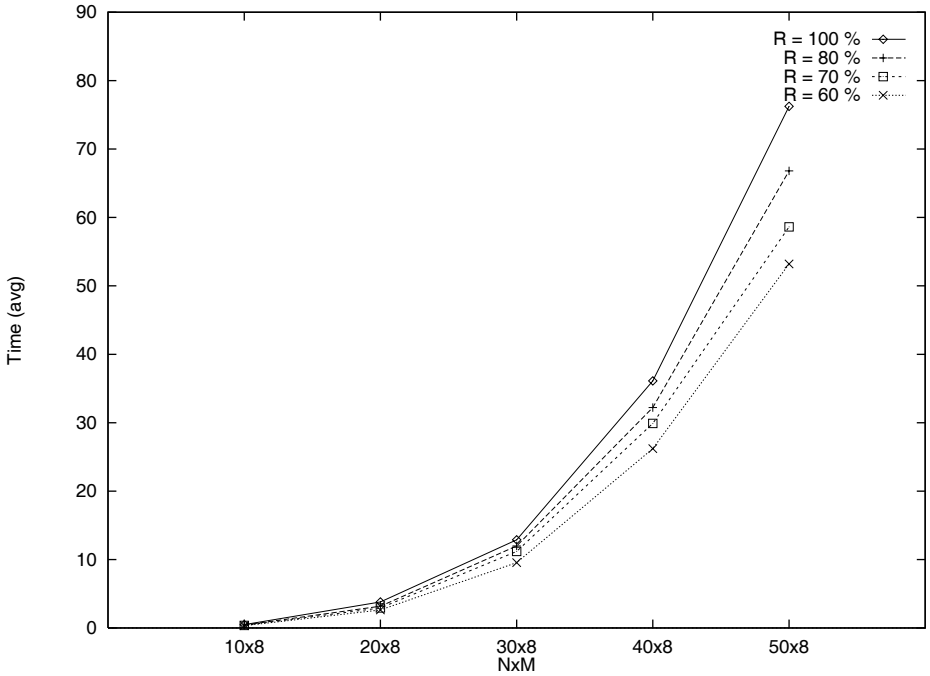


Σχήμα 7.9: Επιτάχυνση που επιτυγχάνεται με την παραλληλοποίηση.

Στο σχήμα 7.9 παρουσιάζεται γραφικά η επιτάχυνση που επιτυγχάνεται ως προς τη γραμμική (linear speedup) που αποτελεί και τη βέλτιστη. Παρατηρούμε ότι οι παράλληλοι αλγόριθμοι είναι ιδιαίτερα αποδοτικοί καθώς η επιτάχυνσή τους είναι πολύ κοντά στη βέλτιστη. Συγκεκριμένα, για $E = 2$ έχουμε $speedup = 1,9$ και για $E = 4$ έχουμε $speedup = 3,7$. Οι πίνακες με τους υπολογιστικούς χρόνους παρατίθενται στο παράάρτημα Γ.

7.5.2 Αξιολόγηση μηχανισμού αποκλεισμού προγραμμάτων από τις γειτονιές αναζήτησης

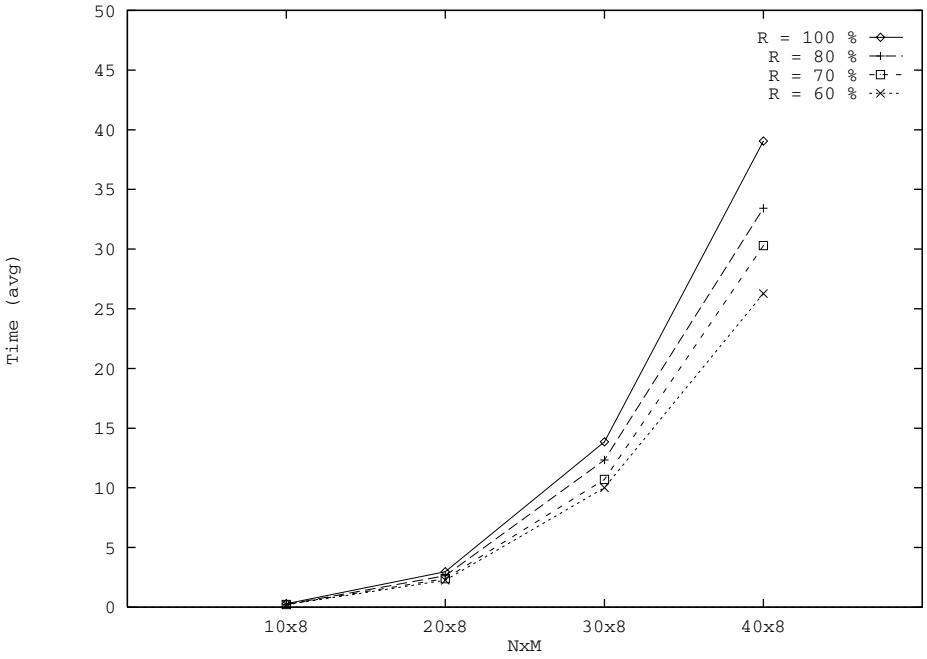
Στην ενότητα αυτή μελετάται πώς ο μηχανισμός περιορισμού μεγέθους μεταβάλλει τον υπολογιστικό χρόνο που απαιτείται για την εκτέλεση των αλγορίθμων TS_C και TS_{TC}. Το μέγεθος $\mathcal{R}$ αποτελεί παράμετρο ελέγχου της λειτουργίας του μηχανισμού, καθορίζει το ποσοστό της γειτονιάς που εξετάζεται. Κατά τη διάρκεια των δοκιμών λύθηκαν προβλήματα μεγέθους 10x8, 20x8, 30x8 και 40x8, ενώ με τον αλγόριθμο TS_C λύθηκαν και προβλήματα μεγέθους 50x8. Στην παράμετρο $\mathcal{R}$ δόθηκαν οι τιμές 100, 80 ,70 και 60, οι οποίες αντιστοιχούν στο ποσοστό της γειτονιάς που εξετάζεται. Οι πίνακες με τους υπολογιστικούς χρόνους για τα διάφορα μεγέθη προβλημάτων και τις τιμές των παραμέτρων παρατίθενται στο παράρτημα Γ.



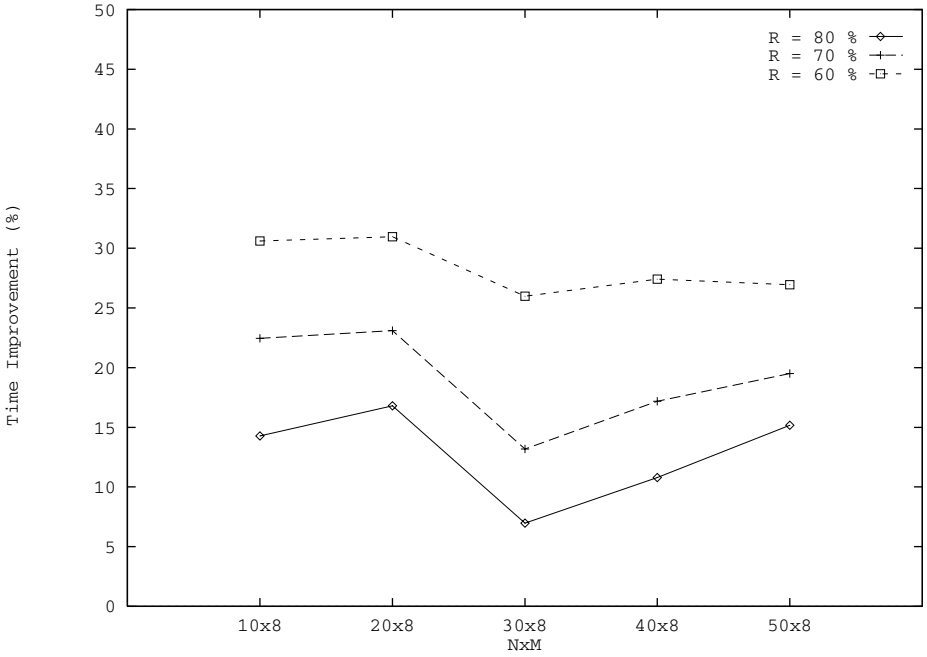
Σχήμα 7.10: Μεταβολή μέσου υπολογιστικού χρόνου για το $N/\ M/\ F,\ Seq - Ind,\ perm/\ C_{max}$.

