ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Αλγόριθμοι Παρεμβολής
για το πρόβλημα Ροϊκής Παραγωγής

Γιάννης Κοπιδάκης

Μεταπτυχιακή Εργασία

ΗΡΑΚΛΕΙΟ, ΚΡΗΤΗ
ΟΚΤΩΒΡΙΟΣ 1993

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Αλγόριθμοι Παρεμβολής για το πρόβλημα Ροϊκής Παραγωγής

Εργασία που υποβλήθηκε από τον
ΙΩΑΝΝΗ ΚΟΠΙΔΑΚΗ
ως μερική απαίτηση για την απόκτηση του
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Ηράκλειο, Οκτώβριος 1993

Συγγραφέας:

Τμήμα Επιστήμης Υπολογιστών, 5 Οκτωβρίου 1993

Εισηγητική Επιτροπή:

Πάνος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής, Επόπτης

Στέλιος Ορφανουδάκης, Καθηγητής, Μέλος

Χρήστος Νικολάου, Αναπληρωτής Καθηγητής, Μέλος

Δεκτή:

Πάνος Κωνσταντόπουλος, Αναπληρωτής Καθηγητής,
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Αλγόριθμοι παρεμβολής για το πρόβλημα ροϊκής παραγωγής

Γιάννης Κοπιδάκης
Μεταπτυχιακή εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

Περίληψη

Στην εργασία αυτή μελετούμε τον αιτιοκρατικό προγραμματισμό Ν εργασιών σε γραμμή ροϊκής παραγωγής Μ μηχανών με χρόνους εξάρμωσης ανεξάρτητους ακολουθίας. Κάθε εργασία αναλύεται σε Μ επιμέρους κατεργασίες και κάθε κατεργασία εκτελείται σε διαφορετική μηχανή. Η σειρά εκτέλεσης των κατεργασιών είναι ίδια για όλες τις εργασίες. Οι Ν εργασίες κατανέμονται σε Β ομάδες και μεταξύ δύο διαδοχικών ομοειδών εργασιών δεν απαιτείται εξάρμωση της μηχανής.

Αντιμετωπίζουμε δύο διαφορετικά προβλήματα: την ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης του προγράμματος και την ελαχιστοποίηση της μέγιστης καθυστέρησης των εργασιών ως προς τις προθεσμίες παράδοσής τους. Το πρώτο πρόβλημα συναντάται συχνά στη σχετική βιβλιογραφία και ανήκει στην κατηγορία των δύσκολων συνδυαστικών προβλημάτων, ενώ πολλοί ευρηματικοί αλγόριθμοι έχουν κατά καιρούς προταθεί για την κατά προσέγγιση επίλυσή του. Για το δεύτερο πρόβλημα δεν υπάρχει γνωστή διαδικασία επίλυσης.

Ως κατάλληλο αλγοριθμικό σχήμα και για τα δύο προβλήματα επιλέξαμε τη διαδικασία παρεμβολής εργασιών, η οποία κατασκευάζει σταδιακά το τελικό πρόγραμμα σε Ν βήματα, ερευνώντας πάντα προς την κατεύθυνση με την καλύτερη τιμή του κριτηρίου βελτίστου. Για τον προγραμματισμό εργασιών με προθεσμίες παράδοσης αναπτύσσουμε τη διαδικασία εφικτής παρεμβολής, η οποία επιδιώκει τον εντοπισμό εφικτών ως προς τις προθεσμίες προγραμμάτων με μικρό συνολικό χρόνο εκτέλεσης, ή, όταν αυτό δεν είναι δυνατόν, την κατασκευή προγραμμάτων με μικρές καθυστερήσεις εργασιών. Μ' αυτόν τον τρόπο η διαδικασία πραγματοποιεί δικριτηριακό προγραμματισμό ως προς την ελαχιστοποίηση της μέγιστης καθυστέρησης και του συνολικού χρόνου εκτέλεσης, παρέχοντας απόλυτη προτεραιότητα στο πρώτο κριτήριο

έναντι του δευτέρου.

Στη συνέχεια προτείνουμε δύο νέους μηχανισμούς βελτίωσης των λύσεων που ενσωματώνονται στη διαδικασία παρεμβολής: την αποθήκευση συνόλου μερικών διατάξεων ανά στάδιο και την επαναληπτική εκτέλεση της παρεμβολής με τυχαίες ουρές προγραμματισμού. Για την αξιολόγηση της απόδοσης των βελτιωμένων αλγορίθμων παρεμβολής πραγματοποιήσαμε συστηματικές δοκιμές με τυχαία προβλήματα διαφόρων μεγεθών. Συνολικά παρήχθησαν 2730 τυχαία προβλήματα και 24660 λύσεις από τις παραλλαγές των αλγορίθμων (κάθε πρόβλημα λύθηκε με 9 περίπου διαφορετικούς τρόπους). Για το πρόβλημα της ελαχιστοποίησης του συνολικού χρόνου εκτέλεσης προγράμματος, συγκρίναμε τους αλγορίθμους παρεμβολής με τον αποδοτικότερο γνωστό ευρηματικό αλγόριθμο. Οι βελτιώσεις που πέτυχαν ήταν κατά μέσο όρο της τάξης του 20%-25%, ενώ σε ορισμένες περιπτώσεις έφτασαν το 45%. Παρόλ' αυτά οι αποκλίσεις από τις βέλτιστες λύσεις παρέμειναν σημαντικές, γεγονός που επιβεβαιώνει την αυξημένη δυσκολία του προβλήματος. Από τις παραπάνω μετρήσεις, κρισιμότερος παράγοντας για την ποιότητα των λύσεων αποδείχθηκε ο αριθμός των μηχανών (M), λόγω της εμφάνισης νεκρών χρόνων κατά την εκτέλεση. Οι δοκιμές που πραγματοποιήθηκαν για το πρόβλημα της ελαχιστοποίησης των καθυστερήσεων αφορούσαν το εφικτό των λύσεων και την αποτελεσματικότητα των δύο νέων μηχανισμών.

Τέλος, παραθέτουμε τα συμπεράσματα της μελέτης μας για την απόδοση των αλγορίθμων παρεμβολής για το πρόβλημα ροϊκής παραγωγής, εντοπίζουμε περιπτώσεις μειωμένης απόδοσής τους για εργασίες με προθεσμίες και προτείνουμε τρόπους αντιμετώπισής τους. Εξετάζουμε, επίσης, τη δυνατότητα εφαρμογής των αλγορίθμων παρεμβολής και σε άλλα αιτιοκρατικά προβλήματα διάταξης εργασιών.

Επόπτης : Πάνος Κωνσταντόπουλος
Αναπληρωτής Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστημίου Κρήτης.

Interpolation Algorithms for the Flowshop Scheduling Problem

Yannis Kopidakis
Master of Science Thesis

Department of Computer Science
University of Crete

Abstract

In the present study we consider the deterministic scheduling of N jobs in a flowshop of M machines with sequence indepedent setup times. Each job is further divided into M tasks and each task is executed on a different machine. The order of task execution is identical for each job. The N jobs are distributed over B groups and between two jobs of the same group no machine setup is needed.

We deal with two different problems: the minimization of total execution time and the minimization of maximum job tardiness. The first problem is often discussed in the literature and belongs to the category of hard combinatorial problems. Many heuristic methods have been proposed to provide approximate solutions. For the second problem no algorithm is known.

We selected the job insertion algorithm as a general schema for both problems. The job insertion method constructs the final program in N stages, always leading towards the best value of the optimality criterion. For the due date job scheduling we develop the feasible insertion method in order to produce feasible solutions with short total execution time, or, if this proves impossible, solutions with minimal job tardiness. In other words, the method performs bicriterion scheduling for the minimization of maximum tardiness and total execution time, providing absolute priority to the first criterion.

In addition, we suggest two solution improvement mechanisms which are incorporated into the insertion procedure: the storage of a working set of partial sequences and the iterative insertion with random programming queues. In order to evaluate the performance of the improved insertion algorithms, we performed systematic tests with random problems of varying size. In this way, 2730 random problems were generated and 24660 solutions were produced by different algorithm versions (each problem was solved in about 9 different ways).

For the minimization of total execution time, we compared the insertion algorithms to the best known flowshop heuristic. The mean improvement reported was 20%-25% and in some cases it reached 45%. Yet, deviation from the optimum was not negligible, demonstrating the great difficulty of the problem. The tests have highlighted that the number of machines (M) is the most critical factor for the quality of solutions, due to the presence of machine idle times. The tests related to the problem of maximum tardiness concerned the solution feasibility and the effectiveness of the new mechanisms.

Finally, we present the conclusions on the performance of insertion algorithms for the flowshop problem, we discuss cases of low performance for jobs with due dates and we propose ways of handling those. We also consider the application of the insertion algorithms in other deterministic scheduling problems.

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

.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Πάνο Κωνσταντόπουλο για την πολύτιμη καθοδήγησή του στην εκπόνηση της εργασίας αυτής και, γενικότερα, για την φιλική και παραγωγική συνεργασία μας.

Επίσης, ευχαριστώ τα μέλη της Ομάδας Συστημάτων Στήριξης Αποφάσεων του Ινστιτούτου Πληροφορικής του Ι.Τ.Ε. για τις χρήσιμες ανταλλαγές απόψεων που είχαμε κατά τη διάρκεια της διαμόρφωσης της εργασίας.

Ιδιαίτερα θα ήθελα να ευχαριστήσω την οικογένεια μου για την πολύπλευρη και σταθερή βοήθεια της και τους φίλους και φίλες που μου συμπαραστάθηκαν ηθικά, αλλά και ουσιαστικά, στα προπτυχιακά και μεταπτυχιακά χρόνια μου.

Τέλος, αισθάνομαι την ανάγκη να ευχαριστήσω όλους τους δασκάλους που είχα στη μακρά θητεία μου από την προσχολική στη μεταπτυχιακή εκπαίδευση. Καθένας με τον τρόπο του βρίσκεται μέσα στις ιδέες και το κείμενο της εργασίας αυτής.

Περιεχόμενα

1 Εισαγωγή **9**

1.1 Λεπτομερειακός προγραμματισμός παραγωγής 9

1.2 Ταξινόμηση των προβλημάτων χρονικού προγραμματισμού 10

1.3 Αντικείμενο της εργασίας 11

1.4 Περιεχόμενο της εργασίας 14

1.5 Συμβολισμοί 16

2 Διερεύνηση του προβλήματος ροϊκής παραγωγής **18**

2.1 Υπολογισμός του συνολικού χρόνου εκτέλεσης προγράμματος 18

2.2 Υπολογισμός κάτω φράγματος για τον ελάχιστο χρόνο περάτωσης προγράμματος 19

2.3 Νεκρός χρόνος στις μηχανές 20

2.4 Παρτιδοποίηση εργασιών 21

2.5 Μεταθετικά προγράμματα 24

2.6 Προθεσμίες εργασιών 26

3 Ανασκόπηση αλγορίθμων επίλυσης **27**

3.1 Ο αλγόριθμος του Johnson 27

3.2 Το πρόβλημα για Μ μηχανές 28

3.3 Ευρηματικοί Αλγόριθμοι 30

3.3.1 Αλγόριθμος CDS 30

3.3.2 Αλγόριθμος Dannenbring 31

3.3.3 Αλγόριθμος Nawaz - Προγραμματισμός με Παρεμβολή Εργασιών . 32

3.3.4 Αλγόριθμος σταδιακής κατασκευής τελικού προγράμματος με χρήση κάτω φραγμάτων για το χρόνο εκτέλεσης 34

3.3.5 Συγκριτική Αξιολόγηση των Αλγορίθμων 36

4 Εφικτή παρεμβολή εργασιών με προθεσμίες **38**

4.1 Τροποποίηση της διαδικασίας παρεμβολής - Εφικτή παρεμβολή 39

4.2 Μικτός προγραμματισμός εργασιών με ημερομηνίες παράδοσης ή χωρίς . 42

4.3 Βελτίωση του χρόνου εκτέλεσης 44

4.4 Αντίστροφη εκτέλεση των δύο φάσεων 48

5 Παρεμβολή εργασιών σε σύνολο μερικών διατάξεων **51**

5.1 Απώλεια βελτίστου και εφικτού των λύσεων των αλγορίθμων παρεμβολής 51

5.2 Αποθήκευση συνόλου μερικών διατάξεων ανά στάδιο 55

5.3 Αλγόριθμος παρεμβολής εργασιών σε σύνολο μερικών διατάξεων 58

5.4 Αλγόριθμος εφικτής παρεμβολής εργασιών με προθεσμίες σε σύνολο μερικών διατάξεων 60

6 Επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού **63**

6.1 Απόδοση του κριτηρίου αρχικής διάταξης 63

6.2 Τυχαία διάταξη εργασιών στην ουρά προγραμματισμού 64

6.3 Επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού 65

6.4 Επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού σε σύνολο μερικών διατάξεων 67

6.5 Επαναληπτική εκτέλεση της παρεμβολής ΕΧΠ σε σύνολο μερικών διατάξεων 68

7 Πειραματική αξιολόγηση αλγορίθμων **72**

7.1 Κριτήρια Επιδόσεων 73

7.2 Δοκιμές για το $N/M/F, P, S_{seq-ind}/C_{max}$ 74

7.2.1 Σύγκριση των μηχανισμών βελτίωσης 76

7.2.2 Σύγχρονη λειτουργία των δύο μηχανισμών 81
7.2.3 Υπεροχή του μηχανισμού επαναληπτικής παρεμβολής 84
7.2.4 Αλλαγή κατανομών 86
7.3 Δοκιμές για το $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$ 87
7.4 Υπολογιστικός χρόνος 91

8 Επίλογος **93**

8.1 Απόδοση αλγορίθμων παρεμβολής για το πρόβλημα ροϊκής παραγωγής . . 93
8.2 Προτάσεις βελτιώσεων 94
8.3 Επεκτάσεις 95

A ΠΑΡΑΡΤΗΜΑ. Μετρήσεις για το $N/M/F, P, S_{seq-ind}/C_{max}$ **101**

B ΠΑΡΑΡΤΗΜΑ. Μετρήσεις για το $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$ **108**

1 Εισαγωγή

1.1 Λεπτομερειακός προγραμματισμός παραγωγής

Ο προγραμματισμός παραγωγής αποτελεί σημαντικό πεδίο εφαρμογής της πληροφορικής στη βιομηχανία. Επιδιώκει την αποδοτική αξιοποίηση των διαθέσιμων πόρων της επιχείρησης και προσδιορίζει λεπτομερώς τις εντολές διαχείρισης των μηχανών παραγωγής, των πρώτων υλών και του ανθρώπινου δυναμικού. Η πολυπλοκότητά του και η διαφορετική φύση των λειτουργιών του επιβάλλουν την ιεραρχική οργάνωσή του σε επίπεδα, καθένα από τα οποία παρουσιάζεται με διαφορετικό ρόλο και χρονική εμβέλεια. Συνήθως, διακρίνουμε τον προγραμματισμό παραγωγής σε Στρατηγικό Προγραμματισμό προϊόντων και επενδύσεων, Μεσοπρόθεσμο Προγραμματισμό πρώτων υλών και δυναμικού και Λεπτομερειακό Προγραμματισμό των εργασιών στο σύστημα. Ο Λεπτομερειακός προγραμματισμός λειτουργεί κατά κανόνα με μικρό χρονικό ορίζοντα (εβδομάδων ή ημερών) και αποφασίζει για το τελικό πρόγραμμα εκτέλεσης των εργασιών, άρα συμπεριλαμβάνει το χρονικό προγραμματισμό εργασιών. Επειδή αποτελεί το χαμηλότερο επίπεδο του προγραμματισμού παραγωγής οφείλει να ικανοποιεί τις προδιαγραφές παραγωγής που θέτουν τα υψηλότερα στην ιεραρχία επίπεδα και κατά συνέπεια τα κριτήρια αποδοτικής λειτουργίας του ποικίλουν ανάλογα με τις αποφάσεις που λαμβάνονται στα επίπεδα εκείνα. Για παράδειγμα, ο ορισμός ημερομηνιών παράδοσης των εργασιών υπαγορεύεται συχνά από τον τρόπο λειτουργίας της επιχείρησης και επιβάλλει το χρονικό προγραμματισμό τους έτσι ώστε να περατώνονται κατά το δυνατόν χωρίς καθυστερήσεις.

Ο χρονικός προγραμματισμός εργασιών είναι δυνατόν να πραγματοποιείται είτε από κάποιον ειδικό, συνήθως το μηχανικό παραγωγής, που αποφασίζει για τη χρονική στιγμή εκτέλεσης κάθε εργασίας, είτε από αυτόματα συστήματα, τα οποία χρησιμοποιούν αναλυτικά μοντέλα και αλγορίθμους επίλυσης. Στην πράξη, επειδή η πλήρης παράσταση όλων των κριτηρίων προγραμματισμού είναι αδύνατη, είναι προτιμότερη η χρήση διαλογικών συστημάτων προγραμματισμού, στα οποία μια αρχική λύση παράγεται αυτόματα και βελτιώνεται στη συνέχεια από τον ειδικό ανάλογα με τα ιδιαίτερα χαρακτηριστικά της παραγωγής. Στην παρούσα εργασία θεωρούμε μόνο το αυτόματο μέρος των συστημάτων χρονικού προγραμματισμού και παρουσιάζουμε αλγορίθμους επίλυσης συγκεκριμένων προβλημάτων βασισμένους σε αναλυτική μελέτη των αντίστοιχων μοντέλων.