Στα σχήματα 7.10 και 7.11 παρουσιάζεται γραφικά η μεταβολή του μέσου υπολογιστικού χρόνου (σε CPU seconds) για το μονοκριτηριακό και το δικριτηριακό πρόβλημα αντίστοιχα και στο σχήμα 7.12 η μεταβολή της ποσοστιαίας βελτίωσης. Παρατηρούμε, καταρχήν, ότι ο υπολογιστικός χρόνος μεγαλώνει σημαντικά καθώς αυξάνεται ο αριθμός N των εργασιών. Αυτό ήταν αναμενόμενο, αφού σε κάθε επανάληψη των αλγορίθμων η εξέταση των προγραμμάτων μιας γειτονιάς αναζήτησης απαιτεί χρόνο $\mathcal{O}(N^2)$. Στα

μεγάλου μεγέθους προβλήματα (40x8, 50x8), όπου οι χρόνοι εκτέλεσης των αλγορίθμων είναι σημαντικοί, φαίνεται καθαρά η σταδιακή μείωση των χρόνων με την αύξηση της τιμής της παραμέτρου $\mathcal{R}$.



Σχήμα 7.11: Μεταβολή μέσου υπολογιστικού χρόνου για το $N/\ M/\ F,\ Seq - Ind,\ perm/\ T_{max},\ C_{max}$.



Σχήμα 7.12: Μεταβολή ποσοστιαίας βελτίωσης.

Κεφάλαιο 8

Επίλογος

8.1 Απόδοση αλγορίθμων τοπικής αναζήτησης

Στην παρούσα εργασία μελετήθηκε το πρόβλημα της ροϊκής παραγωγής, το οποίο θεωρείται ένα από τα δύσκολα προβλήματα χρονικού προγραμματισμού. Συγκεκριμένα μας απασχόλησαν δύο μορφές προβλημάτων : το $N/\ M/\ F,\ Seq-Ind,\ perm/\ C_{max}$, όπου επιδιώκεται η ελαχιστοποίηση του χρόνου περάτωσης προγράμματος και το $N/\ M/\ F,\ Seq-Ind,\ perm/\ T_{max},\ C_{max}$, όπου αναζητούνται προγράμματα εφικτά ως προς τις προθεσμίες των εργασιών ή, όποτε αυτό δεν είναι δυνατό, προγράμματα με μικρή καθυστέρηση των εργασιών με προθεσμίες. Η επίλυσή τους με μεθόδους τοπικής αναζήτησης με απαγόρευση κινήσεων οδήγησε σε ουσιαστικές παρατηρήσεις για τη φύση του προβλήματος και των μεθόδων αυτών.

Από την πειραματική αξιολόγηση των αλγορίθμων στο κεφάλαιο 7 διαπιστώθηκε ότι οι επιδόσεις τους εξαρτώνται από το πλήθος των εργασιών που προγραμματίζονται. Τα καλύτερα αποτελέσματα επιτυγχάνονται σε προβλήματα, όπου ο αριθμός αυτός είναι μεγάλος, ενώ οι χειρότερες εμφανίζονται σε προβλήματα όπου ο αριθμός αυτός είναι μικρός. Στο βαθμό που η απόδοση των αλγορίθμων αποτελεί ένα μέτρο της δυσκολίας των προβλημάτων μπορούμε να θεωρήσουμε ως εύκολα τα προβλήματα προγραμματισμού 30, 40 και 50 εργασιών και ως δύσκολα εκείνα με 8 και 10 εργασίες.

Τα προβλήματα προγραμματισμού εργασιών με πολλές μηχανές θεωρούνται δύσκολα, καθώς η εμφάνιση νεκρών χρόνων στις μηχανές προκαλεί σημαντικές καθυστερήσεις στην εκτέλεση των εργασιών. Η δυσκολία τους έγκειται στην αδυναμία υπολογισμού *a priori* αυτών των νεκρών χρόνων. Οι αλγόριθμοι τοπικής αναζήτησης

εξετάζοντας προγράμματα γειτονιών και επιλέγοντας κάθε φορά το καλύτερο ως προς τα κριτήρια βελτίστου που χρησιμοποιούνται, λαμβάνουν υπόψην αυτούς τους νεκρούς χρόνους. Για το λόγο αυτό η απόδοσή τους δεν εξαρτάται σημαντικά από τον αριθμό των μηχανών του προβλήματος.

Η μελέτη της φύσης της μεθόδου τοπικής αναζήτησης μας οδήγησε στο συμπέρασμα ότι δύο είναι οι κρισιμότεροι παράγοντες της απόδοσής της : η ποιότητα των αρχικών προγραμμάτων που παράγονται και το πλήθος των λύσεων που περιέχονται σε μια γειτονιά και είναι ισοδύναμες ως προς τα κριτήρια που χρησιμοποιούνται. Η πρώτη περίπτωση αντιμετωπίσθηκε με το σχεδιασμό ενός μηχανισμού απαγόρευσης κινήσεων, ώστε να αποφεύγεται η παγίδευση της αναζήτησης σε περιοχές τοπικών βελτίστων. Ο μηχανισμός αυτός αποδεικνύεται επιτυχής, με αποτέλεσμα η συμπεριφορά των αλγορίθμων που αναπτύχθηκαν στην παρούσα εργασία να είναι ανεξάρτητη των αρχικών προγραμμάτων. Το φαινόμενο της ύπαρξης ισοδύναμων λύσεων σε μια γειτονιά παρουσιάζεται ιδιαίτερα έντονο στο μονοκριτηριακό πρόβλημα που μελετήθηκε. Για την αντιμετώπισή του επεκτάθηκε η διαδικασία αναζήτησης, ώστε να εξετάζονται ομάδες γειτονιών. Η νέα μέθοδος ονομάστηκε δενδροειδής αναζήτηση και αποδείχθηκε αξιόλογη, ακόμα και για μικρές τιμές των παραμέτρων της.

Οι επιδόσεις των αλγορίθμων συγκρίθηκαν με τις αντίστοιχες του αλγορίθμου του Κοπιδάκη και για τα δύο προβλήματα που μελετήθηκαν Στο μονοκριτηριακό πρόβλημα, στο οποίο ο αλγόριθμος του Κοπιδάκη παρουσιάζει τις καλύτερες επιδόσεις μεταξύ άλλων μεθόδων, οι λύσεις των αλγορίθμων εμφανίζονται περίπου ισάξιες. Η ύπαρξη δεύτερου κριτηρίου, στο δικριτηριακό πρόβλημα, μειώνει το πλήθος των ισοδύναμων λύσεων, με αποτέλεσμα τη θεαματική βελτίωση της απόδοσης του αλγορίθμου και για τα δύο κριτήρια. Στην περίπτωση αυτή η υπεροχή του αλγορίθμου τοπικής αναζήτησης είναι σταθερή και κατά μέσο όρο ίση με 16,7% για το πρωτεύον κριτήριο και 3,5% για το δευτερεύον.

Ο υπολογιστικός χρόνος που απαιτείται για την εκτέλεση των αλγορίθμων τοπικής αναζήτησης αυξάνεται σημαντικά με τον αριθμό των εργασιών του προβλήματος. Στόχος της εργασίας αποτέλεσε ο σχεδιασμός αποδοτικών αλγορίθμων ως προς τον υπολογιστικό χρόνο που απαιτούν. Για το λόγο αυτό σχεδιάστηκε ένας μηχανισμός απόρριψης προγραμμάτων κατά την αναζήτηση, ώστε να περιορισθεί το μέγεθος της γειτονιάς. Η ενσωμάτωση του μηχανισμού αυτού στους αλγορίθμους οδήγησε στη μείωση του χρόνου εκτέλεσής τους κατά 25%. Στο μονοκριτηριακό πρόβλημα που μελετήθηκε, ο μηχανισμός αυτός προκάλεσε μικρή αλλά σαφή βελτίωση στις επιδόσεις

των αλγορίθμων. Η απόρριψη προγραμμάτων από μια γειτονιά έχει ως συνέπεια και τον αποκλεισμό ισοδύναμων λύσεων με αποτέλεσμα την καλύτερη επιλογή κατευθύνσεων. Διαπιστώσαμε, τέλος, ότι η φύση της διαδικασίας αναζήτησης σε γειτονιά προγραμμάτων επιτρέπει τον παραλληλισμό της. Οι παράλληλοι αλγόριθμοι που προτείνονται στην παρούσα εργασία αποδεικνύονται επιτυχείς, αφού η επιτάχυνση που προκαλούν είναι σχεδόν γραμμική.

8.2 Προτάσεις βελτιώσεων

Το μέγεθος του χώρου των εφικτών λύσεων που εξετάζεται κατά την αναζήτηση είναι καθοριστικό για την απόδοση των αλγορίθμων που αναπτύξαμε. Σημαντικό μειονέκτημα των μηχανισμών σχηματισμού γειτονιών, όπως αυτός που παρουσιάστηκε στην ενότητα 4.4, είναι η επικάλυψη των διαδοχικών γειτονιών. Αυτό σημαίνει ότι σε κάθε επανάληψη των αλγορίθμων ένα μέρος μόνο των προγραμμάτων της γειτονιάς εξετάζεται για πρώτη φορά. Σε αρκετές περιπτώσεις διαπιστώθηκε ότι το πλήθος των νέων προγραμμάτων είναι μικρό. Η χρησιμοποίηση γειτονιών που ορίζονται από διαφορετικές κινήσεις ενδέχεται να αντιμετωπίσει αποτελεσματικά το παραπάνω πρόβλημα. Διαφορετικές κινήσεις οδηγούν την αναζήτηση σε διαφορετικές κατευθύνσεις, με αποτέλεσμα να είναι δυσκολότερη η επικάλυψη των γειτονιών που παράγονται.

Τα αλγοριθμικά σχήματα που παρουσιάστηκαν, μπορούν εύκολα να τροποποιηθούν ώστε κατά την αναζήτηση να χρησιμοποιούνται διαφορετικές κινήσεις. Στην περίπτωση αυτή το είδος της κίνησης μπορεί να αλλάζει είτε σε κάθε επανάληψη του αλγορίθμου, ώστε να ελαχιστοποιείται η επικάλυψη των διαδοχικών γειτονιών, είτε κάθε φορά που εντοπίζεται τοπικό ελάχιστο, ώστε η αναζήτηση να κινηθεί σε διαφορετική κατεύθυνση απομακρυνόμενη από την περιοχή του.

Εναλλακτική προσπάθεια αντιμετώπισης του προβλήματος αποτελεί η παράλληλη εκτέλεση των αλγορίθμων με διαφορετικές παραμέτρους, όπως διαφορετικά αρχικά προγράμματα, γειτονιές αναζήτησης και μεγέθη του συνόλου των απαγορευμένων κινήσεων. Οι αλγόριθμοι θα επικοινωνούν ανά τακτά διαστήματα, για να διαπιστώσουν ποιος έχει εντοπίσει την καλύτερη λύση και να συνεχίσουν την αναζήτηση από τη λύση αυτή. Με τον τρόπο αυτό εξασφαλίζεται η εξέταση διαφορετικών περιοχών του χώρου των εφικτών λύσεων.

8.3 Επεκτάσεις

Οι αλγόριθμοι τοπικής αναζήτησης με απαγόρευση κινήσεων που παρουσιάστηκαν στην εργασία αυτή, εφαρμόστηκαν στο πρόβλημα της ροϊκής παραγωγής, το οποίο αποτελεί ένα δύσκολο συνδυαστικό πρόβλημα. Η λειτουργία τους βασίζεται στην εξέταση συνόλων προγραμμάτων επιλέγοντας εκείνα που ικανοποιούν τους περιορισμούς που θέτουν τα κριτήρια βελτίστου. Αυτό το αλγοριθμικό σχήμα επιτρέπει τη χρησιμοποίησή τους για την επίλυση προβλημάτων, στα οποία είναι δυνατός ο ορισμός ενός χώρου εφικτών λύσεων και μιας μεθόδου επιλογής μέσα στο χώρο αυτό.

Ένα ενδιαφέρον πεδίο εφαρμογής των αλγορίθμων αποτελεί ο πολυκριτηριακός προγραμματισμός. Στην εργασία αυτή αντιμετωπίσαμε το δικριτηριακό προγραμματισμό ως προς τη μέγιστη καθυστέρηση των εργασιών και το χρόνο περάτωσης των προγραμμάτων. Παρατηρήσαμε ότι ο δικριτηριακός χώρος είναι πιο δομημένος από το μονοκριτηριακό, με αποτέλεσμα οι αλγόριθμοι να λειτουργούν αποδοτικότερα στην περίπτωση αυτή. Για το λόγο αυτό υποθέτουμε ότι οι μέθοδοι που παρουσιάσαμε, μπορούν να λειτουργήσουν αποδοτικά και για άλλους συνδυασμούς κριτηρίων. Ιδιαίτερο ενδιαφέρον παρουσιάζει η μελέτη της απόδοσης των αλγορίθμων για τρία ή και περισσότερα κριτήρια.

Μια άλλη ενδιαφέρουσα κατηγορία προβλημάτων, στα οποία μπορεί να εφαρμοστεί η μέθοδος της τοπικής αναζήτησης με απαγόρευση κινήσεων, είναι τα προβλήματα χρονικού προγραμματισμού στα οποία οι χρόνοι εξάρμωσης είναι εξαρτημένοι ακολουθίας. Τα προβλήματα αυτά παρουσιάζονται ιδιαίτερα δύσκολα, ακόμα και στην περίπτωση της μοναδικής μηχανής. Όμως, οι αλγόριθμοι που παρουσιάσαμε, καθώς κινούνται μέσα στο χώρο των εφικτών λύσεων επιλέγουν προγράμματα λαμβάνοντας υπόψη τους εξαρτημένους ακολουθίας χρόνους εξάρμωσης και προσπαθώντας να ικανοποιήσουν τα οποιαδήποτε κριτήρια βελτίστου. Για το λόγο αυτό ενδέχεται να λειτουργήσουν αποδοτικά για την επίλυση των προβλημάτων της κατηγορίας αυτής.

Παράρτημα Α

$N/\ M/\ F,\ Seq - Ind,\ perm/\ C_{max}$

NxM	Dannenbring					
	3	8	13	18	23	28
6x8	0,00	0,00	0,00	0,00	0,00	0,00
8x8	0,46	0,10	0,15	0,16	0,10	0,21
10x8	0,46	0,28	0,14	0,09	0,07	0,08
20x8	-0,16	-0,11	-0,03	-0,10	-0,10	0,04
30x8	-0,25	-0,32	-0,35	-0,20	-0,14	-0,25
40x8	-0,37	-0,82	-0,75	-0,78	-0,80	-0,80
50x8	-0,21	-0,17	-0,12	-0,05	0,04	0,11
10x4	0,33	0,34	0,26	0,16	0,17	0,32
10x12	0,31	0,08	0,02	0,04	0,04	0,09
10x16	0,32	0,09	0,15	0,09	0,09	0,09

Πίνακας Α.1: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Dannebring. $\mathcal{R} = 100\%$.

NxM	Νεκρών Χρόνων					
	3	8	13	18	23	28
6x8	0,00	0,00	0,00	0,00	0,00	0,00
8x8	0,38	0,11	0,11	0,05	0,10	0,16
10x8	0,39	0,12	0,07	0,09	0,04	0,10
20x8	0,21	-0,02	0,16	0,11	0,12	0,11
30x8	-0,20	-0,29	-0,29	-0,21	-0,22	-0,20
40x8	-0,28	-0,63	-0,70	-0,65	-0,59	-0,51
50x8	-0,17	-0,22	-0,12	-0,06	-0,08	-0,10
10x4	0,34	-0,02	0,07	0,00	-0,05	0,00
10x12	0,41	0,06	-0,02	-0,02	-0,07	-0,09
10x16	0,28	0,06	0,01	0,01	0,06	-0,06

Πίνακας Α.2: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο νεκρών χρόνων. $\mathcal{R} = 100\%$.