Τα δεδομένα για τα προβλήματα χρονικού προγραμματισμού στη βιομηχανία (εργασίες προς εκτέλεση, διαθέσιμες μηχανές, χρόνοι επεξεργασίας, ημερομηνίες παράδοσης) θεωρούνται συνήθως πλήρως καθορισμένα και γνωστά κατά τη χρονική στιγμή προγραμματισμού, οπότε και τα προβλήματα χαρακτηρίζονται ως αιτιοκρατικά. Η διαφοροποίηση των παραγωγικών δομών και των κριτηρίων βελτίστου δημιουργεί μια ποικιλία προβλημάτων χρονικού προγραμματισμού. Το πρακτικό ενδιαφέρον των προβλημάτων αυτών είναι δεδομένο, μια και πηγάζουν κατευθείαν από το βιομηχανικό περιβάλλον. Συγχρόνως όμως, παρουσιάζουν και τεράστιο θεωρητικό ενδιαφέρον αφού στην πλειοψηφία τους αποδεικνύονται δύσκολα συνδυαστικά προβλήματα, κατά κανόνα NP-Complete και NP-Hard.

1.2 Ταξινόμηση των προβλημάτων χρονικού προγραμματισμού

Τα προβλήματα χρονικού προγραμματισμού είναι δυνατόν να ταξινομηθούν σε κατηγορίες με διαφορετικό κριτήριο ταξινόμησης. Το σημαντικότερο κριτήριο είναι η πολυπλοκότητα επεξεργασίας στη δομή παραγωγής. Ενας συνηθισμένος διαχωρισμός σύμφωνα με το παραπάνω κριτήριο είναι:

- **μοναδική μηχανή (single processor)**:
  οι εργασίες εκτελούνται σε μια μηχανή.
- **παράλληλες μηχανές (parallel processors)**:
  κάθε εργασία μπορεί να εκτελεστεί εναλλακτικά σε πολλές μηχανές.
- **ροϊκή παραγωγή (flowshop)**:
  κάθε εργασία χωρίζεται σε κατεργασίες κάθεμια από τις οποίες εκτελείται σε ορισμένη μηχανή. Η σειρά εκτέλεσης των κατεργασιών είναι προκαθορισμένη και ίδια για όλες τις εργασίες.
- **γενικό εργοστάσιο (jobshop)**:
  κάθε εργασία χωρίζεται σε κατεργασίες κάθεμια από τις οποίες εκτελείται σε ορισμένη μηχανή. Η σειρά εκτέλεσης των κατεργασιών είναι διαφορετική για κάθε εργασία.

Η ύπαρξη και το είδος των **χρόνων εξάρμωσης (setup times)** των μηχανών αποτελούν διαφορετικό κριτήριο ταξινόμησης των προβλημάτων χρονικού προγραμματισμού. Ως χρόνο εξάρμωσης μηχανής θεωρούμε το χρόνο που μεσολαβεί από την άφιξη μιας

εργασίας στη μηχανή ως την αρχή της επεξεργασίας της. Κατά τη διάρκεια της εξάρμωσης η μηχανή προσαρμόζεται στη φύση της εργασίας, αλλάζοντας είτε εργαλεία, είτε πρώτες ύλες. Συνεπώς, οι χρόνοι εξάρμωσης των μηχανών εξαρτώνται από το είδος της τρέχουσας εργασίας. Ο χρόνος εξάρμωσης ονομάζεται **εξαρτημένος ακολουθίας (sequence-dependent)**, όταν εξαρτάται από τη διαδοχή των εργασιών στο πρόγραμμα, και **ανεξάρτητος ακολουθίας (sequence-independent)**, όταν εξαρτάται μόνο από την εργασία προς εκτέλεση.

Σημαντικό επίσης χαρακτηριστικό ενός προβλήματος χρονικού προγραμματισμού είναι το κριτήριο βελτίστου. Ενδεικτικά αναφέρουμε την ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης προγράμματος (κριτήριο C_{max}), την ελαχιστοποίηση της μέγιστης καθυστέρησης των εργασιών σε σχέση με τις προθεσμίες τους (κριτήριο T_{max}) και την ελαχιστοποίηση του μέσου χρόνου περάτωσης εργασιών (κριτήριο $\sum C_i$).

Προκειμένου να περιγράψουμε συνοπτικά τα προβλήματα χρονικού προγραμματισμού, θα υιοθετήσουμε τον παρακάτω ευρύτατα διαδεδομένο συμβολισμό:

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 ή να περιέχει περισσότερους από ένα περιορισμούς. Επίσης, το πεδίο δ μπορεί να περιλαμβάνει περισσότερα από ένα κριτήρια βελτίστου. Το παραπάνω σχήμα, αν και δεν περιγράφει όλες τις παραμέτρους του προβλήματος, είναι επαρκές στο πλαίσιο της παρούσας εργασίας.

1.3 Αντικείμενο της εργασίας

Η ειδική κατηγορία συνδυαστικών προβλημάτων χρονικού προγραμματισμού που μελετάμε αφορά στην επεξεργασία εργασιών σε αλυσίδα μηχανών. Κάθε εργασία

αποτελείται από επιμέρους κατεργασίες κάθεμια από τις οποίες εκτελείται σε διαφορετική μηχανή. Επειδή η σειρά εκτέλεσης των κατεργασιών είναι η ίδια για όλες τις εργασίες και υπάρχει ενιαία ροή εργασίας, το σύστημα χαρακτηρίζεται ως σύστημα **ροϊκής παραγωγής (flowshop)**. Για τα προβλήματα που εξετάζουμε υποθέτουμε τα παρακάτω:

- Ν εργασίες διαθέσιμες κατά την έναρξη της εκτέλεσης προγραμματίζονται σε Μ μηχανές. Κάθε μηχανή μπορεί να επεξεργάζεται μόνο μια εργασία σε δοσμένη χρονική στιγμή.
- κάθε εργασία αποτελείται από Μ κατεργασίες. Οι κατεργασίες εκτελούνται σε διαφορετική μηχανή η κάθεμια και με την ίδια σειρά για κάθε εργασία. Οι χρόνοι επεξεργασίας των κατεργασιών στις μηχανές είναι γνωστοί εκ των προτέρων. Αν κάποια εργασία δεν διέρχεται από μια μηχανή, θεωρούμε μηδενικό τον αντίστοιχο χρόνο επεξεργασίας.
- κάθε εργασία δεν θεωρείται διαθέσιμη σε μια μηχανή αν δεν έχει ολοκληρωθεί η εκτέλεσή της στην αμέσως προηγούμενη μηχανή.
- η εκτέλεση μιας εργασίας σε δοσμένη μηχανή δεν μπορεί να διακόπτεται (non-preemptive scheduling)
- ο ενδιάμεσος χώρος αποθήκευσης μεταξύ των μηχανών θεωρείται απεριόριστος
- με την επεξεργασία κάθε εργασίας σε κάθε μηχανή συνδέεται και ένας χρόνος εξάρμωσης της μηχανής, ο οποίος απαιτείται για την προσαρμογή της στη φύση της εργασίας. Οι χρόνοι εξάρμωσης θεωρούνται ανεξάρτητοι ακολουθίας.
- οι εργασίες χωρίζονται σε Β ομάδες ανάλογα με το είδος τους. Μεταξύ δύο διαδοχικών ομοειδών εργασιών δεν απαιτείται εξάρμωση της μηχανής. Οι ομάδες των εργασίων είναι ίδιες για όλες τις μηχανές.

Σ' αυτήν την εργασία μας ενδιαφέρουν δύο προβλήματα ροϊκής παραγωγής τα οποία διαφέρουν ως προς την ύπαρξη προθεσμιών παράδοσης και ως προς το κριτήριο βελτίστου. Ακολουθώντας το συμβολισμό της προηγούμενης παραγράφου, θα μελετήσουμε τα:

$N/M/F, S_{seq-ind}/C_{max}$

Θεωρούμε Ν εργασίες για εκτέλεση σε αλυσίδα Μ μηχανών. Στο τρίτο πεδίο του συμβολισμού, με το F δηλώνουμε ότι πρόκειται για γραμμή ροϊκής παραγωγής

(Flowshop), ενώ με το $S_{seq-ind}$ ότι υπάρχουν χρόνοι εξάρμωσης των μηχανών ανεξάρτητοι ακολουθίας. Το κριτήριο βελτίστου είναι η ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης του προγράμματος (C_{max}).

$N/M/F, S_{seq-ind}/T_{max}, C_{max}$

Κάθε εργασία μπορεί να σχετίζεται με προθεσμία παράδοσης που της έχει ανατεθεί από προηγούμενο επίπεδο στην ιεραρχία προγραμματισμού. Εργασίες χωρίς προθεσμίες μπορούν να εκτελεστούν οποιαδήποτε χρονική στιγμή χωρίς να θεωρηθούν καθυστερημένες. Κάθε πρόγραμμα στο οποίο όλες οι εργασίες περατώνονται πριν την προθεσμία τους ονομάζεται εφικτό. Ως πρωτεύον κριτήριο θεωρούμε την ελαχιστοποίηση της μέγιστης καθυστέρησης των εργασιών με προθεσμίες (T_{max}), ενώ η ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης (C_{max}) αποτελεί δευτερεύον κριτήριο βελτίστου.

Η εργασία αυτή πραγματοποιήθηκε εν μέρει στα πλαίσια της συμμετοχής της Ομάδας Συστημάτων Στήριξης Αποφάσεων του Ινστιτούτου Πληροφορικής του Ι.Τ.Ε. στο έργο Distributed Environment for Resource Planning (DERP) του Μεσογειακού Ολοκληρωμένου Προγράμματος Πληροφορικής Ιδιωτικού Τομέα (ΜΟΠΠΙΤ) και είχε ως στόχο εφαρμογής μια γραμμή συναρμολόγησης ψηφιακών κυκλωμάτων, η οποία αποτελεί περίπτωση συστήματος ροϊκής παραγωγής με σχετικά μεγάλους χρόνους εξάρμωσης των μηχανών συναρμολόγησης - συγκόλησης ανάλογα με το είδος του τελικού προϊόντος. Οι υποθέσεις που παρουσιάσαμε παραπάνω συγκροτούν μια απλουστευτική γενίκευση του πραγματικού προβλήματος χρονικού προγραμματισμού της γραμμής το οποίο παρουσιάζεται εξαιρετικά πολύπλοκο και πολυδιάστατο. Οι υποθέσεις αυτές, όμως, έγιναν με σκοπό, από τη μια πλευρά να ανταποκρίνονται κατά το δυνατόν στα πραγματικά δεδομένα της παραγωγής και από την άλλη να συνθέτουν προβλήματα στην κατηγορία των κλασικών συνδυαστικών προβλημάτων διάταξης εργασιών (scheduling) με γενικότερο θεωρητικό ενδιαφέρον.

Ο διαχωρισμός των εργασιών σε δύο κατηγορίες ανάλογα με την ύπαρξη ή όχι προθεσμιών παράδοσης επιβλήθηκε από την ανάγκη για διαφορετική μεταχείριση των εργασιών με προτεραιότητες εκτέλεσης. Εργασίες με προθεσμίες προκύπτουν από επείγουσες παραγγελίες με μικρά χρονικά περιθώρια και η έγκαιρη περάτωσή τους τίθεται ως προϋπόθεση για κάθε αποδεκτή διάταξη των εργασιών. Αντίθετα, για τις εργασίες χωρίς προθεσμίες δεν υπάρχουν χρονικοί περιορισμοί και η θέση τους στην τελική λύση αποφασίζεται με μοναδικό κριτήριο τον καλό συνολικό χρόνο εκτέλεσης. Κατά συνέπεια, στο πρόβλημα $N/M/F, S_{seq-ind}/T_{max}, C_{max}$ επιδιώκεται

καταρχήν το εφικτό ως προς τις προθεσμίες των επειγουσών εργασιών και, όταν αυτό δεν επιτυγχάνεται, η όσο γίνεται μικρότερη καθυστέρησή τους. Η ελαχιστοποίηση του χρόνου εκτέλεσης αποκτά μικρότερη προτεραιότητα έναντι της εξάλειψης των καθυστερήσεων και θεωρείται ως δευτερεύον κριτήριο βελτίστου.

Στις υποθέσεις που παρουσιάσαμε παραπάνω οι χρόνοι εξάρμωσης των μηχανών θεωρούνται ανεξάρτητοι ακολουθίας. Η επιλογή αυτή υπαγορεύτηκε τόσο από τα χαρακτηριστικά της γραμμής παραγωγής που αποτέλεσε την πηγή του γενικότερου θεωρητικού προβλήματος, όσο και από την υπεροχή σε απλότητα συμβολισμών και υπολογισμών του μοντέλου με χρόνους εξάρμωσης ανεξάρτητους ακολουθίας έναντι του μοντέλου με χρόνους εξάρμωσης εξαρτημένους ακολουθίας. Ομως, τα αποτελέσματα και οι αλγόριθμοι που θα παρουσιαστούν επεκτείνονται άμεσα και με προφανή τρόπο για χρόνους εξάρμωσης εξαρτημένους ακολουθίας.

1.4 Περιεχόμενο της εργασίας

Η συνδυαστική πλοκή των προβλημάτων ροϊκής παραγωγής δεν επιτρέπει την ανάπτυξη μεθόδων ακριβούς επίλυσής τους. Στην παρούσα εργασία θα παρουσιάσουμε τη μέθοδο σταδιακής κατασκευής της τελικής λύσης με παρεμβολή εργασιών και θα αναπτύξουμε νέους ευρηματικούς αλγορίθμους για τα προβλήματα $N/M/F, S_{seq-ind}/C_{max}$ και $N/M/F, S_{seq-ind}/T_{max}, C_{max}$ οι οποίοι χρησιμοποιούν τη συγκεκριμένη μέθοδο.

Στη σχετική με το πρόβλημα ροϊκής παραγωγής βιβλιογραφία εξετάζεται σπάνια η περίπτωση εργασιών με προθεσμίες και δεν υπάρχει γνωστή διαδικασία επίλυσης του $N/M/F, S_{seq-ind}/T_{max}, C_{max}$. Για την αντιμετώπισή του τροποποίησαμε τον αλγόριθμο παρεμβολής εργασιών που προτάθηκε από τον M. Nawaz [Naw83] για το πρόβλημα της ελαχιστοποίησης του συνολικού χρόνου εκτέλεσης του προγράμματος. Ο αλγόριθμος αυτός αφενός παράγει σημαντικά καλύτερες λύσεις απ' όλες τις ευρηματικές μεθόδους που κατά καιρούς προτάθηκαν για το $N/M/F, S_{seq-ind}/C_{max}$ και αφετέρου λειτουργεί κατασκευαστικά, επιτρέποντας στη διαδικασία να ελέγχει το εφικτό των διατάξεων ως προς τις προθεμίες και να οδηγεί σε προγράμματα με μικρές καθυστερήσεις εργασιών και καλό συνολικό χρόνο εκτέλεσης. Από τη μετατροπή αυτή προέκυψε ο αλγόριθμος εφικτής παρεμβολής εργασιών που πραγματοποιεί δικριτηριακό προγραμματισμό για το $N/M/F, S_{seq-ind}/T_{max}, C_{max}$ παρέχοντας απόλυτη προτεραιότητα στο κριτήριο της μέγιστης καθυστέρησης έναντι εκείνου του συνολικού χρόνου εκτέλεσης.

Αν και ο αλγόριθμος του Nawaz παρουσιάζει τις καλύτερες επιδόσεις για το

πρόβλημα της ελαχιστοποίησης του συνολικού χρόνου εκτέλεσης, οι λύσεις που παράγει ενδέχεται να απέχουν σημαντικά από το ολικό βέλτιστο, λόγω της αυξημένης πολυπλοκότητας του προβλήματος και της εξάρτησής του από τα δεδομένα. Η μελέτη της φύσης του προβλήματος ροϊκής παραγωγής και της λειτουργίας της διαδικασίας παρεμβολής οδήγησε στο σχεδιασμό δύο νέων μηχανισμών βελτίωσης των λύσεων: του μηχανισμού αποθήκευσης συνόλου μερικών διατάξεων, ο οποίος αποδεικνύεται αποτελεσματικός και για τα δύο προβλήματα που μελετάμε και του μηχανισμού επαναληπτικής παρεμβολής με τυχαίες αρχικές ουρές προγραμματισμού, ο οποίος εφαρμόζεται μόνο για το $N/M/F, S_{seq-ind}/C_{max}$. Η ενσωμάτωση των νέων αυτών μηχανισμών στους αλγορίθμους είχε ως αποτέλεσμα τη δραστική βελτίωση της ποιότητας των λύσεων.

Από συστηματικές δοκιμές των αλγορίθμων για διάφορα μεγέθη τυχαίων προβλημάτων και μεταβαλλόμενες τιμές παραμέτρων προέκυψε ότι οι παραπάνω μηχανισμοί βελτιώνουν την απόδοση των αλγορίθμων παρεμβολής κατά 25% κατά μέσο όρο, ενώ η βελτίωση για ορισμένες κατηγορίες προβλημάτων είναι της τάξης του 45%.

Η σειρά παρουσίασης των παραπάνω θεμάτων στην προκείμενη εργασία αντιστοιχεί σε γενικές γραμμές στη διαδικασία μελέτης του προβλήματος. Στο κεφάλαιο 2 αναλύεται η φύση του προβλήματος ροϊκής παραγωγής και οι ιδιότητες των βέλτιστων λύσεων του. Στη συνέχεια, παρουσιάζονται τα σημαντικότερα γνωστά αναλυτικά αποτελέσματα για το πρόβλημα $N/M/F, S_{seq-ind}/C_{max}$ και οι κυριότεροι από τους ευρηματικούς αλγορίθμους που έχουν προταθεί κατά καιρούς γι' αυτό. Στο κεφάλαιο 4 αναλύεται το πρόβλημα $N/M/F, S_{seq-ind}/T_{max}, C_{max}$ και τεκμηριώνεται ο αλγόριθμος εφικτής παρεμβολής εργασιών. Στα κεφάλαια 5 και 6 αναπτύσσονται δύο νέοι μηχανισμοί βελτίωσης των τελικών λύσεων και ενσωματώνονται στη διαδικασία παρεμβολής: ο μηχανισμός αποθήκευσης συνόλου μερικών διατάξεων και ο μηχανισμός επαναληπτικής παρεμβολής με τυχαίες αρχικές ουρές προγραμματισμού. Στο κεφάλαιο 7 παρουσιάζονται τα αποτελέσματα των πειραματικών συγκρίσεων των αλγορίθμων και αξιολογούνται οι βελτιώσεις των μηχανισμών που προτάθηκαν. Τέλος, στο κεφάλαιο 8, παρατίθενται τα συμπεράσματα της εργασίας για τη φύση του προβλήματος και την αποτελεσματικότητα των αλγορίθμων που αναπτύχθηκαν και προτείνονται επεκτάσεις και άλλες εφαρμογές των αλγορίθμων παρεμβολής.