NxM	Ελάχιστης Παρεμβολής					
	3	8	13	18	23	28
6x8	0,02	0,00	0,00	0,00	0,00	0,00
8x8	0,21	0,11	0,10	0,10	0,10	0,10
10x8	0,26	0,14	0,08	0,10	0,03	0,05
20x8	0,12	-0,18	-0,10	0,07	0,11	0,05
30x8	-0,15	-0,07	-0,24	-0,07	-0,05	-0,08
40x8	-0,14	-0,78	-0,71	-0,72	-0,66	-0,65
50x8	0,14	-0,02	-0,19	0,11	0,09	0,06
10x4	0,40	0,04	0,07	0,01	-0,01	-0,01
10x12	0,29	0,00	-0,03	-0,04	0,00	0,00
10x16	0,37	0,13	0,18	0,14	0,13	0,16

Πίνακας Α.3: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο ελάχιστης παρεμβολής. $\mathcal{R} = 100\%$.

NxM	Palmer					
	3	8	13	18	23	28
6x8	0,00	0,00	0,00	0,00	0,00	0,00
8x8	0,23	0,21	0,10	0,13	0,13	0,10
10x8	0,49	0,30	0,16	0,14	0,03	0,08
20x8	-0,02	0,05	-0,04	0,05	-0,07	-0,05
30x8	-0,31	-0,17	-0,16	-0,12	-0,30	-0,13
40x8	-0,26	-0,80	-0,80	-0,59	-0,78	-0,54
50x8	-0,07	-0,19	-0,17	0,04	0,01	0,08
10x4	0,38	0,18	0,09	0,03	0,03	0,04
10x12	0,28	0,01	-0,09	-0,06	-0,09	-0,08
10x16	0,35	0,21	0,09	0,12	0,09	0,09

Πίνακας Α.4: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Palmer. $\mathcal{R} = 100\%$.

NxM	Widmer - Hertz					
	3	8	13	18	23	28
6x8	0,06	0,04	0,00	0,00	0,00	0,00
8x8	0,44	0,25	0,06	0,05	0,19	0,05
10x8	0,41	0,22	0,13	0,07	0,10	0,13
20x8	0,14	0,01	0,05	0,01	0,16	0,20
30x8	0,08	0,06	-0,05	0,11	0,32	0,19
40x8	0,05	-0,44	-0,52	-0,41	-0,17	-0,41
50x8	-0,03	-0,08	0,00	0,04	-0,11	-0,01
10x4	0,11	0,03	-0,05	-0,05	0,01	-0,01
10x12	0,38	0,25	0,09	0,11	0,11	0,12
10x16	0,17	-0,01	-0,02	-0,02	-0,06	-0,05

Πίνακας Α.5: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Widmer - Hertz. $\mathcal{R} = 100\%$.

NxM	Dannenbring					
	3	8	13	18	23	28
10x8	0,46	0,23	0,09	0,10	0,04	-0,01
20x8	-0,30	-0,32	-0,22	-0,03	-0,07	-0,09
30x8	-0,26	-0,24	-0,26	-0,37	-0,45	-0,22
40x8	-0,92	-0,85	-0,87	-0,84	-0,91	-0,84
50x8	-0,22	-0,20	-0,15	-0,08	-0,04	-0,10
10x4	0,18	0,24	0,10	0,14	0,10	0,08
10x12	0,23	-0,05	0,01	0,00	0,03	0,06
10x16	0,07	0,08	0,02	-0,03	-0,03	-0,01

Πίνακας Α.6: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Dannebring. $\mathcal{R} = 80\%$.

NxM	Dannenbring					
	3	8	13	18	23	28
10x8	0,47	0,15	0,03	0,13	0,03	0,05
20x8	-0,14	-0,17	-0,28	-0,12	-0,08	-0,07
30x8	-0,42	-0,21	-0,25	-0,27	-0,23	-0,06
40x8	-0,85	-0,76	-0,82	-0,81	-0,87	-0,74
50x8	-0,28	-0,23	-0,17	-0,10	-0,08	-0,12
10x4	0,22	0,22	0,17	0,11	0,08	0,13
10x12	0,06	0,12	0,10	-0,03	0,06	-0,02
10x16	0,21	0,14	0,05	0,07	0,07	0,07

Πίνακας Α.7: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Dannebring. $\mathcal{R} = 70\%$.

NxM	Dannenbring					
	3	8	13	18	23	28
10x8	0,16	0,19	0,18	0,17	0,12	0,20
20x8	-0,19	-0,31	-0,01	-0,10	0,01	0,10
30x8	-0,17	-0,08	-0,18	-0,20	-0,24	-0,09
40x8	-0,91	-0,72	-0,70	-0,59	-0,64	-0,59
50x8	-0,18	-0,15	-0,10	-0,05	0,00	-0,08
10x4	0,27	0,19	0,16	0,08	0,11	0,12
10x12	0,13	0,04	0,15	0,04	0,09	0,04
10x16	0,06	0,03	0,04	0,12	0,14	0,14

Πίνακας Α.8: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Dannebring. $\mathcal{R} = 60\%$.

NxM	Νεκρών Χρόνων					
	3	8	13	18	23	28
10x8	0.27	0.15	0.06	0.06	0.00	0.06
20x8	-0.18	-0.13	0.04	0.22	0.05	0.25
30x8	-0.28	-0.36	-0.41	-0.45	-0.31	-0.34
40x8	-0.74	-0.77	-0.72	-0.74	-0.68	-0.70
50x8	-0.27	-0.17	-0.05	-0.01	-0.05	0.01
10x4	0.13	-0.04	0.03	0.04	0.12	0.13
10x12	0.05	-0.12	-0.08	-0.08	-0.09	-0.09
10x16	0.12	-0.02	-0.00	-0.06	-0.06	-0.08

Πίνακας Α.9: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο νεκρών χρόνων. $\mathcal{R} = 80\%$.

NxM	Νεκρών Χρόνων					
	3	8	13	18	23	28
10x8	0.20	0.16	0.12	0.05	0.12	0.11
20x8	-0.16	-0.04	0.06	0.20	0.21	0.28
30x8	-0.41	-0.32	-0.34	-0.35	-0.42	-0.36
40x8	-0.76	-0.77	-0.65	-0.69	-0.62	-0.53
50x8	-0.07	-0.08	-0.14	-0.06	-0.04	0.02
10x4	0.24	0.05	0.06	0.07	0.17	0.16
10x12	0.05	-0.01	-0.05	-0.02	-0.08	-0.08
10x16	0.02	0.02	-0.00	-0.06	-0.06	-0.06

Πίνακας Α.10: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο νεκρών χρόνων. $\mathcal{R} = 70\%$.

NxM	Νεκρών Χρόνων					
	3	8	13	18	23	28
10x8	0.20	0.12	0.11	0.03	0.08	0.13
20x8	0.09	-0.08	0.34	0.11	0.12	0.15
30x8	-0.24	-0.26	-0.28	-0.23	-0.19	-0.17
40x8	-0.76	-0.72	-0.60	-0.56	-0.46	-0.52
50x8	-0.23	-0.15	0.01	0.03	0.04	0.04
10x4	0.03	-0.04	-0.04	0.12	-0.00	0.22
10x12	-0.04	-0.10	-0.03	-0.06	0.08	0.03
10x16	0.08	0.06	0.01	-0.04	-0.06	0.01

Πίνακας Α.11: Μεταβολή σχετικής απόκλισης ως προς T. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο νεκρών χρόνων. $\mathcal{R} = 60\%$.

NxM	2 2	2 3	4 2
8x8	0,10	0,20	1,15
10x8	0,12	0,38	1,27
20x8	0,28	0,56	1,40
30x8	0,09	0,28	0,47
40x8	0,10	0,20	0,30
10x4	0,11	0,28	1,75
10x12	0,11	0,26	0,75
10x16	0,11	0,17	0,53

Πίνακας Α.12: Απόδοση δενδροειδούς αναζήτησης. Σχετική βελτίωση

NxM	Dannenbring					
	3	8	13	18	23	28
6x8	0,05	0,04	0,04	0,04	0,04	0,04
8x8	0,65	0,50	0,55	0,55	0,55	0,55
10x8	0,06	-0,17	-0,20	-0,23	-0,29	-0,29
20x8	-0,07	-0,51	-0,69	-0,77	-0,90	-0,93
30x8	-0,56	-0,90	-1,21	-1,25	-1,14	-1,11
40x8	-1,34	-1,84	-2,00	-2,02	-2,08	-2,20
10x4	0,47	0,16	0,35	0,08	0,07	0,04
10x12	0,08	-0,05	-0,16	-0,16	-0,16	-0,13
10x16	0,08	-0,18	-0,20	-0,21	-0,20	-0,22

Πίνακας Α.13: Παρτιδοποίηση εργασιών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Dannebring. $\mathcal{R} = 100\%$.

NxM	Νεκρών Χρόνων					
	3	8	13	18	23	28
6x8	0,04	0,04	0,04	0,04	0,04	0,04
8x8	0,85	0,52	0,57	0,62	0,62	0,62
10x8	-0,14	-0,25	-0,28	-0,30	-0,30	-0,32
20x8	0,31	-0,36	-0,47	-0,61	-0,59	-0,56
30x8	-0,12	-0,85	-0,99	-0,95	-0,78	-1,11
40x8	-0,51	-1,73	-1,85	-1,76	-2,05	-1,84
10x4	0,55	0,17	0,22	0,03	0,17	0,20
10x12	0,21	0,04	-0,15	-0,18	-0,20	-0,20
10x16	0,16	-0,15	-0,15	-0,22	-0,20	-0,20

Πίνακας Α.14: Παρτιδοποίηση εργασιών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο νεκρών χρόνων. $\mathcal{R} = 100\%$.

NxM	Ελάχιστης Παρεμβολής					
	3	8	13	18	23	28
6x8	0,05	0,04	0,04	0,04	0,04	0,04
8x8	0,66	0,68	0,55	0,55	0,55	0,54
10x8	-0,17	-0,29	-0,29	-0,30	-0,30	-0,32
20x8	-0,03	-0,77	-0,81	-0,87	-0,76	-0,75
30x8	-0,34	-0,90	-1,22	-1,17	-1,01	-1,05
40x8	-1,04	-1,90	-1,93	-1,86	-1,92	-1,93
10x4	0,61	0,28	0,34	0,34	0,18	0,13
10x12	0,37	0,03	-0,14	-0,22	-0,31	-0,31
10x16	0,03	-0,19	-0,21	-0,23	-0,17	-0,23

Πίνακας Α.15: Παρτιδοποίηση εργασιών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο ελάχιστης παρεμβολής. $\mathcal{R} = 100\%$.

NxM	Palmer					
	3	8	13	18	23	28
6x8	0,04	0,04	0,04	0,04	0,04	0,04
8x8	0,70	0,54	0,55	0,55	0,55	0,55
10x8	0,29	0,01	-0,07	-0,14	-0,27	-0,17
20x8	0,13	-0,51	-0,72	-0,70	-0,69	-0,76
30x8	-0,64	-0,95	-1,21	-1,20	-1,26	-1,33
40x8	-1,42	-2,00	-1,95	-2,06	-1,91	-1,98
10x4	0,57	0,16	0,37	0,25	0,09	0,09
10x12	0,04	-0,07	-0,19	-0,19	-0,24	-0,24
10x16	0,13	-0,14	-0,26	-0,19	-0,18	-0,18

Πίνακας Α.16: Παρτιδοποίηση εργασιών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Palmer. $\mathcal{R} = 100\%$.

NxM	Widmer - Hertz					
	3	8	13	18	23	28
6x8	0,08	0,10	0,04	0,04	0,04	0,04
8x8	0,83	0,67	0,60	0,58	0,58	0,58
10x8	0,18	-0,13	-0,26	-0,27	-0,29	-0,29
20x8	0,36	-0,48	-0,56	-0,59	-0,54	-0,63
30x8	-0,01	-0,62	-0,66	-1,01	-0,79	-0,87
40x8	-0,32	-1,50	-1,54	-1,51	-1,73	-1,63
10x4	0,65	0,24	0,22	0,22	0,21	0,17
10x12	0,38	-0,19	-0,17	-0,20	-0,21	-0,22
10x16	-0,03	-0,20	-0,23	-0,27	-0,24	-0,24

Πίνακας Α.17: Παρτιδοποίηση εργασιών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Widmer - Hertz. $\mathcal{R} = 100\%$.

NxM	Dannebring					
	3	8	13	18	23	28
6x8	0,11	0,11	0,09	0,09	0,09	0,09
8x8	0,40	0,26	0,23	0,16	0,16	0,16
10x8	0,49	0,29	0,26	0,24	0,26	0,24
20x8	-0,02	0,12	0,08	0,13	0,15	0,15
30x8	-0,31	-0,31	-0,31	-0,14	-0,12	-0,11
40x8	-0,16	-0,18	-0,17	-0,12	-0,01	0,06
10x4	0,07	-0,02	-0,04	-0,02	-0,02	-0,02
10x12	0,25	0,18	0,12	0,10	0,13	0,14
10x16	0,34	0,17	0,17	0,15	0,08	0,12

Πίνακας Α.18: Αλλαγή κατανομών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Dannebring. $\mathcal{R} = 100\%$.

NxM	Νεκρών Χρόνων					
	3	8	13	18	23	28
6x8	0,11	0,11	0,11	0,11	0,09	0,09
8x8	0,40	0,27	0,22	0,20	0,17	0,17
10x8	0,27	0,17	0,31	0,28	0,31	0,23
20x8	0,04	-0,26	-0,15	0,03	0,06	0,04
30x8	-0,17	-0,35	-0,25	-0,08	-0,01	0,03
40x8	-0,08	-0,03	-0,08	0,03	0,03	-0,00
10x4	0,13	0,06	0,04	0,04	0,02	0,03
10x12	0,30	0,13	0,18	0,11	0,13	0,11
10x16	0,31	0,16	0,17	0,07	0,09	0,13

Πίνακας Α.19: Αλλαγή κατανομών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο νεκρών χρόνων. $\mathcal{R} = 100\%$.

NxM	Ελάχιστης Παρεμβολής					
	3	8	13	18	23	28
6x8	0,11	0,11	0,11	0,11	0,09	0,09
8x8	0,36	0,19	0,17	0,16	0,16	0,16
10x8	0,28	0,23	0,14	0,18	0,20	0,21
20x8	0,01	-0,17	-0,13	-0,01	0,01	0,13
30x8	0,11	-0,01	-0,18	0,17	0,34	0,46
40x8	0,08	-0,19	0,04	0,17	0,25	0,14
10x4	0,12	-0,05	-0,02	0,04	0,08	0,12
10x12	0,36	0,31	0,10	0,11	0,11	0,11
10x16	0,33	0,15	0,24	0,19	0,19	0,15

Πίνακας Α.20: Αλλαγή κατανομών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο ελάχιστης παρεμβολής. $\mathcal{R} = 100\%$.

NxM	Palmer					
	3	8	13	18	23	28
6x8	0,11	0,11	0,11	0,11	0,09	0,09
8x8	0,41	0,18	0,19	0,22	0,22	0,21
10x8	0,45	0,31	0,17	0,15	0,07	0,13
20x8	0,20	0,01	0,06	0,35	0,45	0,25
30x8	0,02	-0,05	0,12	0,26	0,27	0,20
40x8	-0,00	-0,08	0,12	0,17	0,23	0,20
10x4	-0,00	0,00	0,03	0,06	0,07	0,08
10x12	0,31	0,23	0,11	0,12	0,14	0,14
10x16	0,36	0,18	0,21	0,21	0,21	0,17

Πίνακας Α.21: Αλλαγή κατανομών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Palmer. $\mathcal{R} = 100\%$.

NxM	Widmer - Hertz					
	3	8	13	18	23	28
6x8	0,20	0,09	0,09	0,09	0,09	0,09
8x8	0,87	0,19	0,16	0,16	0,19	0,19
10x8	0,53	0,18	0,22	0,15	0,13	0,10
20x8	0,50	0,09	0,09	0,10	0,18	0,34
30x8	0,08	0,12	0,09	0,19	0,62	0,55
40x8	0,17	-0,04	0,35	0,09	0,47	0,48
10x4	0,00	-0,05	0,05	0,04	-0,04	-0,04
10x12	0,47	0,37	0,15	0,23	0,31	0,22
10x16	0,48	0,26	0,17	0,16	0,17	0,16

Πίνακας Α.22: Αλλαγή κατανομών. Μεταβολή σχετικής απόκλισης. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Widmer - Hertz. $\mathcal{R} = 100\%$.