1.5 Συμβολισμοί

Ακολουθεί κατάλογος με τους απαραίτητους ορισμούς μεγεθών και συμβόλων που χρησιμοποιούνται σ' ολόκληρη την εργασία. Το σχήμα 1 επεξηγεί τη σχέση των μεγεθών αυτών παριστάνοντας γενικά την εκτέλεση ενός προγράμματος εργασιών σε αλυσίδα μηχανών.

N : αριθμός εργασιών

M : αριθμός μηχανών

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 (ισχύει $C_i = C_{iM}$)

G_i : ομάδα στην οποία ανήκει η εργασία i (μεταξύ διαδοχικών εργασιών της ίδιας ομάδας δεν μεσολαβεί χρόνος εξάρμωσης)

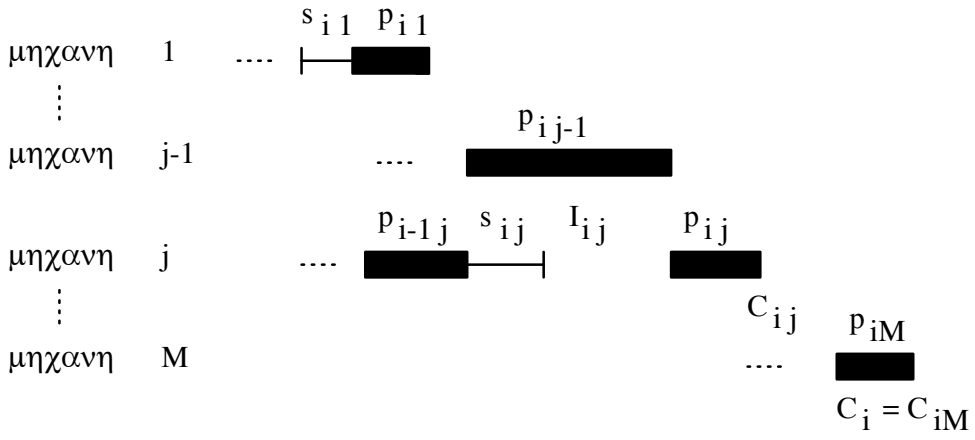
B : αριθμός διαφορετικών ομάδων εργασιών

dd_i : προθεσμία της εργασίας i

T_i : καθυστέρηση της εργασίας i (ισχύει $T_i = C_i - dd_i$, αν $C_i > dd_i$ και $T_i = 0$, αλλιώς)

ΕΜΠ : εργασίες με προθεσμίες

ΕΧΠ : εργασίες χωρίς προθεσμίες



Σχήμα 1: Εκτέλεση προγράμματος σε αλυσίδα μηχανών

2 Διερεύνηση του προβλήματος ροϊκής παραγωγής

2.1 Υπολογισμός του συνολικού χρόνου εκτέλεσης προγράμματος

Για το πρόβλημα ροϊκής παραγωγής, ο συνολικός χρόνος εκτέλεσης προγράμματος N εργασιών σε M μηχανές ταυτίζεται με το μέγιστο χρόνο περάτωσης των εργασιών στην τελευταία μηχανή. Υποθέτοντας ότι $C_{max}(S)$ είναι ο χρόνος περάτωσης του προγράμματος S, ισχύει:

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

Κατά συνέπεια εμφανίζεται απαραίτητος ένας μηχανισμός υπολογισμού του χρόνου περάτωσης των εργασιών σε κάθε μηχανή του συστήματος. Θα υιοθετήσουμε το μηχανισμό που περιγράφεται στο [CGD91] τροποποιημένο κατάλληλα ώστε να συμπεριλάβει χρόνους εξάρμωσης των μηχανών όπως θεωρούνται στο πλαίσιο της παρούσας εργασίας. Γενικά, κατά την άφιξη της εργασίας i στην μηχανή j υπάρχουν δύο ενδεχόμενα:

- η μηχανή j είναι απασχολημένη με την εκτέλεση της προηγούμενης εργασίας του προγράμματος ή της εξάρμωσης της μηχανής για την τρέχουσα. Τότε η εργασία i μένει στον ενδιάμεσο χώρο αποθήκευσης της μηχανής μέχρι να ολοκληρωθεί η εργασία i-1 και η εξάρμωση της μηχανής για την i. Από τη χρονική στιγμή περάτωσης της προηγούμενης εργασίας $C_{i-1\,j}$ ως την ολοκλήρωση της εργασίας i μεσολαβεί χρόνος ίσος με το άθροισμα των χρόνων εξάρμωσης και επεξεργασίας της εργασίας. Στην περίπτωση αυτή ισχύει:

$$C_{i\,j} = C_{i-1\,j} + s_{i\,j} + p_{i\,j} \tag{2}$$

- η μηχανή j είναι διαθέσιμη αμέσως για την εκτέλεση της εργασίας i και η επεξεργασία ξεκινά τη χρονική στιγμή περάτωσης της i στην προηγούμενη μηχανή $C_{i\,j-1}$. Αυτό προϋποθέτει ότι έχει ολοκληρωθεί τόσο η προηγούμενη εργασία i-1, όσο και η εξάρμωση της μηχανής j για την εργασία i. Τότε, για το χρόνο περάτωσης της εργασίας i στη μηχανή j ισχύει:

$$C_{i\,j} = C_{i\,j-1} + p_{i\,j} \tag{3}$$

  Να σημειωθεί ότι στην περίπτωση αυτή προκύπτει νεκρός χρόνος (idle time) για τη μηχανή j μεταξύ των εργασιών i-1 και i, κατά τον οποίο η μηχανή παραμένει ανεκμετάλλευτη.

Στη γενική περίπτωση, συνδυάζοντας τις (2) και (3) προκύπτει ότι ο χρόνος περάτωσης της εργασίας i στη μηχανή j υπολογίζεται ως εξής:

$$C_{i\,j} = max(C_{i\,j-1} + p_{i\,j}, C_{i-1\,j} + s_{i\,j} + p_{i\,j}) \tag{4}$$

Η σχέση (4) μπορεί να χρησιμοποιηθεί για τον υπολογισμό του συνολικού χρόνου εκτέλεσης προγράμματος σε αλυσίδα μηχανών. Ισχύει για χρόνους εξάρμωσης ανεξάρτητους ακολουθίας, αλλά μπορεί να τροποποιηθεί άμεσα για χρόνους εξάρμωσης εξαρτημένους ακολουθίας χωρίς ουσιαστικές αλλαγές. Πρέπει να προστεθεί, επίσης, ότι ο συνολικός χρόνος εκτέλεσης προγράμματος N εργασιών σε M μηχανές ισούται με το χρόνο που το πρόγραμμα απασχολεί την τελευταία μηχανή της αλυσίδας παραγωγής. Αρα, μπορεί να υπολογιστεί ως το άθροισμα των χρόνων επεξεργασίας και εξάρμωσης των εργασιών στην τελευταία μηχανή συν το νεκρό χρόνο στη μηχανή αυτή, το χρόνο δηλαδή που μεσολαβεί μεταξύ της εκτέλεσης δύο διαδοχικών εργασιών. Ετσι:

$$C_{max} = \sum_{i=1}^{N}(p_{i\,M} + s_{i\,M}) + \sum_{i=1}^{N} I_{i\,M} \tag{5}$$

Από τη σχέση (5) γίνεται φανερό ότι η ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης εξασφαλίζεται με παράλληλη εξάλειψη εξαρμώσεων των μηχανών και νεκρών χρόνων στην εκτέλεση του προγράμματος. Στη συνέχεια θα εξετάσουμε χωριστά τους δύο αυτούς παράγοντες στο πλαίσιο των υποθέσεων της παρούσας εργασίας.

2.2 Υπολογισμός κάτω φράγματος για τον ελάχιστο χρόνο περάτωσης προγράμματος

Εστω σ μια μερική διάταξη εργασιών, δηλαδή μια διάταξη l εργασιών με $l < N$. Εστω, επίσης, i εργασία η οποία δεν ανήκει στο σ. Θεωρώντας ως σi το πρόγραμμα που προκύπτει από εκτέλεση της εργασίας i αμέσως μετά τη διάταξη σ και ως C(a,m) το χρόνο περάτωσης της μερικής διάταξης a στη μηχανή m, από τη σχέση (4) προκύπτει:

$$C(\sigma i, m) = max\{C(\sigma i, m-1), C(\sigma, m) + s_{i\,m}\} + p_{i\,m} \tag{6}$$

Αν i αποτελεί υποψήφια εργασία για εκτέλεση μετά τη διάταξη σ και θεωρήσουμε το σύνολο π των υπόλοιπων εργασιών, τότε το πρόγραμμα σiπ αποτελεί διαφορετική λύση για κάθε δυνατή διάταξη των εργασιών του π. Εαν $b\,\epsilon\,\pi$ και b η τελευταία (N-οστη) εργασία του προγράμματος, ισχύει:

$$C(\sigma i\pi, \mathrm{M}) \geq max_{1\leq m\leq M}\{C(\sigma i, m) + \sum_{j\epsilon\pi}(s_{j\,m} + p_{j\,m}) + \sum_{k=m+1}^{M} p_{b\,k}\} \tag{7}$$

Στην παραπάνω σχέση το πρώτο άθροισμα αντιπροσωπεύει το χρόνο εκτέλεσης των εργασιών του συνόλου π στην τρέχουσα μηχανή m, ενώ το δεύτερο άθροισμα το χρόνο εκτέλεσης της τελευταίας εργασίας στις μηχανές μετά την m. Επειδή η σχέση (7) δε συμπεριλαμβάνει νεκρούς χρόνους μηχανών που προκύπτουν από εκτέλεση εργασιών του συνόλου π, μπορεί να χρησιμοποιηθεί για να υπολογίσει ένα κάτω φράγμα για το χρόνο εκτέλεσης όλων των προγραμμάτων που ξεκινούν με εκτέλεση της μερικής διάταξης σi. Συμβολίζουμε με LB(a) το κάτω φράγμα για το χρόνο εκτέλεσης όλων των προγραμμάτων με αρχική μερική διάταξη την a. Τότε:

$$LB(\sigma i) = min_{b \epsilon \pi}\{max_{1 \leq m \leq M}\{C(\sigma i, m) + \sum_{j \epsilon \pi}(s_{j\,m} + p_{j\,m}) + \sum_{k=m+1}^{M} p_{b\,k}\}\} \tag{8}$$

Η παραπάνω ανάλυση επεκτείνεται στον προσδιορισμό κάτω φραγμάτων για το χρόνο εκτέλεσης οποιουδήποτε προγράμματος, χωρίς καμία γνώση γύρω από αρχικές μερικές διατάξεις εργασιών. Θεωρώντας σ = {} και LB(i,b) το κάτω φράγμα για κάθε πρόγραμμα που ξεκινά με την εργασία i και τελειώνει με την εργασία b, είναι:

$$LB(i,b) = max_{1 \leq m \leq M}\{s_{i\,1} - s_{i\,m} + \sum_{k=1}^{m-1} p_{i\,k} + \sum_{j=1}^{N}(s_{j\,m} + p_{j\,m}) + \sum_{k=m+1}^{M} p_{b\,k}\} \tag{9}$$

Από εφαρμογή της παραπάνω σχέσης για κάθε δυνατό ζεύγος εργασιών i,b προκύπτει ένα κάτω φράγμα για το χρόνο εκτέλεσης οποιουδήποτε προγράμματος. Εστω I ένα συγκεκριμένο πρόβλημα της κατηγορίας $N/M/F, S_{seq-ind}/C_{max}$, E(I) το σύνολο των εργασιών του προβλήματος και LB(I) το κάτω φράγμα για το χρόνο εκτέλεσής του. Ισχύει:

$$LB(I) = min\{LB(i,b)\}\ ,\ \forall i, b\, \epsilon\, E(I) \tag{10}$$

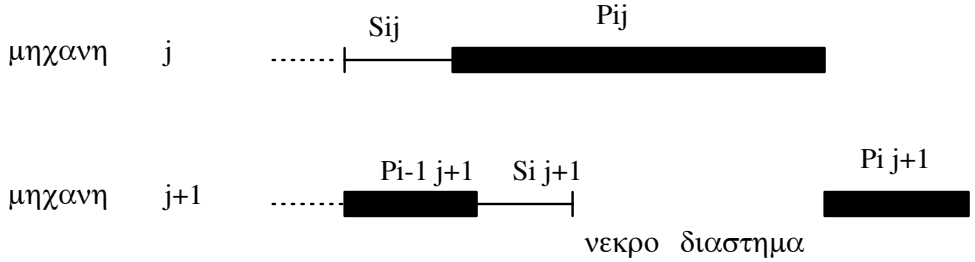
2.3 Νεκρός χρόνος στις μηχανές

Οπως αναφέρθηκε στην παράγραφο 2.1, κατά την εκτέλεση εργασιών σε αλυσίδα μηχανών είναι δυνατόν να προκύψει χρονικό διάστημα μέσα στο οποίο μια μηχανή του συστήματος δεν χρησιμοποιείται. **Νεκρός χρόνος (idle time)** παρουσιάζεται σε περίπτωση που κάποια μηχανή έχει ολοκληρώσει τόσο την τρέχουσα κατεργασία της, όσο και την εξάρμωση για την επόμενη, ενώ η επεξεργασία της επόμενης εργασίας στην προηγούμενη μηχανή δεν έχει ακόμα τελειώσει. Οι νεκροί χρόνοι των μηχανών αποδεικνύονται σημαντική πηγή καθυστερήσεων της παραγωγής. Επιπλέον μειώνουν το βαθμό χρησιμοποίησης του μηχανολογικού εξοπλισμού του εργοστασίου και δημιουργούν ανεκμετάλλευτη δυναμικότητα.

Η εμφάνιση νεκρών διαστημάτων στη γραμμή παραγωγής οφείλεται στην ανομοιομορφία των χρόνων επεξεργασίας των εργασιών στις μηχανές. Εστω ότι η εργασία i εκτελείται στη μηχανή j και παράλληλα η προηγούμενή της στο πρόγραμμα, i-1, εκτελείται στη μηχανή j+1. Αν

$$s_{i\,j} + p_{i\,j} \gg p_{i-1\,j+1} + s_{i\,j+1}$$

τότε το ενδεχόμενο να περιμένει η μηχανή j+1 για την εργασία i είναι πολύ πιθανό. Η περίπτωση απεικονίζεται στο σχήμα 2. Προφανώς, ομοιομορφία στους χρόνους επεξεργασίας συνεπάγεται ομαλή ροή του προγράμματος και μικρή πιθανότητα εμφάνισης νεκρών χρόνων.



Σχήμα 2: Εμφάνιση νεκρού χρόνου λόγω ανομοιομορφίας των χρόνων επεξεργασίας

Στην πράξη η κατανομή των χρόνων επεξεργασίας των εργασιών διαφέρει σημαντικά από μηχανή σε μηχανή, λόγω της διαφορετικής φύσης των λειτουργιών που εκτελεί η κάθεμιά. Αντίστοιχες διαφορές υπάρχουν και στους χρόνους εξάρμωσης. Κατά συνέπεια το πρόβλημα παρουσιάζεται εξαιρετικά πολύπλοκο, ειδικά για μεγάλο αριθμό μηχανών όπου ενδέχεται να εμφανίζονται διαφορές στους χρόνους επεξεργασίας σε περισσότερα από ένα σημεία της γραμμής παραγωγής. Στην περίπτωση αυτή η εξάλειψη του νεκρού χρόνου για μια μηχανή μπορεί να συνεπάγεται εμφάνιση νεκρού χρόνου για άλλη μηχανή, θεωρώντας πάντα το ίδιο ζεύγος διαδοχικών εργασιών.

2.4 Παρτιδοποίηση εργασιών

Από τη σχέση (5) της 2.1 προκύπτει ότι στην ελαχιστοποίηση του χρόνου περάτωσης του προγράμματος σε αλυσίδα μηχανών συμβάλλει σε μεγάλο βαθμό η εξάλειψη των εξαρμώσεων των μηχανών, όπου αυτή είναι δυνατή. Επειδή κατά κανόνα τα προϊόντα παράγονται σε μεγάλο αριθμό μονάδων, οι χρόνοι εξάρμωσης παρουσιάζονται πολύ μικρότεροι από τους χρόνους επεξεργασίας στην πράξη. Παρόλ' αυτά, αποτελούν σημαντική καθυστέρηση στη διαδικασία παραγωγής. Προκειμένου να αποφευχθεί

η εξάρμωση των μηχανών, εργασίες που ανήκουν στην ίδια ομάδα είναι δυνατό να ομαδοποιούνται και να τοποθετούνται σε διαδοχικές θέσεις στο τελικό πρόγραμμα. Μ' αυτόν τον τρόπο δεν απαιτείται προσαρμογή των μηχανών ή αλλαγή εργαλείων και η εκτέλεση των εργασιών γίνεται χωρίς διακοπή. Η ομαδοποίηση των ομοειδών εργασιών ώστε να εκτελούνται συνεχόμενα χωρίς εξάρμωση ονομάζεται παρτιδοποίηση. Αντίστοιχα, παρτίδα ονομάζεται μια ομάδα διαδοχικών ομοειδών εργασιών στο τελικό πρόγραμμα.