Παράρτημα Β

$N/\ M/\ F,\ Seq - Ind,\ perm/\ T_{max},\ C_{max}$

NxM	ΕΛ	ΒΛ	Trel	Crel
6x8	7	23	-26,68	-4,53
8x8	20	9	-6,96	-1,00
10x8	19	11	-12,98	-1,00
20x8	18	11	-19,62	-1,98
30x8	23	7	-7,23	-2,35
40x8	22	5	-5,49	-3,08
Μ.Ο.	18,16	11	-13,16	-2,32
10x4	23	6	-5,60	-0,40
10x8	19	11	-12,98	-1,00
10x12	17	13	-17,08	-1,97
10x16	14	16	-21,65	-1,76
Μ.Ο.	18,25	11,5	-14,32	-1,28

Πίνακας Β.1: Αποτελέσματα για $ddper = 70\%$, $R = 100\%$.

NxM	$\mathcal{R}$ = 80 %			
	ΕΛ	ΒΛ	Trel	Crel
10x8	19	11	-12,94	-1,04
20x8	18	11	-19,62	-2,21
30x8	23	7	-7,23	-2,39
40x8	22	5	-5,49	-3,01
M.O.	20,5	8,5	-11,32	-2,16
10x4	23	6	-5,60	-0,37
10x8	19	11	-12,94	-1,04
10x12	17	13	-17,12	-1,88
10x16	14	16	-21,65	-1,79
M.O.	18,25	11,5	-14,32	-1,27

Πίνακας Β.2: Αποτελέσματα για $ddper = 70\%$, $R = 80\%$.

NxM	$\mathcal{R}$ = 70 %			
	ΕΛ	ΒΛ	Trel	Crel
10x8	19	11	-12,94	-1,08
20x8	18	11	-19,62	-2,04
30x8	23	7	-7,23	-2,29
40x8	22	5	-5,49	-2,95
M.O.	20,5	8,5	-11,32	-2,09
10x4	23	6	-5,60	-0,24
10x8	19	11	-12,94	-1,08
10x12	17	13	-17,26	-1,97
10x16	14	16	-21,67	-1,82
M.O.	18,25	11,5	-14,36	-1,27

Πίνακας Β.3: Αποτελέσματα για $ddper = 70\%$, $R = 70\%$.

NxM	$\mathcal{R}$ = 60 %			
	ΕΛ	ΒΛ	Trel	Crel
10x8	19	11	-12,99	-1,07
20x8	18	11	-19,62	-2,06
30x8	23	7	-7,23	-2,34
40x8	22	5	-5,49	-2,94
Μ.Ο.	20,5	8,5	-11,33	-2,10
10x4	23	6	-5,60	-0,27
10x8	19	11	-12,99	-1,07
10x12	17	13	-17,27	-2,11
10x16	14	16	-21,67	-1,76
Μ.Ο.	18,25	11,5	-14,38	-1,30

Πίνακας Β.4: Αποτελέσματα για $ddper = 70\%$, $R = 60\%$.

T	ΕΛ	ΒΛ	Trel	Crel
3	18,11	11,22	-13,64	-1,92
8	18,11	11,22	-13,72	-2,02
13	18,11	11,22	-13,73	-2,05
18	18,11	11,22	-13,72	-2,01
23	18,11	11,22	-13,67	-2,02
28	18,11	11,22	-13,68	-2,00
Μ.Ο.	18,11	11,22	-13,69	-2,00

Πίνακας Β.5: Αποτελέσματα ως προς T για $ddper = 70\%$.

NxM	ΕΛ	ΒΛ	Trel	Crel
6x8	6	23	-20,14	-8,17
8x8	8	22	-14,09	-5,29
10x8	7	23	-17,42	-5,04
20x8	8	21	-23,81	-6,38
30x8	6	24	-21,19	-4,54
40x8	6	23	-25,06	-3,49
Μ.Ο.	6,83	22,66	-20,28	-5,48
10x4	10	20	-14,32	-4,32
10x8	7	23	-17,42	-5,04
10x12	7	22	-21,17	-3,07
10x16	5	25	-20,12	-4,37
Μ.Ο.	7,25	22,50	-18,25	-4,20

Πίνακας Β.6: Αποτελέσματα για $ddper = 90\%$, $R = 100\%$.

NxM	$\mathcal{R} = 80\ \%$			
	ΕΛ	ΒΛ	Trel	Crel
10x8	7	23	-17,54	-6,17
20x8	8	21	-24,10	-8,39
30x8	6	24	-21,20	-7,53
40x8	6	23	-26,49	-5,10
Μ.Ο.	6,75	22,75	-22,33	-6,79
10x4	10	20	-15,60	-4,43
10x8	7	23	-17,54	-6,17
10x12	7	22	-22,21	-3,66
10x16	5	25	-20,65	-4,23
Μ.Ο.	7,25	22,50	-19,00	-4,62

Πίνακας Β.7: Αποτελέσματα για $ddper = 90\%$, $R = 80\%$.

NxM	$\mathcal{R}$ = 70 %			
	ΕΛ	ΒΛ	Trel	Crel
10x8	7	23	-17,55	-6,19
20x8	8	21	-24,10	-8,78
30x8	6	24	-21,20	-7,41
40x8	6	23	-26,59	-5,84
M.O.	6,75	22,75	-22,36	-7,06
10x4	10	20	-15,60	-4,78
10x8	7	23	-17,55	-6,19
10x12	7	22	-22,21	-4,08
10x16	5	25	-20,65	-4,74
M.O.	7,25	22,50	-19,01	-4,95

Πίνακας Β.8: Αποτελέσματα για $ddper = 90\%$, $R = 70\%$.

NxM	$\mathcal{R}$ = 60 %			
	ΕΛ	ΒΛ	Trel	Crel
10x8	7	23	-17,58	-6,17
20x8	8	21	-24,74	-9,12
30x8	6	24	-22,04	-7,84
40x8	6	23	-26,78	-5,67
M.O.	6,75	22,75	-22,29	-7,20
10x4	10	20	-15,98	-5,14
10x8	7	23	-17,58	-6,17
10x12	7	22	-22,74	-4,06
10x16	5	25	-21,18	-4,76
M.O.	7,25	22,50	-19,37	-5,04

Πίνακας Β.9: Αποτελέσματα για $ddper = 90\%$, $R = 60\%$.

NxM	ΕΛ	ΒΛ	Trel	Crel
6x8	12	17	-29,32	-5,09
8x8	21	9	-7,53	-2,17
10x8	17	10	-13,64	-1,86
20x8	21	9	-22,62	-2,32
30x8	23	7	-9,18	-2,34
40x8	18	10	-6,17	-3,74
Μ.Ο.	18,66	10,3	-14,74	-2,92
10x4	20	8	-6,75	-0,12
10x8	17	10	-13,64	-1,86
10x12	21	9	-20,19	-1,27
10x16	18	12	-22,14	-1,63
Μ.Ο.	19	9,75	-15,68	-1,22

Πίνακας Β.10: Παρτιδοποίηση εργασιών για $ddper = 70\%$.

NxM	ΕΛ	ΒΛ	Trel	Crel
6x8	7	23	-21,27	-9,63
8x8	10	19	-12,84	-7,04
10x8	9	21	-18,36	-5,73
20x8	7	23	-24,12	-5,41
30x8	8	22	-20,20	-6,86
40x8	7	22	-26,37	-3,62
Μ.Ο.	8	21,67	-20,53	-6,38
10x4	10	20	-15,46	-4,03
10x8	9	21	-18,36	-5,73
10x12	9	20	-20,53	-3,32
10x16	6	24	-22,14	-3,94
Μ.Ο.	8,5	21,25	-19,12	-4,26

Πίνακας Β.11: Παρτιδοποίηση εργασιών για $ddper = 90\%$.