Σε προβλήματα προγραμματισμού εργασιών σε μια μηχανή η παρτιδοποίηση αποτελεί το κρισιμότερο σημείο. Συνήθως, επιδιώκεται το μέγεθος παρτίδας να μεγιστοποιείται και άρα, ο συνολικός χρόνος εξάρμωσης για τη μηχανή να ελαχιστοποιείται. Να σημειωθεί ότι ο συνολικός χρόνος εξάρμωσης επηρεάζεται και από τη σειρά διαδοχής των παρτίδων. Η σειρά εκτέλεσης των ομοειδών εργασιών μέσα σε μια παρτίδα καθορίζεται από το κριτήριο βελτίστου που υποθέτουμε. Το μέγεθος των παρτίδων για προγράμματα που εκτελούνται σε μια μηχανή μπορεί να περιορίζεται από το κριτήριο βελτίστου που επιλέγεται (π.χ. ελαχιστοποίηση της μέγιστης ή συνολικής καθυστέρησης για εργασίες με προθεσμίες, ελαχιστοποίηση του μέσου χρόνου διέλευσης). Οταν επιδιώκεται η ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης σε μια μηχανή (C_{max}) με χρόνους εξάρμωσης ανεξάρτητους ακολουθίας, είναι φανερό ότι στη βέλτιστη λύση όλες οι ομοειδείς εργασίες εκτελούνται διαδοχικά. Τότε, ο αριθμός των παρτίδων της τελικής λύσης ισούται με τον αριθμό των ομάδων και το μέγεθός τους είναι το μέγιστο δυνατό [HC84].

Για το πρόβλημα ροϊκής παραγωγής, η κρισιμότητα του μηχανισμού παρτιδοποίησης εργασιών μειώνεται κατά πολύ. Ο λόγος γι' αυτό είναι ότι καθώς προσπαθούμε να αποφύγουμε το χαμένο χρόνο λόγω εξάρμωσης των μηχανών επιβάλλοντας διαδοχική εκτέλεση ομοειδών εργασιών ενδέχεται να παρουσιάζονται περισσότερα νεκρά διαστήματα στο πρόγραμμα και ο συνολικός νεκρός χρόνος να καθυστερήσει την περάτωσή του. Θ' αποδείξουμε την παρακάτω πρόταση:

- **Με κριτήριο την ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης, στο βέλτιστο πρόγραμμα εκτέλεσης σε σειρά Μ μηχανών Ν εργασιών που ανήκουν σε Β ομάδες, οι ομοειδείς εργασίες δεν εκτελούνται κατ' ανάγκη διαδοχικά.**

  Θ' αποδείξουμε την πρόταση με τη βοήθεια του αντιπαραδείγματος του σχήματος 3. Οι εργασίες 1 και 2 ανήκουν στην ίδια ομάδα και η διαδοχική εκτέλεσή τους γίνεται χωρίς εξάρμωση σε όλες τις μηχανές. Παρόλ' αυτά, το βέλτιστο πρόγραμμα για τα δεδομένα του σχήματος 3 είναι το 1-3-2, στο οποίο οι εργασίες 1 και 2 δεν

ανήκουν σε μια παρτίδα, αλλά μεταξύ τους παρεμβάλλεται η 3.

Χρονοι Επεξεργασιας

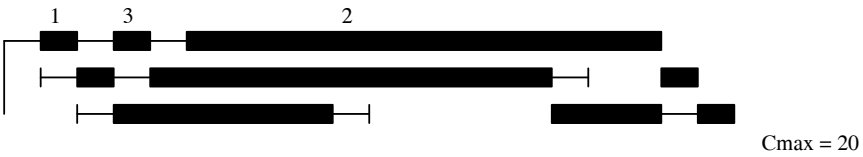
εργ	M1	M2	M3
1	1	1	6
2	13	1	1
3	1	11	3

Χρονοι Εξαρμωσης

εργ	M1	M2	M3
1	1	1	1
2	1	1	1
3	1	1	1

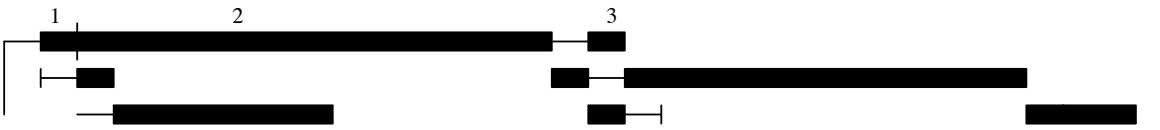
Ομαδες Εργασιων : G(1) = G(2) = 1
G(3) = 2

Βελτιστο προγραμμα = 1– 3 – 2 :



Cmax = 20

Ομαδοποιημενες Εργασιες στο : 1 – 2 – 3 :



$C_{max} = 31$

Σχήμα 3: Οι ομοειδείς εργασίες δεν εκτελούνται διαδοχικά στο βέλτιστο πρόγραμμα

Κατά συνέπεια, η παρτιδοποίηση έχει θετικά αποτελέσματα αν συνδυάζεται με κάποια ομοιομορφία των χρόνων επεξεργασίας των εσωτερικών της κάθε παρτίδας εργασιών. Αντίθετα, αν υπάρχει μεγάλη ανομοιομορφία στους χρόνους επεξεργασίας και κατά συνέπεια μεγάλα νεκρά διαστήματα, η εξάρμωση των μηχανών μπορεί να εκτελείται σε νεκρό χρόνο χωρίς να επηρεάσει το συνολικό χρόνο εκτέλεσης. Γενικά, σε προβλήματα με μεγάλους χρόνους εξάρμωσης σε σχέση με τους χρόνους επεξεργασίας επιδιώκεται μεγάλο μέγεθος παρτίδας, ενώ σε προβλήματα με μικρούς χρόνους εξάρμωσης αποκτά μεγαλύτερο βάρος η εξάλειψη των νεκρών χρόνων.

Ο τρόπος υπολογισμού του συνολικού χρόνου εκτέλεσης προγράμματος όπως παρουσιάστηκε στην 2.1, είναι αναγκαίο να τροποποιηθεί προκειμένου να μη συμπεριλάβει την εκτέλεση των εξαρμώσεων μεταξύ ομοειδών εργασιών. Οι χρόνοι εξάρμωσης θεωρούνται ανεξάρτητοι ακολουθίας, αν και οι τύποι υπολογισμού γενικεύονται εύκολα για χρόνους εξάρμωσης εξαρτημένους ακολουθίας. Ετσι, η σχέση

(4) παίρνει τη μορφή:

$$C_{i\,j} = max(C_{i\,j-1} + p_{i\,j}, C_{i-1\,j} + p_{i\,j} + s'_{i\,j}) \tag{11}$$

με

$$s'_{i\,j} = \begin{cases} 0 & \text{αν G(i) = G(i-1)} \\ s_{i\,j} & \text{αλλιώς} \end{cases}$$

όπου $G(i)$ είναι η ομάδα στην οποία ανήκει η εργασία i.

2.5 Μεταθετικά προγράμματα

Μεταθετικό πρόγραμμα (permutation schedule) για αλυσίδα μηχανών ονομάζεται το πρόγραμμα στο οποίο η σειρά διέλευσης των εργασιών είναι η ίδια για κάθε μηχανή. Δεν επιτρέπονται, δηλαδή, αλλαγές στη σειρά εκτέλεσής τους από μηχανή σε μηχανή. Κατά συνέπεια, ένα μεταθετικό πρόγραμμα N εργασιών περιγράφεται πλήρως από μια διάταξη εργασιών, μέλος του συνόλου των N! διαφορετικών δυνατών μεταθέσεων τους.

Σ' ένα **μη μεταθετικό πρόγραμμα (non-permutation schedule)** επιτρέπονται αλλαγές στη σειρά διέλευσης των εργασιών από τις μηχανές. Ετσι, ενδέχεται οι εργασίες να εκτελούνται με διαφορετική σειρά σε κάθε μηχανή. Τα μη μεταθετικά προγράμματα περιγράφονται από διαφορετική διάταξη των N εργασιών σε κάθεμια από τις M μηχανές. Το σύνολο όλων των δυνατών μη μεταθετικών προγραμμάτων αριθμεί $(N!)^M$ στοιχεία και περιέχει προφανώς το σύνολο των μεταθετικών προγραμμάτων.

Τα μη μεταθετικά προγράμματα αποτελούν γενικά καλύτερες λύσεις για το πρόβλημα της ελαχιστοποίησης του συνολικού χρόνου εκτέλεσης σε αλυσίδα μηχανών. Αυτό οφείλεται στις δυνατότητες που προσφέρουν για μείωση των νεκρών χρόνων στις μηχανές. Η ελαστικότητα σε αναδιατάξεις από μηχανή σε μηχανή βοηθά την προσαρμογή του προγράμματος στις ανομοιομορφίες των χρόνων επεξεργασίας. Αντίθετα, κάποιο μεταθετικό πρόγραμμα ενδέχεται να εμφανίζει νεκρούς χρόνους σε συγκεκριμένη μηχανή, ενώ διαφορετική διάταξη των εργασιών μπορεί να συνεπάγεται εξάλειψή τους από τη συγκεκριμένη μηχανή με παράλληλη εμφάνιση νεκρών χρόνων σε άλλη.

Στην παρούσα εργασία, θα μας απασχολήσουν μόνο μεταθετικές λύσεις για το πρόβλημα M μηχανών. Οι λόγοι γι' αυτήν την επιλογή είναι δύο:

- Οι συνθήκες παραγωγής δεν επιτρέπουν συχνά αναδιατάξεις εργασιών στις ουρές αποθήκευσης των υπό κατεργασία προϊόντων. Η εκτέλεση μη μεταθετικών

προγραμμάτων προϋποθέτει απαραίτητα διαχείριση των ταμιευτήρων ενδιάμεσης αποθήκευσης, ώστε σε κάθε στάδιο της παραγωγής ν' αλλάζει η σειρά διέλευσης των εργασιών. Σε γραμμές παραγωγής με μεγάλο βαθμό αυτοματοποίησης, όπου οι ουρές αναμονής στα ενδιάμεσα στάδια είναι του τύπου FIFO (First-In-First-Out), αυτό δεν είναι εφικτό και η ροή εργασίας παίρνει αναγκαστικά τη μορφή μεταθετικού προγράμματος.

- Το σύνολο των μεταθετικών λύσεων είναι πολύ μικρότερο απ' αυτό των μη μεταθετικών. Ετσι, περιορίζοντας το χώρο έρευνας για το βέλτιστο διευκολύνεται κατά πολύ η διαδικασία προγραμματισμού. Το μεγαλύτερο μέρος της βιβλιογραφίας για το πρόβλημα ροϊκής παραγωγής θεωρεί μόνο μεταθετικά προγράμματα και οι υπάρχοντες αλγόριθμοι επίλυσής του παράγουν αποκλειστικά μεταθετικές λύσεις.

Αν ενσωματώσουμε στο συμβολισμό της παραγράφου 1.1 τον περιορισμό των λύσεων σε μεταθετικά προγράμματα, τα προβλήματα που μελετάμε στην εργασία αυτή γίνονται αντίστοιχα $N/M/F, P, S_{seq-ind}/C_{max}$ και $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$, όπου το P δηλώνει μεταθετικά προγράμματα.

Θεωρητικό ενδιαφέρον παρουσιάζει η σύγκριση μεταθετικών και μη μεταθετικών λύσεων για το πρόβλημα ροϊκής παραγωγής. Εστω η_P ο λόγος απόδοσης των μεταθετικών προγραμμάτων για πολλές μηχανές με:

$$\eta_P = \frac{C_{max}(P)}{C^*_{max}}$$

όπου $C_{max}(P)$ ο συνολικός χρόνος εκτέλεσης οποιουδήποτε μεταθετικού προγράμματος και C^*_{max} ο συνολικός χρόνος εκτέλεσης του βέλτιστου προγράμματος. Αποδεικνύεται [PSW91] ότι υπάρχει οικογένεια προβλημάτων M μηχανών με M = 2N, για τα οποία ο λόγος απόδοσης μεταθετικών προγραμμάτων δεν φράσσεται προς τα πάνω από καμιά σταθερά. Συγκεκριμένα:

$$\eta_P \geq \frac{\sqrt{M} + \frac{1}{2}}{2}$$

Η παραπάνω σχέση φανερώνει σημαντική διαφορά στην απόδοση μεταθετικών και μη μεταθετικών προγραμμάτων. Ομως, ο υπολογισμός της στηρίζεται σε ανάλυση της χειρότερης περίπτωσης και πραγματοποιείται με τη θεωρητική κατασκευή ενός προβλήματος ροϊκής παραγωγής με ειδική δομή και αριθμητικά δεδομένα, που αποτελούν μάλλον μη ρεαλιστική περίπτωση αλυσίδας μηχανών. Στην πράξη οι συνθήκες παραγωγής και τα μεγέθη που εμφανίζονται επιβάλλουν ομαλή ροή εργασίας για τα

μεταθετικά προγράμματα και ο λόγος απόδοσης τους εμφανίζεται σημαντικά μικρότερος. Αυτό αποδεικνύουν και τα αποτελέσματα των δοκιμών που πραγματοποιήθηκαν στα πλαίσια της εργασίας και παρουσιάζονται στο κεφάλαιο 7. Στις δοκιμές αυτές με μεγάλο αριθμό τυχαίων προβλημάτων (ο αριθμός μηχανών κυμαίνονταν από 4 έως 16 και ο αριθμός εργασιών από 6 έως 50) ο μέσος λόγος απόδοσης των μεταθετικών προγραμμάτων που παρήχθησαν δεν ξεπέρασε για όλα τα μεγέθη προβλημάτων την τιμή 1.2.

2.6 Προθεσμίες εργασιών

Το πρόβλημα ροϊκής παραγωγής με προθεσμίες εργασιών δεν έχει μελετηθεί αναλυτικά στη βιβλιογραφία. Παρουσιάζεται εξαιρετικά πολύπλοκο λόγω της δυσκολίας ανάθεσης προθεσμιών παράδοσης στις ενδιάμεσες κατεργασίες.

Η προθεσμία κάθε εργασίας αναφέρεται στην τελική κατεργασία της, δηλαδή σχετίζεται με το χρόνο περάτωσης της στην τελευταία μηχανή. Προκειμένου να προγραμματιστεί η χρονική στιγμή έναρξης μιας εργασίας στην πρώτη μηχανή της αλυσίδας, είναι αναγκαίο να προσδιοριστούν προθεσμίες της για κάθε μηχανή. Αυτό θα μπορούσε να πραγματοποιηθεί με ένα μηχανισμό ανάθεσης προθεσμιών με φορά αντίστροφη της παραγωγής. Ο μηχανισμός αυτός θα αφαιρούσε τους χρόνους επεξεργασίας και εξάρμωσης στην τελευταία μηχανή για κάθε εργασία από την προθεσμία της τελευταίας κατεργασίας της και θα υπολόγιζε την προθεσμία της αμέσως προηγούμενης κατεργασίας. Ετσι αναδρομικά η αναγωγή θα έφτανε στην πρώτη κατεργασία και κατά συνέπεια θα μπορούσε να προσδιοριστεί η βραδύτερη δυνατή χρονική στιγμή εισόδου κάθε εργασίας στο σύστημα, ώστε αυτή να μην χάνει την προθεσμία της.

Ο μηχανισμός που περιγράφηκε παραπάνω δεν λειτουργεί στην πράξη λόγω της εμφάνισης νεκρών χρόνων μεταξύ της εκτέλεσης διαδοχικών εργασιών στις μηχανές. Οι νεκροί χρόνοι δεν μπορούν σε καμιά περίπτωση να υπολογιστούν εκ των προτέρων, αφού εξαρτώνται από τη σειρά εκτέλεσης των εργασιών και τους σχετικούς χρόνους επεξεργασίας τους. Αρα ο προγραμματισμός εργασιών με προθεσμίες για αλυσίδα παραγωγής διαφέρει ριζικά από τον αντίστοιχο για μια μηχανή. Θα μελετήσουμε αυτό το πρόβλημα στο κεφάλαιο 4.

3 Ανασκόπηση αλγορίθμων επίλυσης

3.1 Ο αλγόριθμος του Johnson

Το πρόβλημα προγραμματισμού εργασιών σε αλυσίδα μηχανών αντιμετωπίστηκε για πρώτη φορά από τον S. Johnson το 1954 [Joh54]. Ο Johnson θεώρησε N εργασίες που πρέπει να υποστούν κατεργασία σε δύο στάδια (M = 2) με τους χρόνους εξάρμωσης των μηχανών σταθερούς και υποχρεωτικούς για κάθε κατεργασία. Με την παραπάνω υπόθεση οι χρόνοι εξάρμωσης ενσωματώνονται στους χρόνους επεξεργασίας.

Από αυτήν την πρώτη προσέγγιση του προβλήματος προέκυψε και το σημαντικότερο αναλυτικό αποτέλεσμα για την εύρεση βέλτιστων λύσεων σε προβλήματα $N/2/F/C_{max}$. Ο Johnson απέδειξε ότι η βέλτιστη λύση για δύο στάδια είναι ένα μεταθετικό πρόγραμμα στο οποίο η εργασία i προηγείται της εργασίας j αν ισχύει :

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

Η παραπάνω πρόταση υποδεικνύει την τοποθέτηση στην αρχή του προγράμματος εργασιών με μικρό χρόνο επεξεργασίας στην πρώτη μηχανή , ενώ στο τέλος του προγράμματος αυτών με μικρό χρόνο επεξεργασίας στη δεύτερη μηχανή. Σε περίπτωση ισότητας για το κριτήριο της πρότασης δεν έχει σημασία ποιά από τις εργασίες προηγείται και τα προγράμματα που προκύπτουν από διαφορετικές διατάξεις των εργασιών αυτών έχουν τον ίδιο συνολικό χρόνο εκτέλεσης. Χρησιμοποιώντας την παραπάνω πρόταση, ο Johnson διατύπωσε τον παρακάτω πολυωνυμικό αλγόριθμο, ο οποίος παράγει βέλτιστες λύσεις για το πρόβλημα δύο σταδίων :

Βήμα 1 Ψάξε τον πίνακα των χρόνων επεξεργασίας των εργασιών στις μηχανές και προσδιόρισε τον ελάχιστο χρόνο επεξεργασίας.

Βήμα 2 Αν 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 Ψάξε τους χρόνους επεξεργασίας των εργασιών που δεν έχουν προγραμματιστεί και προσδιόρισε τον ελάχιστο χρόνο. Σε περίπτωση ισότητας των τιμών των χρόνων επεξεργασίας, δεν έχει σημασία ποιόν θα θεωρήσουμε ελάχιστο.

Βήμα 4 Αν αυτός ανήκει σε κατεργασία της πρώτης μηχανής, τοποθέτησε την αντίστοιχη εργασία μετά τις ήδη προγραμματισμένες στην αρχή του προγράμματος. Αν αυτός

ανήκει σε κατεργασία της δεύτερης μηχανής, τοποθέτησε την αντίστοιχη εργασία πριν τις ήδη προγραμματισμένες στο τέλος του προγράμματος.

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

Ο Johnson απέδειξε επίσης ότι στη βέλτιστη λύση για το πρόβλημα Μ σταδίων οι εργασίες εκτελούνται χωρίς αναδιατάξεις στις δύο πρώτες και στις δύο τελευταίες μηχανές. Από την παραπάνω πρόταση προκύπτει το συμπέρασμα ότι το βέλτιστο πρόγραμμα για τρία στάδια είναι μεταθετικό, αφού η σειρά εκτέλεσης των εργασιών είναι ίδια για τις δύο πρώτες μηχανές (1η και 2η), επίσης για τις δύο τελευταίες (2η και 3η) και άρα μεταβατικά για όλες.

Το πρόβλημα $N/3/F/C_{max}$, όπου τα στάδια παραγωγής είναι τρία, λύνεται βέλτιστα από τον αλγόριθμο του Johnson στην ειδική περίπτωση που όλες οι κατεργασίες στη δεύτερη μηχανή έχουν μικρότερους χρόνους επεξεργασίας από κάθε κατεργασία της πρώτης ή της τρίτης μηχανής. Υποθέτουμε, δηλαδή, ότι σε καμιά περίπτωση η δεύτερη μηχανή δεν είναι η κρισιμότερη ως προς το φόρτο εργασίας. Αν λοιπόν ισχύει :

$min(p_{i1}) \geq max(p_{j2})$ για i=1,..,N, j=1,..,N

ή

$min(p_{i3}) \geq max(p_{j2})$ για i=1,..,N, j=1,..,N

τότε ο Johnson έδειξε ότι ο παραπάνω αλγόριθμος μπορεί να εφαρμοστεί αν δημιουργήσουμε ένα νέο πρόβλημα δύο σταδίων με μετασχηματισμένους χρόνους επεξεργασίας για την εργασία κ ως εξής : $p'_{k1} = p_{k1} + p_{k2}$ και $p'_{k2} = p_{k2} + p_{k3}$. Οι λύσεις που παράγονται από τον αλγόριθμο για το νέο πρόβλημα δύο μηχανών είναι βέλτιστες για το αρχικό πρόβλημα των τριών μηχανών.

3.2 Το πρόβλημα για Μ μηχανές

Η θεωρητική απλότητα καθώς και η πρακτική σημασία του αλγορίθμου του Johnson, ο οποίος σε πολυωνυμικό χρόνο λύνει βέλτιστα το πρόβλημα για δύο μηχανές, έδωσαν μεγάλη ώθηση στην έρευνα στον τομέα των αιτιοκρατικών προβλημάτων διάταξης εργασιών σε Μ μηχανές. Ομως, παρά τη σημαντική προσπάθεια προς αυτήν την κατεύθυνση, κανένα ουσιαστικό βήμα δεν πραγματοποιήθηκε προς την επέκταση των αναλυτικών αποτελεσμάτων για το γενικότερο πρόβλημα ροϊκής παραγωγής [DPS92]. Το

πρόβλημα για Μ στάδια επεξεργασίας φαίνεται να χάνει τη δομή και τις καλές ιδιότητες που παρουσιάζει για 2 στάδια.

Η επιβεβαίωση της παραπάνω υπόθεσης προήλθε από τη θεωρία της υπολογιστικής πλοκής. Το πρόβλημα της βέλτιστης διάταξης Ν εργασιών σε Μ μηχανές με κριτήριο την ελαχιστοποίηση του C_{max} αποδείχθηκε ότι ανήκει στην κατηγορία των NP-Complete προβλημάτων. Η απόδειξη για την περίπτωση προγραμμάτων χωρίς διακοπές (nonpreemptive) έγινε το 1976 από τους M.R. Garey, D.S. Johnson και R.Sethi [GJS76], ενώ για την περίπτωση προγραμμάτων με διακοπές (preemptive) το 1978 από τους T. Gonzalez και S. Sahni [GS78]. Ετσι, η αναζήτηση βέλτιστων λύσεων καθίσταται πρακτικά αδύνατη για μεγάλα μεγέθη προβλημάτων και κατ' ανάγκη η προσπάθεια στρέφεται προς τη χρήση γενικών μεθόδων βελτιστοποίησης ή ευρηματικών αλγορίθμων για την προσεγγιστική επίλυσή τους.

Από τις προσπάθειες εφαρμογής γενικών μεθόδων βελτιστοποίησης κυριότερη είναι η προσαρμογή της διαδικασίας **διαμερισμού και φραγής (branch-and-bound)** για την εύρεση υποβέλτιστων λύσεων για το πρόβλημα. Η μέθοδος αυτή υπήρξε αρκετά δημοφιλής κατά τις δεκαετίες του '60 και '70, αλλά από τις σχετικές μελέτες δεν προέκυψαν θεαματικά αποτελέσματα για αποδοτικά φράγματα και κανόνες περιορισμού του χώρου έρευνας [Bak75, Ash70, SD67, GW64, IL65, MB67, Gup71]. Η εξαγωγή ιδιοτήτων των βέλτιστων λύσεων του προβλήματος στη γενική μορφή του αποδείχθηκε εξαιρετικά δύσκολη λόγω της πολυπλοκότητάς του. Η εξάρτηση της δυσκολίας του προβλήματος από τους χρόνους επεξεργασίας των εργασιών στις μηχανές περιόρισε σημαντικά την αποτελεσματικότητα των μηχανισμών υπολογισμού φραγμάτων για τις λύσεις, καθώς και των κανόνων αποκλεισμού καταστάσεων από το χώρο έρευνας.

Αξιόλογες λύσεις για το πρόβλημα ροϊκής παραγωγής παρουσίασε μια σχετικά πρόσφατη γενική μέθοδος βελτιστοποιήσης, η μέθοδος **taboo search**. Η μέθοδος εντάσσεται στην κατηγορία επαναληπτικών διαδικασιών βελτίωσης της λύσης συνδυαστικών προβλημάτων με τοπική έρευνα (local search). Πραγματοποιεί αντιμεταθέσεις εργασιών κρατώντας συνεχώς μια λίστα κινήσεων που απαγορεύονται στην τρέχουσα επανάληψη (taboo list). Ετσι, η έρευνα του συνόλου των προγραμμάτων κατευθύνεται και αποφεύγονται κύκλοι στη διαδικασία, καθώς και η παγίδευσή της σε τοπικά βέλτιστα. Μια προσαρμογή της γενικής μεθόδου taboo search σε προβλήματα προγραμματισμού εργασιών σε αλυσίδα μηχανών παρουσιάζεται στο [WH89] και συγκρίνεται με άλλες μεθόδους επίλυσης. Από τη συγκεκριμένη μελέτη προκύπτει ότι η μέθοδος υπερέχει έναντι των γνωστών ευρηματικών μεθόδων, αλλά ο υπερβολικά

μεγάλος απαιτούμενος υπολογιστικός χρόνος για την εκτέλεσή της δεν επιτρέπει την εφαρμογή της σε προβλήματα μεγάλου μεγέθους (για 20 εργασίες ο υπολογιστικός χρόνος ανέρχεται κατά μέσο όρο σε 695 δευτερόλεπτα).

3.3 Ευρηματικοί Αλγόριθμοι

Η αποτυχία των συστηματικών μεθόδων βελτιστοποίησης να αντιμετωπίσουν το πρόβλημα ροϊκής παραγωγής για μεγάλο αριθμό εργασιών έστρεψε το ενδιαφέρον στην ανάπτυξη ευρηματικών μεθόδων για προσεγγιστικές λύσεις στο πρόβλημα. Κατά καιρούς προτάθηκαν ποικίλοι αλγόριθμοι με διαφορές στην απόδοση και την ταχύτητα. Κοινό χαρακτηριστικό των περισσότερων από αυτούς είναι η προσπάθεια για δημιουργία τεχνητών προβλημάτων δύο σταδίων και εφαρμογή του αλγορίθμου του Johnson στα μετασχηματισμένα δεδομένα.

Παρακάτω παρατίθενται σύντομα οι χαρακτηριστικότεροι από τους ευρηματικούς αλγορίθμους για το πρόβλημα $N/M/F, P/C_{max}$. Αλλοι αλγόριθμοι ή παραλλαγές των παρακάτω αναφέρονται στα [Lei90, CGD91, CB92, DPS92].

3.3.1 Αλγόριθμος CDS

Ο περισσότερο γνωστός και διαδεδομένος αλγόριθμος για το πρόβλημα M μηχανών προτάθηκε το 1970 από τους Campbell, Dudek και Smith [CDS70]. Ο αλγόριθμος δημιουργεί και επιλύει M-1 βοηθητικά προβλήματα δύο μηχανών και αποδέχεται την καλύτερη από τις M-1 λύσεις. Ακολουθεί η αρχική μορφή του αλγορίθμου η οποία δεν συμπεριλαμβάνει χρόνους εξάρμωσης. Να σημειωθεί ότι είναι δυνατή η επέκτασή του για χρόνους εξάρμωσης ανεξάρτητους ακολουθίας [CGD91].

Βήμα 1 Εστω P = M-1 ο συνολικός αριθμός βοηθητικών προβλημάτων δύο σταδίων προς λύση. Αρχικά k := 1.

Βήμα 2 Για κάθε εργασία i, υπολόγισε τους χρόνους επεξεργασίας της p_{i1}^k και p_{i2}^k για την πρώτη και δεύτερη μηχανή αντίστοιχα, ως εξής :

$$p_{i1}^k = \sum_{m=1}^{k} p_{im}$$

και

$$p_{i2}^k = \sum_{m=M-k+1}^{M} p_{im}$$

Βήμα 3 Λύσε το υπαριθμόν k βοηθητικό πρόβλημα δύο σταδίων με τους χρόνους επεξεργασίας του βήματος 2, χρησιμοποιώντας τον αλγόριθμο Johnson. Εστω S^k η λύση που προκύπτει και $C_{max}(S^k)$ ο συνολικός χρόνος εκτέλεσης του προγράμματος S^k στις M μηχανές. Αν k = P τότε συνέχισε στο βήμα 4, αλλιώς κάνε k := k + 1 και γύρισε στο βήμα 2.

Βήμα 4 Μεταξύ των P διαφορετικών προγραμμάτων S^k, με k=1,..,P, που παρήχθησαν από τα βοηθητικά προβλήματα δύο σταδίων, διάλεξε αυτό με το μικρότερο $C_{max}(S^k)$ ως τελική λύση.

Είναι φανερό ότι η παραπάνω διαδικασία παράγει και εξετάζει προκαθορισμένο αριθμό συνολικών προγραμμάτων P με

$$P = M - 1$$

Από πειραματικές μελέτες [CDS70, CGD91], ο αλγόριθμος CDS αποδείχθηκε εξαιρετικά αποτελεσματικός και σταθερός για διάφορα μεγέθη προβλημάτων και καθιερώθηκε ως κλασσικός για το πρόβλημα ροϊκής παραγωγής. Αναλύοντας θεωρητικά τη διαδικασία, οι Nowicki και Smutnicki [NS89] προσδιόρισαν ένα άνω φράγμα για το λόγο απόδοσης η_{CDS} του αλγορίθμου. Ο λόγος απόδοσης του αλγορίθμου A, η_A ορίζεται ως εξής:

$$\eta_A = \frac{C_{max}(A)}{C^*_{max}}$$

όπου $C_{max}(A)$ ο συνολικός χρόνος εκτέλεσης της λύσης που παράγει ο αλγόριθμος A και C^*_{max} ο συνολικός χρόνος εκτέλεσης της βέλτιστης λύσης. Ισχύει:

$$\eta_{CDS} \leq \lceil \frac{M}{2} \rceil$$

3.3.2 Αλγόριθμος Dannenbring

Ο αλγόριθμος που προτάθηκε το 1977 από τον Dannenbring [Dan77] λειτουργεί σε δύο φάσεις: κατά την πρώτη φάση λύνεται ένα βοηθητικό πρόβλημα δύο σταδίων και κατά τη δεύτερη φάση η λύση βελτιώνεται με μια επαναληπτική διαδικασία αντιμετάθεσης γειτονικών εργασιών. Ετσι, το βοηθητικό πρόβλημα δύο σταδίων προσφέρει σ' αυτήν την περίπτωση μια αρχική ακολουθία εργασιών, η οποία αναθεωρείται ριζικά στη συνέχεια. Είναι φανερό ότι η δεύτερη φάση είναι ουσιαστικότερη για τη βελτίωση της ποιότητας της λύσης, αλλά και εξαιρετικά χρονοβόρα συγχρόνως. Επειδή η λειτουργία της δεύτερης φάσης κατατάσσει τον αλγόριθμο στη γενικότερη κατηγορία

μεθόδων βελτιστοποίησης με τοπική έρευνα (local search), θα αγνοήσουμε τη διαδικασία αντιμετάθεσης γειτονικών εργασιών στην ανάλυσή μας και θα περιοριστούμε στο μετασχηματισμό του προβλήματος και την αναγωγή του στις δύο μηχανές κατά την πρώτη φάση.

Στην πρώτη φάση του αλγορίθμου δημιουργείται ένα τεχνητό πρόβλημα δύο μηχανών με χρόνους επεξεργασίας ίσους με σταθμισμένα αθροίσματα των αρχικών χρόνων επεξεργασίας. Συγκεκριμένα υπολογίζονται τα:

$p'_{i1} = \sum_{j=1}^{M} (M - j + 1)\, p_{ij}$ για i=1,2,..,N

και

$p'_{i2} = \sum_{j=1}^{M} j\, p_{ij}$ για i=1,2,..,N

και λύνεται το παραγόμενο πρόβλημα 2 μηχανών με τον αλγόριθμο του Johnson. Θεωρώντας τελική τη λύση που προκύπτει από την πρώτη φάση, ο αλγόριθμος αποδεικνύεται θεωρητικά και πειραματικά κατώτερος έναντι του CDS. Στο [NS91] υπολογίζεται το άνω φράγμα του λόγου απόδοσης του αλγορίθμου η_{Dann} με ανάλυση της χειρότερης περίπτωσης. Συγκεκριμένα:

$$\eta_{Dann} \leq \frac{M}{\sqrt{2}}$$

Πειραματικές μελέτες με τυχαία δεδομένα έδειξαν ότι και στη μέση περίπτωση η ποιότητα της λύσης της πρώτης φάσης είναι σημαντικά χαμηλότερη έναντι αυτής των λύσεων του CDS.

3.3.3 Αλγόριθμος Nawaz - Προγραμματισμός με Παρεμβολή Εργασιών

Ο M.Nawaz πρότεινε το 1983 ένα νέο ευρηματικό αλγόριθμο για τον προγραμματισμό N εργασιών σε M μηχανές με διαφορετική φιλοσοφία και λειτουργία [Naw83]. Ο αλγόριθμος δεν μετασχηματίζει το πρόβλημα σε άλλο δύο μηχανών το οποίο λύνεται με τον αλγόριθμο Johnson, όπως στους αλγορίθμους CDS και Dannenbring. Κατασκευάζει σταδιακά την τελική λύση με συστηματικό τρόπο, τοποθετώντας μια προς μια τις εργασίες στη λύση και επιλέγοντας κάθε φορά τη μερική ακολουθία με το μικρότερο χρόνο εκτέλεσης. Παράγει την τελική λύση σε ακριβώς N βήματα και σε κάθε βήμα αποφασίζει για τη θέση μιας εργασίας, παρεμβάλλοντάς την σε όλες τις δυνατές θέσεις της μερικής ακολουθίας που έχει μέχρι στιγμής προκύψει. Στην αρχή της διαδικασίας οι εργασίες τοποθετούνται σε μια λίστα που ονομάζεται ουρά προγραμματισμού και καθορίζει τη

σειρά παρεμβολής των εργασιών. Το κριτήριο αρχικής διάταξης των εργασιών στην ουρά προγραμματισμού επηρεάζει σημαντικά την απόδοση του αλγορίθμου. Ο Nawaz πρότεινε διάταξη των εργασιών στην ουρά προγραμματισμού κατά φθίνουσα τάξη του συνολικού χρόνου επεξεργασίας τους σ' όλες τις μηχανές, προκειμένου οι εργασίες με μεγάλους χρόνους επεξεργασίας να αποκτήσουν υψηλότερη προτεραιότητα και να προγραμματιστούν στην αρχή της διαδικασίας.