NxM	ΕΛ	ΒΛ	Trel	Crel
6x8	7	23	-27,00	-3,65
8x8	20	9	-10,32	-1,43
10x8	19	11	-10,85	-1,14
20x8	18	11	-15,34	-1,68
30x8	23	7	-6,82	-2,00
40x8	22	5	-6,70	-2,75
Μ.Ο.	18,16	11	-12,83	-2,11
10x4	23	6	-4,91	-0,70
10x8	19	11	-10,85	-1,14
10x12	17	13	-16,74	-1,85
10x16	14	16	-19,30	-1,94
Μ.Ο.	18,25	11,5	-12,95	-1,40

Πίνακας Β.12: Αλλαγή κατανομών για $ddper = 70\%$.

NxM	ΕΛ	ΒΛ	Trel	Crel
6x8	6	24	-21,23	-7,18
8x8	9	21	-13,84	-6,83
10x8	8	22	-18,10	-4,57
20x8	7	23	-21,63	-7,34
30x8	7	23	-19,38	-4,03
40x8	6	23	-23,05	-4,32
Μ.Ο.	7,16	22,67	-19,54	-8,57
10x4	10	20	-13,43	-4,11
10x8	8	22	-18,10	-4,57
10x12	8	21	-18,39	-3,51
10x16	7	22	-22,17	-4,00
Μ.Ο.	8,25	21,25	-18,02	-4,05

Πίνακας Β.13: Αλλαγή κατανομών για $ddper = 90\%$.

T	ΕΛ	ΒΛ	Trel	Crel
3	7	22,55	-19,65	-4,79
8	7	22,55	-19,73	-4,95
13	7	22,55	-19,71	-4,92
18	7	22,55	-19,70	-4,89
23	7	22,55	-19,67	-4,80
28	7	22,55	-19,66	-4,82
Μ.Ο.	7	22,55	-19,68	-4,86

Πίνακας Β.14: Αποτελέσματα ως προς T για $ddper = 90\%$.

Παράρτημα Γ

Υπολογιστικός χρόνος

NxM	DANN	IDLE	MINS	PALM	WIHE
6x8	0,10	0,12	0,12	0,12	0,12
8x8	0,24	0,26	0,26	0,26	0,27
10x8	0,50	0,55	0,48	0,53	0,53
20x8	3,42	3,81	3,66	3,60	3,95
30x8	12,58	12,89	11,48	11,28	13,05
40x8	32,68	36,11	34,22	35,46	40,27
50x8	70,04	72,85	76,46	78,06	83,88
10x4	0,30	0,33	0,36	0,35	0,36
10x12	0,59	0,63	0,62	0,65	0,66
10x16	0,71	0,75	0,76	0,73	0,78

Πίνακας Γ.1: Υπολογιστικός χρόνος σε CPU sec.

NxM	Dannenbring			
	100 %	80 %	70 %	60 %
6x8	0,10	-	-	-
8x8	0,24	-	-	-
10x8	0,50	0,42	0,38	0,34
20x8	3,42	3,11	2,79	2,51
30x8	12,58	11,25	9,85	8,75
40x8	32,68	27,66	25,39	21,90
50x8	70,04	65,90	61,72	58,18
10x4	0,30	0,27	0,25	0,21
10x12	0,59	0,53	0,51	0,43
10x16	0,71	0,63	0,54	0,50

Πίνακας Γ.2: Μεταβολή υπολογιστικού χρόνου ως προς R. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο Dannebring.

NxM	Νεκρών Χρόνων			
	100 %	80 %	70 %	60 %
6x8	0,12	-	-	-
8x8	0,26	-	-	-
10x8	0,55	0,42	0,38	0,34
20x8	3,81	3,17	2,93	2,63
30x8	12,89	11,99	11,19	9,54
40x8	36,11	32,21	29,90	26,21
50x8	72,85	66,80	61,63	58,21
10x4	0,33	0,27	0,24	0,22
10x12	0,63	0,51	0,49	0,44
10x16	0,75	0,63	0,56	0,52

Πίνακας Γ.3: Μεταβολή υπολογιστικού χρόνου ως προς R. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο των νεκρών χρόνων.

NxM	EDD			
	100 %	80 %	70 %	60 %
6x8	0,05	-	-	-
8x8	0,12	-	-	-
10x8	0,28	0,23	0,22	0,19
20x8	2,98	2,64	2,38	2,26
30x8	13,85	12,33	10,70	10,02
40x8	39,06	33,42	30,30	26,27
10x4	0,13	0,11	0,09	0,09
10x12	0,44	0,36	0,32	0,29
10x16	0,61	0,50	0,45	0,41

Πίνακας Γ.4: Υπολογιστικός χρόνος σε CPU sec. Παραγωγή αρχικών προγραμμάτων με τη μέθοδο EDD.

NxM	Επεξεργαστές	DANN	IDLE	MINS	PALM	WIHE	EDD
20x8	1	133	141	143	140	153	115
	2	67	72	72	71	78	58
	4	34	38	37	36	38	30
30x8	1	482	478	440	433	483	457
	2	243	241	221	217	243	229
	4	126	125	112	109	122	116
40x8	1	1252	1339	1311	1315	1542	1495
	2	632	676	662	664	778	755
	4	319	352	333	335	392	380

Πίνακας Γ.5: Μεταβολή υπολογιστικού χρόνου με το πλήθος των επεξεργαστών.

Βιβλιογραφία

[Γεω91] Θ. Γεωργίου. Το πρόβλημα της μέγιστης καθυστέρησης σε μία μηχανή με χρόνους εξάρμωσης, 1991.

[AC89] J.R.A. Allwright and D.B. Carpenter. A Distributed Implementation of Simulated Annealing for the Traveling Salesman Problem. *Parallel Computing*, 10:335--338, 1989.

[AK89] E. Aarts and J. Korst. *Simulated Annealing and Boltzman Machines.* Wiley, Chichester, 1989.

[Ash70] S. Ashour. An Experimental Investigation and Comparative Evaluation of Flow-shop Scheduling Technics. *Operations Research*, 18:541--548, 1970.

[Bak74] K.R. Baker. *Introduction to Sequencing and Scheduling.* Wiley, New York, 1974.

[Bak75] K.R. Baker. A Comparative Study of Flow Shop Algorithms. *Operations Research*, 23(1):62--73, January-February 1975.

[BD78] J. Bruno and P. Downey. Complexity of Task Sequencing with Deadlines, Set-Up Times and Changeover Costs. *SIAM Journal of Computing*, 7(4):393--404, November 1978.

[BD93] A. Bertoni and M. Dorigo. Implicit Parallelism in Genetic Algorithms. *Artificial Intelligence*, 61:307--314, 1993.

[BL91] J.W. Barnes and M. Laguna. Solving the Multiple-Machine Weighted Flow Time Problem Using Tabu Search. Technical Report ORP91-04, The University of Texas at Austin, 1991.

[BL93] N. Boissin and J.L. Lutton. A parallel simulated annealing algorithm. *Parallel Computing*, 19(8):859--872, August 1993.

[BR78] F. Burns and J. Rooker. Three-Stage Flow-Shops with Recessive Second Stage. *Operations Research*, 26(1):207--208, January-February 1978.

[CB92] J. Cao and D. Bedworth. Flowshop Scheduling in Serial multi-product processes with Transfer and Setup Times. *International Journal of Production Research*, 20(1):1819--1830, 1992.

[Cer85] V. Cerny. A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41--51, 1985.

[CG90] N. Carriero and D. Gelernter. *How to write parallel programs.* MIT press, Cambridge, Massachusetts, 1990.

[CMM67] R.W. Conway, W.L. Maxwell, and L.W. Miller. *Theory of Scheduling.* Addison-Wesley, Reading, Mass., 1967.

[Dan77] D.G. Dannenbring. An Evaluation of Flow Shop Scheduling Heuristics. *Management Science*, 23(11):1174--1182, July 1977.

[DPS92] R.A. Dudek, S.S. Panwalkar, and M.L. Smith. The lessons of flowshop scheduling research. *Operations Research*, 40(1):7--13, January-February 1992.

[DT64] R.A. Dudek and O.F. Jr. Teuton. Development of M-stage Decision Rule for Scheduling *n* Jobs Through *m* Machines. *Operations Research*, 12(3):471--497, May-June 1964.