Στο i βήμα του αλγορίθμου, προγραμματίζουμε την i-οστή εργασία της ουράς προγραμματισμού, τοποθετώντας την διαδοχικά στις i δυνατές θέσεις της μερικής ακολουθίας μήκους i-1, η οποία έχει προκύψει από το προηγούμενο βήμα. Από τα i προγράμματα μήκους i που παράγονται, επιλέγεται αυτό με το μικρότερο συνολικό χρόνο εκτέλεσης. Η σειρά εκτέλεσης των εργασιών είναι φανερό ότι μπορεί να μεταβάλλεται σε κάθε βήμα, αλλά η σχετική σειρά εκτέλεσης δύο εργασιών δεν είναι δυνατόν να αναθεωρηθεί σε επόμενα βήματα.

Ακολουθεί ο αλγόριθμος όπως διατυπώθηκε από τον Nawaz (χωρίς χρόνους εξάρμωσης για τις μηχανές) :

Βήμα 1 Για κάθε εργασία i, με i = 1,2,..,N, υπολόγισε :

$$T_i = \sum_{j=1}^{M} p_{ij}$$

Βήμα 2 Σχημάτισε την ουρά προγραμματισμού με φθίνουσα διάταξη των εργασιών κατά T_i.

Βήμα 3 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία S_{opt}^1 με μήκος 1.
Θέσε k = 2.

Βήμα 4 Πάρε την k εργασία της ουράς προγραμματισμού και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας S_{opt}^{k-1} μήκους k-1. Από τα k παραγόμενα προγράμματα S_i^k με i = 1,..,k (στη μερική ακολουθία S_i^k με μήκος k, η τρέχουσα εργασία βρίσκεται στην i θέση), διάλεξε αυτό με το μικρότερο C_{max} και ονόμασέ το S_{opt}^k.

Βήμα 5 Αν $\kappa < N$, θέσε k = k + 1 και εκτέλεσε το βήμα 4. Αλλιώς, πήγαινε στο βήμα 6.

Βήμα 6 Δώσε ως τελική λύση, το S_{opt}^N.

Ο αλγόριθμος του Nawaz υπολογίζει την τελική λύση σε N στάδια, ενώ ο αριθμός μερικών προγραμμάτων που παράγει κατά το στάδιο i είναι είναι ακριβώς i. Αρα, ο συνολικός αριθμός προγραμμάτων P που θεωρεί ο αλγόριθμος είναι :

$$P = 2 + .. + N = \frac{N(N+1)}{2} - 1$$

εκ των οποίων τα N είναι πλήρεις και τα υπόλοιπα μερικές ακολουθίες εργασιών.

Η απόδοση του αλγορίθμου παρεμβολής εργασιών εξαρτάται σε μεγάλο βαθμό από το κριτήριο διάταξης των εργασιών στην ουρά προγραμματισμού. Διαφορετικές αρχικές διατάξεις συνεπάγονται και διαφορετικές λύσεις με ριζικές αλλαγές στη σχετική σειρά εκτέλεσης των εργασιών. Η υπόθεση του Nawaz για το κριτήριο του συνολικού χρόνου επεξεργασίας, δίνει υψηλότερη προτεραιότητα στις μεγάλες εργασίας, αλλά δεν εγγυάται θεωρητικά την βέλτιστη απόδοση του αλγορίθμου.

Ακόμα και στην περίπτωση καλής αρχικής διάταξης εργασιών στην ουρά προγραμματισμού, η ποιότητα των λύσεων μπορεί να είναι χαμηλή. Επειδή ο αλγόριθμος προχωρεί κατά "μυωπικό" τρόπο, επιλέγοντας σε κάθε βήμα το μερικό πρόγραμμα με το μικρότερο C_{max}, ενδέχεται οι αποφάσεις για τη σχετική σειρά εκτέλεσης των εργασιών να μην οδηγούν τελικά σε τελικό πρόγραμμα με ελαχιστοποιημένο C_{max}. Τρόπος αναθεώρησης αυτών των αποφάσεων δεν υπάρχει και ο κίνδυνος για κακές λύσεις παρουσιάζεται έντονος, ειδικά σε περιπτώσεις μεγάλης ανομοιομορφίας των χρόνων επεξεργασίας των εργασιών από μηχανή σε μηχανή. Παρόλ' αυτά ο αλγόριθμος παράγει καλύτερες λύσεις από τους προηγούμενος και παρουσιάζει εξαιρετικά σταθερό λόγο απόδοσης για μικρά και μεγάλα μεγέθη προβλημάτων [Naw83, Lei90].

3.3.4 Αλγόριθμος σταδιακής κατασκευής τελικού προγράμματος με χρήση κάτω φραγμάτων για το χρόνο εκτέλεσης

Οπως αναφέρθηκε στην παράγραφο 2.2, μπορούμε να υπολογίσουμε πάντα ένα κάτω φράγμα για το συνολικό χρόνο εκτέλεσης όλων των προγραμμάτων που ξεκινούν με την εκτέλεση μιας συγκεκριμένης μερικής ακολουθίας εργασιών. Αν και δεν έχουν βρεθεί αποδοτικοί μηχανισμοί υπολογισμού τέτοιων φραγμάτων, μπορούμε να συγκρίνουμε την ποιότητα μερικών προγραμμάτων με βάση τα κάτω φράγματα για το προβλεπόμενο C_{max}. Ετσι, μπορούμε να κατασκευάσουμε σταδιακά την τελική λύση, προσπαθώντας να ελαχιστοποιούμε συνεχώς τα κάτω φράγματα του χρόνου εκτέλεσης για ολόκληρο το πρόγραμμα. Σε κάθε βήμα, λοιπόν, τοποθετούμε στο τέλος της τρέχουσας μερικής

ακολουθίας εργασιών την απρογραμμάτιστη εργασία η οποία ελαχιστοποιεί το κάτω φράγμα για το χρόνο εκτέλεσης όλων των προγραμμάτων που αρχίζουν με εκτέλεση της μερικής ακολουθίας που προκύπτει.

Ακολουθεί η περιγραφή του αλγορίθμου που υλοποεί τη γενική αυτή ιδέα, όπως παρουσιάζεται στο [CGD91] :

Βήμα 1 Εστω π = {1,2,...,N} το σύνολο των απρογραμμάτιστων εργασιών και σ = { } η μερική ακολουθία που αντιπροσωπεύει το τρέχον πρόγραμμα.

Βήμα 2 Βρες την εργασία α ϵπ, τέτοια ώστε LB(σα) να είναι ελάχιστο, όπου LB(μ) είναι το κάτω φράγμα για το χρόνο εκτέλεσης όλων των προγραμμάτων που αρχίζουν με τη μερική ακολουθία μ.

Βήμα 3 Πρόσθεσε την α στο τέλος της μερικής ακολουθίας σ και αφαίρεσέ την από το π.

Βήμα 4 Αν π = { }, τότε τέλος. Αλλιώς, εκτέλεσε το βήμα 2.

Ο αλγόριθμος αυτός παράγει την τελική λύση του σε N βήματα ακριβώς. Στο βήμα i εξετάζουμε N-i+1 μερικές ακολουθίες και υπολογίζουμε για κάθε μια το κάτω φράγμα του χρόνου εκτέλεσης όλων των προγραμμάτων που ξεκινούν με εκτέλεσή της. Αρα, συνολικά θεωρούμε :

$$P = N + (N-1) + .. + 2 = \frac{N(N+1)}{2} - 1$$

προγράμματα και για το κάθε ένα καλούμε μια χρονοβόρα διαδικασία υπολογισμού κάτω φραγμάτων. Ετσι, ο απαιτούμενος υπολογιστικός χρόνος αυξάνεται σημαντικά.

Ο αλγόριθμος είναι κατασκευαστικός, όπως και αυτός του Nawaz, λειτουργεί όμως με διαφορετικό σκεπτικό. Στον αλγόριθμο Nawaz φροντίζουμε η μερική ακολουθία που μέχρι στιγμής έχουμε κατασκευάσει να έχει μικρό χρόνο εκτέλεσης, ενώ στο νέο αλγόριθμο φροντίζουμε η τρέχουσα μερική ακολουθία να εξασφαλίζει ένα καλό κάτω φράγμα, προβλέποντας κατά κάποιο τρόπο την αποδοτική κατεύθυνση έρευνας. Στη γενική περίπτωση, ο αλγόριθμος δεν εξασφαλίζει καλή ποιότητα λύσεων μια και δεν υπάρχουν μηχανισμοί υπολογισμού "σφιχτών" κάτω φραγμάτων, όπως αναφέραμε στην παράγραφο 3.2.

3.3.5 Συγκριτική Αξιολόγηση των Αλγορίθμων

Ο αλγόριθμος CDS συγκαταλέγεται ανάμεσα στους πρώτους ευρηματικούς αλγορίθμους με καλή ποιότητα λύσεων και θεωρείται κλασσικός για προβλήματα N εργασιών σε M μηχανές. Η απόδοσή του παρουσιάζει ικανοποιητική σταθερότητα ακόμα και για μεγάλα μεγέθη προβλημάτων. Στη βιβλιογραφία αναφέρεται συχνά και αποτελεί μέτρο σύγκρισης αποδόσεων διαφορετικών μεθόδων [CGD91, CB92, Lei90]. Επιπλέον, αποτελεί τη βάση για την ανάπτυξη άλλων ευρηματικών μεθόδων και χρησιμοποιείται συχνά για την παραγωγή μιας αρχικής λύσης η οποία βελτιώνεται στη συνέχεια με διάφορες μεθόδους [Lei90]. Το βασικότερο πλεονέκτημά του έγκειται στην ταχύτητα προσδιορισμού των λύσεων.

Οι λύσεις που παράγει ο αλγόριθμος του Dannenbring παρουσιάζονται βελτιωμένες σημαντικά από αυτές του αλγορίθμου CDS. Ομως, η χρονοβόρα διαδικασία τοπικής έρευνας που αποτελεί το βασικό κορμό του, τον κατατάσσει σε διαφορετική κατηγορία προσεγγιστικών μεθόδων. Θεωρώντας την πρώτη φάση του αλγορίθμου ως ανεξάρτητη διαδικασία, η απόδοσή της εμφανίζεται θεωρητικά και πειραματικά κατώτερη του CDS.

Ο αλγόριθμος σταδιακής κατασκευής του τελικού προγράμματος παράγει σε προκαθορισμένο μικρό αριθμό επαναλήψεων την τελική λύση, χωρίς εγγυήσεις για την ποιότητά της. Πειραματικές μελέτες απέδειξαν ότι η απόδοσή του είναι αισθητά χειρότερη από τους δύο προηγούμενους αλγορίθμους, αλλά και από άλλους με ίδιας τάξης μεγέθους χρόνους εκτέλεσης [CGD91]. Αυτό πρέπει να αποδοθεί στην εξαιρετικά μεγάλη πολυπλοκότητα του προβλήματος και την εξάρτησή του από τα δεδομένα. Ετσι, οι βασισμένες σε κάτω φράγματα εκτιμήσεις για την ποιότητα των τελικών λύσεων που κάνει ο αλγόριθμος στα πρώτα βήματά του, δεν μπορούν να είναι ακριβείς και να λάβουν υπόψη τους νεκρούς χρόνους που θα προκύψουν από τις απρογραμμάτιστες εργασίες.

Από τους αλγορίθμους που περιγράφηκαν, την καλύτερη και σταθερότερη απόδοση παρουσιάζει αυτός του Nawaz. Γενικά, δοκιμές με τυχαία προβλήματα τοποθετούν τον αλγόριθμο αυτό στην κορυφή του καταλόγου των ευρηματικών μεθόδων για το πρόβλημα αλυσίδας παραγωγής [Lei90, Naw83, WH89]. Το βασικό μειονέκτημα της μεθόδου είναι η αύξηση του απαιτούμενου υπολογιστικού χρόνου για πολύ μεγάλα μεγέθη προβλημάτων, μια και ο αριθμός των προγραμμάτων που εξετάζει είναι ανάλογος του τετραγώνου του αριθμού των εργασιών. Αν και η διαφορά στους σχετικούς (προς τους άλλους αλγορίθμους) χρόνους τρεξίματος της διαδικασίας είναι αισθητή, οι απόλυτοι χρόνοι παραμένουν σε πολύ χαμηλό επίπεδο και χαρακτηρίζονται μικροί για διαδικασία

βελτιστοποίησης. Ετσι, η πρακτική αξία της δεν μειώνεται ακόμα και για μεγάλο αριθμό εργασιών ($N \geq 30$).

4 Εφικτή παρεμβολή εργασιών με προθεσμίες

Το πρόβλημα $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$ που μελετάμε στην παρούσα εργασία, όπως τέθηκε στην παράγραφο 1.3, είναι ο προγραμματισμός εργασιών με προθεσμίες παράδοσης σε αλυσίδα μηχανών με πρωτεύον κριτήριο την ελαχιστοποίηση της μέγιστης καθυστέρησης και δευτερεύον την ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης. Μεταξύ δηλαδή του συνόλου των εφικτών ως προς τις προθεσμίες προγραμμάτων, αναζητούμε αυτό με το μικρότερο C_{max}. Σε περίπτωση αδυναμίας εντοπισμού εφικτών προγραμμάτων, αναζητούμε λύσεις με ελαχιστοποιημένες καθυστερήσεις εργασιών και καλούς, κατά το δυνατόν, χρόνους εκτέλεσης. Το υπό συζήτηση πρόβλημα αναφέρεται σπάνια στη σχετική με γραμμές ροϊκής παραγωγής βιβλιογραφία και δεν έχουν αναπτυχθεί αλγόριθμοι για την αντιμετώπισή του, ούτε διαδικασίες αναγωγής του σε άλλο επιλύσιμο πρόβλημα προγραμματισμού εργασιών. Θα παρουσιάσουμε στη συνέχεια ένα νέο ευρηματικό αλγόριθμο, τον αλγόριθμο εφικτής παρεμβολής, ο οποίος στηρίζεται στη μέθοδο παρεμβολής εργασιών της παραγράφου 3.3.3 και αναζητεί εφικτές λύσεις για το πρόβλημα ροϊκής παραγωγής με προθεσμίες με ελαχιστοποιημένο χρόνο εκτέλεσης.

Προκειμένου να συμπεριλάβουμε τις προθεσμίες των εργασιών στη διαδικασία προγραμματισμού, τροποποιήσαμε κατάλληλα τη διαδικασία παρεμβολής εργασιών του αλγορίθμου του Nawaz. Ο αλγόριθμος αυτός παρουσιάζει ορισμένα πλεονεκτήματα έναντι άλλων, τα οποία τον καθιστούν κατάλληλο ως βασικό αλγοριθμικό σχήμα για το προκείμενο πρόβλημα. Συγκεκριμένα:

- Ο αλγόριθμος του Nawaz παράγει τις καλύτερες λύσεις μεταξύ όλων των μεθόδων της κατηγορίας του για το κριτήριο της ελαχιστοποίησης του συνολικού χρόνου εκτέλεσης. Αυτό γίνεται βέβαια με σημαντικό κόστος σε υπολογιστικό χρόνο, μια και το σύνολο των διατάξεων που εξετάζει ο αλγόριθμος είναι αρκετά μεγαλύτερο.
- Η διαδικασία προγραμματισμού εργασιών με παρεμβολή κατασκευάζει σταδιακά την τελική λύση και εξετάζει σε κάθε βήμα το συνολικό χρόνο εκτέλεσης της τρέχουσας μερικής ακολουθίας εργασιών. Αρα, είναι δυνατό να ελέγχεται το εφικτό των μερικών διατάξεων σε κάθε στάδιο και η διαδικασία να απορρίπτει κατευθύνσεις που οδηγούν σε ανέφικτα ως προς τις προθεσμίες προγράμματα. Κατά συνέπεια δεν είναι αναγκαίος και παρακάμπτεται ο μηχανισμός αναγωγής προθεσμιών σε κάθε ενδιάμεσο στάδιο της παραγωγής, όπως σκιαγραφήθηκε στην παράγραφο 2.6.

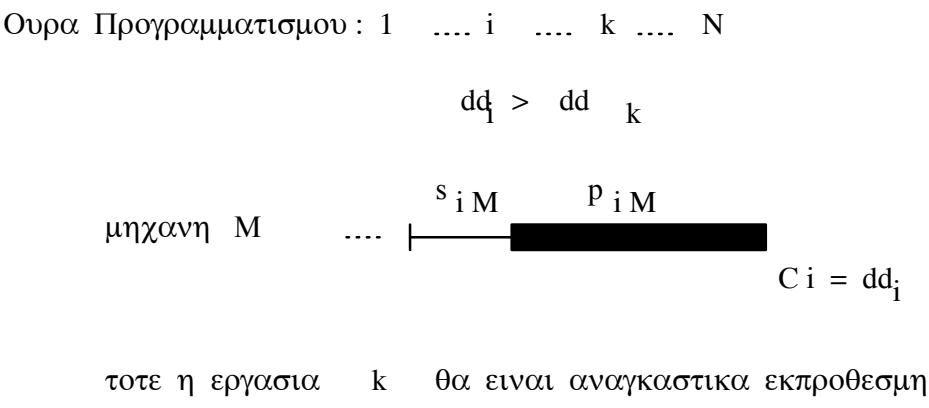
4.1 Τροποποίηση της διαδικασίας παρεμβολής - Εφικτή παρεμβολή

Ο αλγόριθμος που αναπτύξαμε για προγραμματισμό εργασιών με προθεσμίες εκμεταλλεύεται τη σταδιακή κατασκευή του τελικού προγράμματος για να εξετάζει σε κάθε βήμα μόνο τα εφικτά προγράμματα που είναι δυνατό να παραχθούν από την τρέχουσα μερική διάταξη εργασιών. Η διαδικασία παρεμβολής εργασιών στο i βήμα της τοποθετεί την i εργασία της ουράς προγραμματισμού σε όλες τις δυνατές θέσεις της μερικής ακολουθίας μήκους i-1 και από τις i παραγόμενες μερικές ακολουθίες μήκους i επιλέγει αυτήν με το μικρότερο χρόνο εκτέλεσης. Η τροποποιημένη διαδικασία παρεμβάλλει την εργασία i στις θέσεις της μερικής ακολουθίας μήκους i-1 για τις οποίες η νέα μερική ακολουθία που προκύπτει είναι εφικτή ως προς τις προθεσμίες όλων των μέχρι στιγμής προγραμματισμένων εργασιών (της i συμπεριλαμβανομένης). Από τις εφικτές μερικές διατάξεις μήκους i επιλέγουμε αυτήν με το μικρότερο χρόνο εκτέλεσης.

Κρίσιμο στοιχείο για το νέο αλγόριθμο είναι το κριτήριο αρχικής διάταξης των εργασιών στην ουρά προγραμματισμού. Ενώ στον αλγόριθμο του Nawaz οι εργασίες διατάσσονται κατά φθίνουσα τάξη του συνολικού χρόνου επεξεργασίας τους, τώρα η διάταξη μεταβάλλεται. Τοποθετούμε τις εργασίες στην ουρά προγραμματισμού κατά αύξουσα σειρά προθεσμιών. Ετσι, εργασίες με μικρές προθεσμίες παράδοσης προγραμματίζονται πρώτες στη διαδικασία. Με αυτόν τον τρόπο προσδίδουμε μεγαλύτερη προτεραιότητα στις εργασίες με σφικτές χρονικές προθεσμίες, για την θέση των οποίων στο τελικό πρόγραμμα περιμένουμε να υπάρχουν πολύ μικρότερα περιθώρια. Με τη συγκεκριμένη λοιπόν αρχική ουρά προγραμματισμού ο αλγόριθμος είναι πολύ πιθανότερο να εντοπίσει μια εφικτή διάταξη αν αυτή υπάρχει, μια και φροντίζει πρώτα να παρεμβάλει τις επείγουσες χρονικά εργασίες και σε επόμενα βήματα να εξετάσει τις εφικτές θέσεις για τις εργασίες που παρουσιάζουν χαλαρότητα. Προκειμένου να επιβεβαιώσουμε τις παραπάνω υποθέσεις για την απόδοση του κριτηρίου αρχικής διάταξης, υλοποιήσαμε δύο παραλλαγές του αλγορίθμου εφικτής παρεμβολής εργασιών. Η πρώτη εκδοχή πραγματοποιούσε εφικτή παρεμβολή εργασιών με διάταξη στην ουρά προγραμματισμού κατά συνολικό χρόνο επεξεργασίας (κριτήριο Nawaz), ενώ η δεύτερη με διάταξη κατά προθεσμίες.

Τα αποτελέσματα των δοκιμών σε ορισμένα τεχνητά προβλήματα ροϊκής παραγωγής απέδειξαν ότι ο αλγόριθμος στην πρώτη εκδοχή του αδυνατούσε να εντοπίσει εφικτά προγράμματα, ακόμα και σε περιπτώσεις χαλαρών προθεσμιών. Αυτό πρέπει να αποδοθεί στο ότι μια εργασία με προθεσμία ενδέχεται να προγραμματιστεί όσο αργότερα γίνεται από τη διαδικασία, αν αυτό συνεπάγεται καλύτερο χρόνο εκτέλεσης, ενώ σε επόμενα

βήματα μια εργασία με μικρότερη ημερομηνία παράδοσης δεν έχει πλέον κανένα περιθώριο για παρεμβολή σε θέση στην οποία δεν χάνει την προθεσμία της. Η περίπτωση φαίνεται στο σχήμα 4, όπου η εργασία k βρίσκεται μετά την εργασία i στην αρχική ουρά προγραμματισμού και $dd_k < dd_i$. Μετά την τοποθέτηση της i σε θέση χωρίς καθόλου χρονική χαλαρότητα ($C_i = dd_i$), η k δεν μπορεί να παρεμβληθεί ώστε να προκύπτει εφικτό πρόγραμμα. Εισαγωγή της k πριν την i συνεπάγεται καθυστέρηση της i, ενώ τοποθέτησή της μετά την i, συνεπάγεται προφανώς καθυστέρηση της ίδιας.



Σχήμα 4: Απώλεια εφικτής λύσης στην περίπτωση που η εργασία i περατώνεται ακριβώς πριν την προθεσμία της.

Το δεύτερο σημαντικό στοιχείο που προέκυψε από τη σύγκριση των δύο παραλλαγών του αλγορίθμου ήταν η αμελητέα διαφορά των συνολικών χρόνων εκτέλεσης των λύσεων που παρήγαγαν. Το κριτήριο αρχικής διάταξης του Nawaz δεν απέδιδε συνδυασμένο με τη διαδικασία εφικτής παρεμβολής, αφού οι δυνατές θέσεις των εργασιών σε κάθε βήμα καθορίζονταν από το εφικτό των μερικών διατάξεων που προέκυπταν. Αρα, η τροποποιημένη παρεμβολή εργασιών με προθεσμίες λειτουργούσε αποδοτικότερα με αρχική διάταξη εργασιών κατά ημερομηνίες παράδοσης τόσο ως προς τις καθυστερήσεις των εργασιών, όσο και ως προς το συνολικό χρόνο εκτέλεσης.

Σε περίπτωση αδυναμίας εντοπισμού εφικτής μερικής διάταξης πρέπει να ορίσουμε τη συμπεριφορά του αλγορίθμου. Επειδή ιεραρχούμε τα κριτήρια βελτίστου έτσι ώστε το T_{max} να προέχει του C_{max}, οφείλουμε να ελαχιστοποιήσουμε τις καθυστερήσεις των εργασιών και να αγνοήσουμε το κριτήριο του καλού συνολικού χρόνου εκτέλεσης τους. Από τα προβλήματα προγραμματισμού εργασιών σε μια μηχανή είναι γνωστό ότι με κριτήριο την ελαχιστοποίηση της μέγιστης καθυστέρησης το βέλτιστο πρόγραμμα σχηματίζεται από διάταξη των εργασιών κατά αύξουσα τάξη προθεσμιών (διάταξη

Earliest Due Date) [Jac55]. Το αποτέλεσμα αυτό μπορεί να επεκταθεί και για γραμμές ροϊκής παραγωγής σε περιπτώσεις ομαλής ροής εργασίας στο σύστημα. Πράγματι, αν δεν υπάρχουν μεγάλα διαστήματα νεκρών χρόνων στις μηχανές είναι προφανές ότι η μέγιστη καθυστέρηση ελαχιστοποιείται όταν οι εργασίες διέλθουν από τα σύστημα με τη σειρά που καθορίζουν οι προθεσμίες τους. Αυτό δεν ισχύει σε περίπτωση που η διάταξη των εργασιών κατά αύξουσες προθεσμίες συνεπάγεται εμφάνιση σημαντικών νεκρών χρόνων και, κατά συνέπεια, καθυστέρηση της ολοκλήρωσης ορισμένων εργασιών ακόμα περισσότερο. Επειδή, όμως, η περίπτωση αυτή αποτελεί εξαίρεση, θα υιοθετήσουμε τη διάταξη εργασιών κατά αύξουσες προθεσμίες όταν έχουμε ήδη οδηγηθεί σε απώλεια του εφικτού των λύσεων, καλύπτωντας έτσι την περισσότερο συνηθισμένη περίπτωση.

Ετσι, με σκοπό να ελαχιστοποιήσουμε κατά το δυνατό τις καθυστερήσεις που προκύπτουν σε περίπτωση αδυναμίας εντοπισμού εφικτής θέσης για την τρέχουσα εργασία, την τοποθετούμε στο τέλος του μερικού προγράμματος. Το ίδιο κάνουμε και για τις εργασίες που παρεμβάλλονται σε επόμενα βήματα της διαδικασίας με σκοπό να μην μεγαλώσουμε την καθυστέρηση εργασιών που έχουν προθεσμία μικρότερη και, άρα, έχουν προγραμματιστεί σε προηγούμενα βήματα. Το τελικό αποτέλεσμα της πολιτικής αυτής είναι ότι από τη στιγμή που χάνουμε το εφικτό των λύσεων και μετά, διατάσσουμε τις εργασίες με προθεσμίες που απομένουν προς προγραμματισμό κατά αύξουσα τάξη προθεσμιών.

Αλγόριθμος εφικτής παρεμβολής εργασιών με προθεσμίες

Βήμα 1 Σχημάτισε την ουρά προγραμματισμού με διάταξη των εργασιών κατά αύξουσα σειρά προθεσμιών.

Βήμα 2 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία A^1_{opt} με μήκος 1. Αν η εργασία περατώνεται πριν την προθεσμία της, τότε FEASIBLE_FLAG = TRUE, αλλιώς FEASIBLE_FLAG = FALSE.
Θέσε k = 2.

Βήμα 3 Αν FEASIBLE_FLAG = TRUE, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας A^{k-1}_{opt} μήκους k-1. Από τα k παραγόμενα προγράμματα A^k_i με i = 1,..,k (στη μερική ακολουθία A^k_i με μήκος k, η εργασία J_k βρίσκεται στην i θέση), διάλεξε το εφικτό ως προς τις προθεσμίες όλων των εργασιών με το μικρότερο C_{max} και ονόμασέ το

A^k_{opt}. Αν δεν υπάρχει εφικτό πρόγραμμα, θέσε FEASIBLE_FLAG = FALSE και

$$A^k_{opt} = A^{k-1}_{opt}\ J_k$$

Αν FEASIBLE_FLAG = FALSE, θέσε

$$A^k_{opt} = A^{k-1}_{opt}\ J_k$$

Βήμα 4 Αν $k <$ N, θέσε k = k + 1 και εκτέλεσε το βήμα 3. Αλλιώς, θέσε $A_{opt} = A^k_{opt}$ και πήγαινε στο βήμα 5.

Βήμα 5 Δώσε ως τελική λύση το πρόγραμμα A_{opt}.

Η παραπάνω διαδικασία αναζητεί προγράμματα με μικρό χρόνο εκτέλεσης μεταξύ του συνόλου των εφικτών προγραμμάτων ή προγράμματα με μικρές κατά το δυνατό καθυστερήσεις. Δεν εγγυάται την εύρεση εφικτών λύσεων αν αυτές υπάρχουν, ούτε, πολύ περισσότερο, βέλτιστων λύσεων για το $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$ με πρωτεύον κριτήριο την ελαχιστοποίηση της μέγιστης καθυστέρησης. Οι λόγοι για την απώλεια του εφικτού και βελτίστου των λύσεων αναλύονται στο κεφάλαιο 5. Πραγματοποιεί δικριτηριακό προγραμματισμό ως προς την ελαχιστοποίηση της μέγιστης καθυστέρησης και του συνολικού χρόνου εκτέλεσης, δίνοντας απόλυτη προτεραιότητα στο πρώτο. Βασικό κριτήριο επιλογής των λύσεων είναι η μέγιστη καθυστέρηση που παρουσιάζουν, αλλά εξετάζονται κατά το δυνατόν προγράμματα με καλό συνολικό χρόνο εκτέλεσης.

Για τον αριθμό των μερικών διατάξεων που εξετάζει ο αλγόριθμος εφικτής παρεμβολής εργασιών, P, ισχύει:

$$P = \frac{N(N+1)}{2}$$

4.2 Μικτός προγραμματισμός εργασιών με ημερομηνίες παράδοσης ή χωρίς

Στο πλαίσιο της παρούσας εργασίας υποθέτουμε ότι υπάρχουν δύο κατηγορίες εργασιών: **εργασίες με προθεσμίες** (τις ονομάζουμε **ΕΜΠ**) και **εργασίες χωρίς προθεσμίες** (τις ονομάζουμε **ΕΧΠ**). Η ανάθεση προθεσμιών σε ορισμένες εργασίες γίνεται από ανώτερο επίπεδο προγραμματισμού της παραγωγής και επιβάλλει μεγαλύτερη προτεραιότητα εκτέλεσης έναντι των εργασιών για την περάτωση των οποίων δεν υπάρχει χρονικός περιορισμός. Προκειμένου να τηρηθεί η προτεραιότητα των ΕΜΠ θα παρουσιάσουμε ένα ολοκληρωμένο αλγόριθμο, ο οποίος προγραμματίζει με διαφορετικό τρόπο τις δύο

κατηγορίες εργασιών. Για τις ΕΜΠ χρησιμοποιεί τη διαδικασία εφικτής παρεμβολής εργασιών όπως περιγράφηκε, με αρχική διάταξη εργασιών στην ουρά προγραμματισμού κατά ημερομηνίες παράδοσης. Για τις ΕΧΠ χρησιμοποιεί τη διαδικασία παρεμβολής του αλγορίθμου Nawaz με αρχική διάταξη εργασιών στην ουρά προγραμματισμού κατά φθίνουσα τάξη του συνολικού χρόνου επεξεργασίας. Το παραπάνω κριτήριο διάταξης των ΕΧΠ υιοθετείται προκειμένου οι εργασίες με μεγάλο χρόνο επεξεργασίας να αποκτήσουν μεγαλύτερη προτεραιότητα στη διαδικασία προγραμματισμού. Ετσι, ο αλγόριθμος που παρουσιάζεται αναλυτικά παρακάτω, λειτουργεί σε δύο φάσεις, με διαφορετική διαδικασία παρεμβολής και αρχική ουρά προγραμματισμού σε κάθε φάση.

Αλγόριθμος παρεμβολής ΕΜΠ και ΕΧΠ σε χωριστές μερικές διατάξεις

Φάση 1: Εφικτή παρεμβολή των εργασιών με προθεσμία

Βήμα 1 Διάταξε στην ουρά προγραμματισμού τις εργασίες με προθεσμία κατά αύξουσα τάξη προθεσμιών. Εστω D ο αριθμός τους.

Βήμα 2 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία A^1_{opt} με μήκος 1. Αν η εργασία περατώνεται πριν την προθεσμία της, τότε FEASIBLE_FLAG = TRUE, αλλιώς FEASIBLE_FLAG = FALSE. Θέσε k = 2.

Βήμα 3 Αν FEASIBLE_FLAG = TRUE, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας A^{k-1}_{opt} μήκους k-1. Από τα k παραγόμενα προγράμματα A^k_i με i = 1,..,k (στη μερική ακολουθία A^k_i με μήκος k, η εργασία J_k βρίσκεται στην i θέση), διάλεξε το εφικτό ως προς τις προθεσμίες όλων των εργασιών με το μικρότερο C_{max} και ονόμασέ το A^k_{opt}. Αν δεν υπάρχει εφικτό πρόγραμμα, θέσε FEASIBLE_FLAG = FALSE και

$$A^k_{opt} = A^{k-1}_{opt}\ J_k$$

Αν FEASIBLE_FLAG = FALSE, θέσε

$$A^k_{opt} = A^{k-1}_{opt}\ J_k$$

Βήμα 4 Αν $k < D$, θέσε k = k + 1 και εκτέλεσε το βήμα 3. Αλλιώς, θέσε $A_{opt} = A^k_{opt}$ και πήγαινε στο βήμα 5.

Φάση 2: Παρεμβολή των εργασιών χωρίς προθεσμία

Βήμα 5 Για κάθε εργασία i χωρίς προθεσμία, υπολόγισε :

$$T_i = \sum_{j=1}^{M} p_{ij}$$

Βήμα 6 Διάταξε στην ουρά προγραμματισμού τις εργασίες χωρίς προθεσμία κατά φθίνουσα τάξη των T_i. Εστω F ο αριθμός τους.

Βήμα 7 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία B^1_{opt} με μήκος 1.
Θέσε k = 2.

Βήμα 8 Πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας B^{k-1}_{opt} μήκους k-1. Από τα k παραγόμενα προγράμματα B^k_i με i = 1,..,k (στη μερική ακολουθία B^k_i με μήκος k, η εργασία J_k βρίσκεται στην i θέση), διάλεξε αυτό για το οποίο η διάταξη $A_{opt}\, B^k_i$ έχει το μικρότερο C_{max} και ονόμασέ το B^k_{opt}.

Βήμα 9 Αν $k < F$, θέσε k = k + 1 και εκτέλεσε το βήμα 8. Αλλιώς, θέσε $B_{opt} = B^k_{opt}$ και πήγαινε στο βήμα 10.

Βήμα 10 Δώσε ως τελική λύση το πρόγραμμα $A_{opt}\, B_{opt}$.

Για τον αριθμό P των μερικών διατάξεων που εξετάζει ο αλγόριθμος προγραμματισμού ΕΜΠ και ΕΧΠ σε χωριστά προγράμματα, ισχύει:

$$P = (1 + 2 + .. + D) + (1 + 2 + .. + F) = \frac{D(D+1)}{2} + \frac{F(F+1)}{2}$$

όπου D και F ο αριθμός των εργασιών με προθεσμίες (ΕΜΠ) και των εργασιών χωρίς προθεσμίες (ΕΧΠ) αντίστοιχα. Επειδή $N = D + F$ το σύνολο των προγραμμάτων που παράγει ο αλγόριθμος είναι μικρότερο από το αντίστοιχο του αλγορίθμου παρεμβολής του Nawaz. Για κάθε μερική διάταξη ο αλγόριθμος υπολογίζει το C_{max}, ενώ για τις μερικές διατάξεις της φάσης 1, υπολογίζει και τους χρόνους περάτωσης των εργασιών στην τελευταία μηχανή από τη σχέση 4 του 2.1.