[Eva87] J.R. Evans. Structural Analysis of the Local Search Heuristics in Combinatorial Optimization. *Computers and Operations Research*, 14(6):465--477, 1987.

[GJ91] M.R. Garey and D.S. Johnson. *Computers and Intractability A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1991.

[GJS76] M.R. Garey, D.S. Johnson, and R. Sethi. Complexity of Flow Shop and Job Shop Scheduling. *Mathematics of Operations Research*, 1(2):117--129, May 1976.

[GLLK79] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, 5:287--326, 1979.

[Glo89] F. Glover. Tabu Search - Part I. *ORSA Journal on Computing*, 1(3):190--206, 1989.

[Glo90a] F. Glover. Tabu Search - Part II. *ORSA Journal on Computing*, 2(1):4--32, 1990.

[Glo90b] F. Glover. Tabu Search: A Tutorial. *Interfaces*, 20(4):74--94, 1990.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[GR78] J.N.D. Gupta and S.S. Reddi. Improved Dominance Conditions for the Three-Machine Flowshop Scheduling Problem. *Operations Research*, 26(1):200--203, January-February 1978.

[Gra81] S.C. Graves. A Review of Production Scheduling. *Operations Research*, 29:646--675, 1981.

[GS78] T. Gonzalez and S. Sahni. Flowshop and Jobshop Schedules: Complexity and Approximation. *Operations Research*, 26(1):36--52, January-February 1978.

[GSS87] J.N.D. Gupta, J.G. Shanthikumar, and W. Szwarc. Generating Improved Dominance Conditions for the Flowshop Problem. *Computers and Operations Research*, 14(1):41--45, 1987.

[Gup70] J.N.D. Gupta. M-Stage Flow Shop Scheduling by Branch and Bound. *Operations Research*, 7(1):37--43, January-February 1970.

[GW64] R.J. Giglio and H.M. Wagner. Approximate Solutions to the Three-Machine Scheduling Problem. *Operations Research*, 12(2):305--324, March-April 1964.

[HC84] A.C. Hax and D. Candea. *Production and Inventory Management*. Prentice Hall, Inc., 1984.

[Hol75] J.H. Holland. *Adaption in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.

[IS65] E.J. Ignall and L.E. Schrage. Application of the Branch-and-Bound Technique to Some Flow-Shop Scheduling Problems. *Operations Research*, 13(3):400--412, May-June 1965.

[JAMS89] D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partioning. *Operations Research*, 37(6):865--892, November-December 1989.

[JGAM88] V.K. Janakiram, E.F. Gehringer, D.P. Agrawal, and R. Mehrota. A randomized parallel branch-and-bound algorithm. *International Journal of Parallel Programming*, 17(3):277--301, 1988.

[Joh54] S.M. Johnson. Optimal Two- and Three-stage Production Schedules with Setup Times Included. *Naval Research Logist. Quart.*, 1:61--68, 1954.

[Κοπ93] Γ. Κοπιδάκης. Αλγόριθμοι παρεμβολής για το πρόβλημα της ροϊκής παραγωγής, 1993.

[KGV83] S. Kirkpatrick, C.D. Jr Gelat, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671--680, 1983.

[Kim93] Y.D. Kim. A new Branch and Bound algorithm for Minimizing Mean Tardiness in Two-Machine Flowshops. *Computers and Operations Research*, 20(4):391--401, 1993.

[Kir84] S. Kirkpatrick. Optimization by Simulated Annealing: Quantitative Studies. *Journal of Statistical Physics*, 34:975--986, 1984.

[KK88] T. Kawaguchi and S. Kyan. Deterministic Scheduling in Computer Systems: A Survey. *Journal of Operations Research*, 31(2):190--216, June 1988.

[LAL92] P.J.M.van Laarhoven, E.H.L. Aarts, and J. K. Lenstra. Job Shop Scheduling by Simulated Annealing. *Operations Research*, 40(1):113--125, January-February 1992.

[LBG89] M. Laguna, J.W. Barnes, and F.W. Glover. Scheduling Jobs with Linear Delay Penalties and Sequence Dependent Costs and Times Using Tabu Search. Technical Report ORP89-01, The University of Texas at Austin, 1989.

[Lei90] R. Leinsten. Flowshop Sequencing Problems with Limited Buffer Storage. *International Journal of Production Research*, 28(11):2085--2100, 1990.

[LH89] G.E. Liepins and M.R. Hilliard. Genetic Algorithms: Foundations and Applications. *Annals of Operations Research*, 21:31--58, 1989.

[LLKS85] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Willey and Sons, 1985.

[Lom65] Z. Lomnicki. A Branch-and-Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem. *Operational Research Quarterly*, 16(1):89--100, March 1965.

[MB67] G.B. McMahon and P.G. Burton. Flow Shop Scheduling with the Branch and Bound Method. *Operations Research*, 15(3):473--481, May-June 1967.

[MGPO89] M. Malek, M. Guruswamy, M. Pandya, and H. Owens. Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 21:59--84, 1989.

[MGSK88] H. Muehlenbein, M. Gorges-Schleuter, and O. Kraemer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65--85, 1988.

[MP89] C.L. Monma and C.N. Potts. On the Complexity of Scheduling with Batch Setup Times. *Operations Research*, 37, September--October 1989.

[MRR+53] W. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087--1092, 1953.

[MRR93] J. Mittenthal, M. Raghavachari, and A.I. Rana. A Hybrid Simulated Annealing Approach for Single Machine Scheduling Problems with Non-Regular Penalty Functions. *Computers and Operations Research*, 20(2):103--111, 1993.

[MSS89] H. Matsuo, C.J. Suh, and R.S. Sullivan. A Controlled Search Simulated Annealing Method for the Single Machine Weighted Tardiness Problem. *Annals of Operations Research*, 21:85--108, 1989.

[Naw83] M. Nawasz. A Heuristic Algorithm for the *m*-Machine, *n*-Job Flowshop Sequencing Problem. *OMEGA*, 11(1):91--95, 1983.

[OS90] F.A. Ogbu and D.K. Smith. The Application of the Simulated Annealing Algorithm to the Solution of the $n/m/C_{max}$ Flowshop Problem. *Computers and Operations Research*, 17(3):243--253, 1990.

[Pal65] D.S. Palmer. Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time -- A Quick Method of Obtaining a Near Optimum. *Operational Research Quarterly*, 16(1):101--107, March 1965.

[PGD91] C. Proust, N.D. Gupta, and V. Deschamps. Flowshop Scheduling with Setup, Processing and Removal Times separated. *International Journal of Production Research*, 29(3):479--493, 1991.

[Ric83] E. Rich. *Artificial Intelligence*. McGraw-Hill, 1983.

[SD67] R.D. Smith and R.A. Dudek. A General Algorithm for Solution of the *n*-job, *m*-machine Problem of Flow-Shop. *Operations Research*, 15(1):71--82, January-February 1967.

[Sin93] M. Sinclair. Comparison of the performance of modern heuristics for combinatorial optimization on real data. *Computers and Operations Research*, 20(7):687--695, September 1993.

[Szw73] W. Szwarc. Optimal Elimination Methods in the $m \times n$ Flow Shop Scheduling Problem. *Operations Research*, 21(6):1250--1259, November-December 1973.

[Szw78] W. Szwarc. Dominance Conditions for the Three-Machine Flow-Shop Problem. *Operations Research*, 26(1):203--206, January-February 1978.

[Τσα93] Ν. Τσατσάκης. Το πρόβλημα του συνολικού χρόνου περάτωσης N εργασιών σε μια μηχανή με χρόνους εξάρμωσης., 1993.

[TO89] J.M. Troya and M. Ortega. A study of parallel branch-and-bound algorithms with best-bound-first search. *Parallel Computing*, 11:121--126, 1989.

[Wer93] F. Werner. On the Heuristic Solution of the Permutation Flow Shop Problem by Path Algorithms. *Computers and Operations Research*, 20(7):707--722, 1993.

[WH89] M. Widmer and A. Hertz. A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, 41:186--193, 1989.

[Χαρ93] Κ. Χαριτωνίδης. Δικριτηριακός προγραμματισμός σε μία μηχανή με χρόνους εξάρμωσης, 1993.