4.3 Βελτίωση του χρόνου εκτέλεσης

Ο αλγόριθμος προγραμματισμού ΕΜΠ και ΕΧΠ σε χωριστά προγράμματα παράγει ένα τελικό πρόγραμμα που αποτελείται από δύο ουσιαστικά ανεξάρτητα υποπρογράμματα:

το υποπρόγραμμα A_{opt} των ΕΜΠ και το υποπρόγραμμα B_{opt} των ΕΧΠ. Η φάση 2 προγραμματίζει τις ΕΧΠ πάντα μετά το τέλος της μερικής διάταξης A_{opt} που σχηματίστηκε κατά την πρώτη φάση. Βέβαια, η παρεμβολή των ΕΧΠ υπολογίζει και λαμβάνει υπόψη της το C_{max} του προγράμματος που προκύπτει από τη συνένωση της A_{opt} με την τρέχουσα μερική ακολουθία εργασιών B^k_{opt}.

Σε περίπτωση, όμως, που οι προθεσμίες των ΕΜΠ είναι αρκετά χαλαρές ώστε να υπάρχουν περιθώρια για αργότερη εκτέλεσή τους στο τελικό πρόγραμμα από αυτήν που ορίζει η διάταξη A_{opt}, ενδέχεται να υπάρχουν εφικτές λύσεις με μικρότερο C_{max} στις οποίες ΕΧΠ εκτελούνται πριν από κάποιες ΕΜΠ. Αρα, μια λύση στην οποία ΕΜΠ και ΕΧΠ αναμιγνύονται στο τελικό πρόγραμμα μπορεί να είναι σημαντικά βελτιωμένη ως προς το συνολικό χρόνο εκτέλεσης. Το παράδειγμα του σχήματος 5 παρουσιάζει μια τέτοια περίπτωση. Η εργασία 2 στο τελικό πρόγραμμα 1-2-3 του αλγορίθμου της προηγούμενης παραγράφου, περατώνεται τη χρονική στιγμή 15, ενώ η προθεσμία της είναι 20. Ο χρόνος εκτέλεσης για το πρόγραμμα είναι 22. Παρατηρούμε ότι το πρόγραμμα 1-3-2 έχει χρόνο εκτέλεσης 18 , ενώ είναι και εφικτό. Αρα, η χρονική χαλαρότητα της εργασίας 2 επιτρέπει την εκτέλεση της εργασίας 3 πριν από αυτήν, με τελικό αποτέλεσμα τη βελτίωση του C_{max}. Κατά συνέπεια, οφείλουμε να εξετάζουμε τη δυνατότητα παρεμβολής ΕΧΠ μεταξύ των ήδη προγραμματισμένων ΕΜΠ.

Σχήμα 5: Προγραμματισμός ΕΜΠ και ΕΧΠ σε εννιαίο πρόγραμμα: η εργασία 3 μπορεί να εκτελεσθεί μεταξύ των 1 και 2.

Παρακάτω παρουσιάζουμε ένα βελτιωμένο αλγόριθμο παρεμβολής ο οποίος διαφέρει από τον αλγόριθμο προγραμματισμού ΕΜΠ και ΕΧΠ σε χωριστά προγράμματα μόνο στη φάση 2. Οι εργασίες χωρίς προθεσμίες παρεμβάλλονται τώρα σ' ολόκληρη τη μερική διάταξη των εργασιών που έχουν μέχρι στιγμής προγραμματιστεί και από τα εφικτά προγράμματα που παράγονται, αποθηκεύεται σε κάθε βήμα αυτό με το μικρότερο χρόνο εκτέλεσης. Αν η φάση 1 δεν εντοπίσει εφικτό υποπρόγραμμα A_{opt}, τότε η φάση 2 δεν εξετάζει ποτέ την εισαγωγή ΕΧΠ μέσα στο υποπρόγραμμα A_{opt}. Αυτό γίνεται με προφανή στόχο τη διατήρηση σε χαμηλά επίπεδα των καθυστερήσεων που έχουν προκύψει κατά τη φάση 1. Αν επιτρέπαμε εκτέλεση μιας ΕΧΠ πριν από κάποια ΕΜΠ που παρουσιάζει ήδη χρονική καθυστέρηση, τότε η καθυστέρηση αυτή θα μεγάλωνε και το ίδιο είναι πιθανό και για ΕΜΠ που εκτελούνται μετά από αυτήν. Κατά συνέπεια λοιπόν, το κριτήριο της ελαχιστοποίησης της μέγιστης καθυστέρησης διατηρεί μεγαλύτερη προτεραιότητα έναντι εκείνου της ελαχιστοποίησης του συνολικού χρόνου εκτέλεσης. Ο αλγόριθμος σχεδιάστηκε ώστε να προτιμά λύσεις με μικρές καθυστερήσεις και μεγαλύτερους χρόνους εκτέλεσης γιατί οι συνθήκες παραγωγής σε εργοστασιακά περιβάλλοντα επιβάλουν συνήθως την άμεση εκτέλεση επειγουσών χρονικά εργασιών, έστω και αν αυτό αποβαίνει εις βάρος της συνολικής παραγωγής και του ρυθμού λειτουργίας των μηχανών.

Αλγόριθμος παρεμβολής ΕΜΠ και ΕΧΠ σε ενιαία διάταξη

Φάση 1: Εφικτή παρεμβολή των εργασιών με προθεσμία

Βήμα 1 Διάταξε στην ουρά προγραμματισμού τις εργασίες με προθεσμία κατά αύξουσα τάξη προθεσμιών. Εστω D ο αριθμός τους.

Βήμα 2 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία A^1_{opt} με μήκος 1. Αν η εργασία περατώνεται πριν την προθεσμία της, τότε FEASIBLE_FLAG = TRUE, αλλιώς FEASIBLE_FLAG = FALSE.
Θέσε k = 2.

Βήμα 3 Αν FEASIBLE_FLAG = TRUE, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας A^{k-1}_{opt} μήκους k-1. Από τα k παραγόμενα προγράμματα A^k_i με i = 1,..,k (στη μερική ακολουθία A^k_i με μήκος k, η εργασία J_k βρίσκεται στην i θέση), διάλεξε το εφικτό ως προς τις προθεσμίες όλων των εργασιών με το μικρότερο C_{max} και ονόμασέ το A^k_{opt}. Αν δεν υπάρχει εφικτό πρόγραμμα, θέσε FEASIBLE_FLAG = FALSE και

$$A^k_{opt} = A^{k-1}_{opt}\ J_k$$

Αν FEASIBLE_FLAG = FALSE, θέσε

$$\mathrm{A}_{opt}^{k} = \mathrm{A}_{opt}^{k-1}\ J_k$$

Βήμα 4 Αν κ $< D$, θέσε k = k + 1 και εκτέλεσε το βήμα 3.
Αλλιώς, θέσε $\mathrm{A}_{opt} = \mathrm{A}_{opt}^{k}$ και πήγαινε στο βήμα 5.

Φάση 2: Παρεμβολή των εργασιών χωρίς προθεσμία

Βήμα 5 Για κάθε εργασία i χωρίς προθεσμία, υπολόγισε :

$$T_i = \sum_{j=1}^{M} p_{ij}$$

Βήμα 6 Διάταξε τις εργασίες χωρίς προθεσμία κατά φθίνουσα τάξη των T_i. Εστω F ο αριθμός τους. Θέσε k = 1. Αν FEASIBLE_FLAG = TRUE, θέσε $B_{opt}^{0} = A_{opt}$. Αν FEASIBLE_FLAG = FALSE, θέσε $B_{opt}^{0} = \{\}$.

Βήμα 7 Αν FEASIBLE_FLAG = TRUE, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις D+k δυνατές θέσεις της μερικής ακολουθίας B_{opt}^{k-1} μήκους D+k-1. Από τα D+k παραγόμενα προγράμματα B_i^k με i = 1,..,D+k (στη μερική ακολουθία B_i^k με μήκος D+k, η εργασία J_k βρίσκεται στην i θέση), διάλεξε το εφικτό με το μικρότερο C_{max} και ονόμασέ το B_{opt}^{k}.

Αν FEASIBLE_FLAG = FALSE, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας B_{opt}^{k-1} μήκους k-1. Από τα k παραγόμενα προγράμματα B_i^k με i = 1,..,k (στη μερική ακολουθία B_i^k με μήκος k, η εργασία J_k βρίσκεται στην i θέση), διάλεξε αυτό για το οποίο η διάταξη $A_{opt}\ B_i^k$ έχει το μικρότερο C_{max} και ονόμασέ το B_{opt}^{k}.

Βήμα 8 Αν k $< F$, θέσε k = k + 1 και εκτέλεσε το βήμα 7.
Αλλιώς, θέσε $B_{opt} = B_{opt}^{k}$ και πήγαινε στο βήμα 9.

Βήμα 9 Αν FEASIBLE_FLAG = TRUE, δώσε ως τελική λύση το πρόγραμμα B_{opt}. Αν FEASIBLE_FLAG = FALSE, δώσε ως τελική λύση το $A_{opt}\ B_{opt}$.

Ο αριθμός των μερικών διατάξεων που εξετάζει ο αλγόριθμος προγραμματισμού ΕΜΠ και ΕΧΠ σε ενιαία διάταξη είναι στη γενική περίπτωση:

$$P = (1 + 2 + .. + D) + (D + 1 + .. + F) = 1 + .. + N = \frac{N(N+1)}{2}$$

όπου D και F ο αριθμός των εργασιών με προθεσμίες (ΕΜΠ) και των εργασιών χωρίς προθεσμίες (ΕΧΠ) αντίστοιχα. Για κάθε μερική διάταξη ο αλγόριθμος υπολογίζει το C_{max} και τους χρόνους περάτωσης των εργασιών στην τελευταία μηχανή από τη σχέση (4) του 2.1.

4.4 Αντίστροφη εκτέλεση των δύο φάσεων

Επειδή ο αλγόριθμος προγραμματισμού ΕΜΠ και ΕΧΠ σε ενιαίο πρόγραμμα λειτουργεί αποδίδοντας απόλυτη προτεραιότητα στο κριτήριο T_{max}, είναι πολύ πιθανό να παράγει λύσεις με κακή απόδοση ως προς το κριτήριο C_{max}. Η μερική διάταξη των ΕΜΠ καθορίζεται στην αρχή της διαδικασίας και ίσως ο αλγόριθμος κατά τη δεύτερη φάση του να μην μπορέσει να καλύψει νεκρούς χρόνους που έχουν παρουσιαστεί σ' αυτήν. Αρα, οι ΕΜΠ αποτελούν τον ανασταλτικό παράγοντα για λύσεις με μικρό χρόνο εκτέλεσης και θα μπορούσαν να προγραμματίζονται στο τέλος της διαδικασίας κατά τρόπο ώστε να μην καταστρέφουν την απόδοση ως προς το C_{max}.

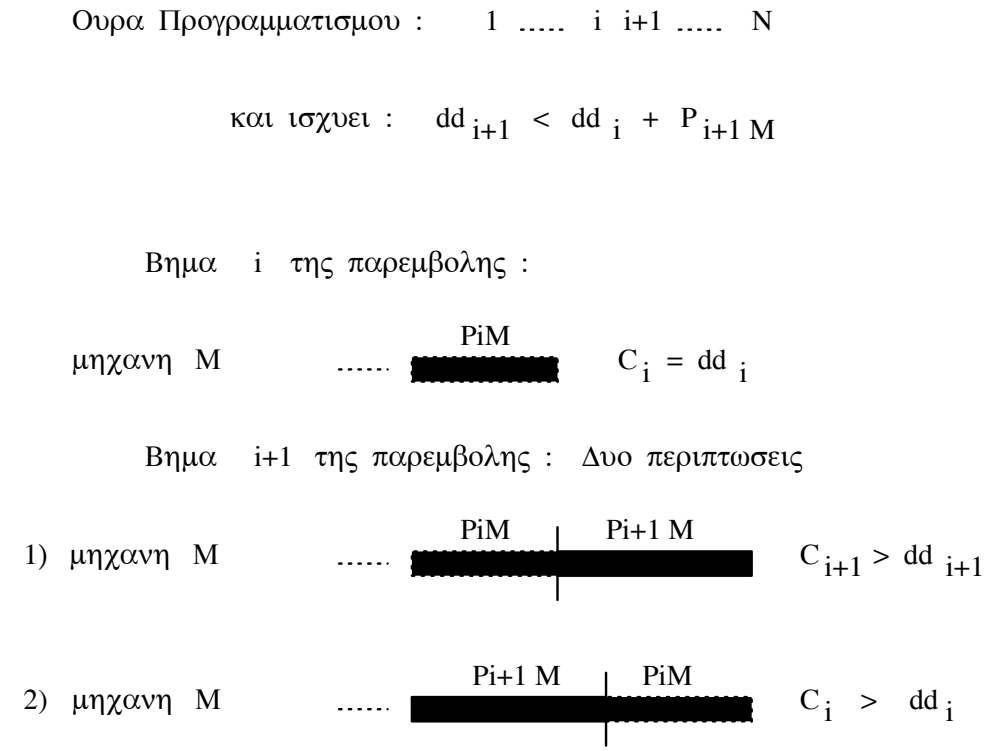
Η ιδέα της εφικτής παρεμβολής των ΕΜΠ στο τέλος της διαδικασίας επιβάλλει την αντίστροφη εκτέλεση των δύο φάσεων. Κατά την πρώτη φάση προγραμματίζουμε τις ΕΧΠ και παράγουμε ένα καλό ως προς το χρόνο εκτέλεσης πρόγραμμα. Κατά τη δεύτερη φάση αναζητούμε εφικτές θέσεις για τις ΕΜΠ μεριμνώντας για την όσο γίνεται μικρότερη διαταραχή στο ήδη διαμορφωμένο πρόγραμμα. Ο αλγόριθμος που υλοποιεί τα παραπάνω εκτελεί αρχικά τη φάση 2 και στη συνέχεια τη φάση 1 του αλγορίθμου προγραμματισμού ΕΜΠ και ΕΧΠ σε ενιαίο πρόγραμμα.

Μετά από υλοποίηση του νέου αλγορίθμου και αξιολόγησή του με τεχνητά προβλήματα ροϊκής παραγωγής, διαπιστώθηκε ότι δεν πλεονεκτεί έναντι του αλγορίθμου που πραγματοποιεί αρχικά την παρεμβολή των ΕΜΠ και στη συνέχεια των ΕΧΠ. Συγκεκριμένα:

1. Η μέγιστη καθυστέρηση και ο αριθμός των καθυστερημένων εργασιών μεγαλώνουν όταν προγραμματίζουμε τις ΕΜΠ στη δεύτερη φάση. Αυτό οφείλεται στην τοποθέτηση των ΕΜΠ σε θέσεις που εμφανίζουν μικρότερη χρονική χαλαρότητα εν γένει, αφού αυτό μπορεί να επιβάλλει το κριτήριο C_{max}. Εστω ότι στο βήμα i η ΕΜΠ i προγραμματίζεται ώστε να ολοκληρώνεται ακριβώς πριν την προθεσμία της. Εστω επίσης ότι η εργασία i+1 έχει μεγαλύτερη προθεσμία από την i, αλλά δεν προλαβαίνει να εκτελεστεί μετά την i, λόγω μεγάλου χρόνου επεξεργασίας. Η

περίπτωση φαίνεται στο σχήμα 6, όπου $dd_{i+1} < dd_i + p_{i+1\,M}$. Για την εργασία i+1 δεν υπάρχει θέση που να οδηγεί σε εφικτό πρόγραμμα, αφού αν αυτή τοποθετεί στο βήμα i+1 πριν την i, η i χάνει την προθεσμία της, ενώ αν τοποθετηθεί μετά την i, χάνει η ίδια την προθεσμία της. Ενδεχομένως στο βήμα i η εργασία i να μπορούσε να εκτελεστεί ενωρίτερα και άρα η εργασία i+1 να μην έχανε την προθεσμία της. Επειδή στο βήμα i+1 δεν αναθεωρούνται οι αποφάσεις του βήματος i, δεν θα υπάρξει στην περίπτωση αυτή εφικτή λύση.

2. Ο συνολικός χρόνος εκτέλεσης των προγραμμάτων όταν προγραμματίζουμε στο τέλος τις ΕΜΠ δεν βελτιώνεται ουσιαστικά. Δεν είναι πάντα δυνατό να τοποθετούνται οι ΕΜΠ στο ήδη διαμορφωμένο πρόγραμμα κατά τρόπο ώστε να μην εμφανίζονται διαταραχές. Στην πράξη δημιουργούν συχνά νέους νεκρούς χρόνους μια και η παρεμβολή τους γίνεται με βασικό κριτήριο το εφικτό των παραγόμενων διατάξεων.



Σχήμα 6: Η εκτέλεση της i χωρίς χρονική χαλαρότητα συνεπάγεται απώλεια του εφικτού.

Για τους παραπάνω λόγους κρίναμε τον αλγόριθμο που αντιστρέφει τις δύο φάσεις προγραμματισμού υποδεέστερο και απορίψαμε την εφαρμογή του στο πρόβλημα $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$. Ως βασικό αλγοριθμικό σχήμα για το προκείμενο πρόβλημα θεωρούμε τη διαδικασία παρεμβολής ΕΜΠ και ΕΧΠ σε ενιαίο πρόγραμμα

όπως περιγράφηκε στο 4.3. Στο αμέσως επόμενο κεφάλαιο θα ενσωματώσουμε στον αλγόριθμο αυτόν ένα μηχανισμό βελτίωσης των λύσεων τόσο ως προς το κριτήριο της μέγιστης καθυστέρησης, όσο και ως προς εκείνο του συνολικού χρόνου εκτέλεσης.

5 Παρεμβολή εργασιών σε σύνολο μερικών διατάξεων

5.1 Απώλεια βελτίστου και εφικτού των λύσεων των αλγορίθμων παρεμβολής

Ο αλγόριθμος παρεμβολής του Nawaz και ο αλγόριθμος εφικτής παρεμβολής εργασιών με προθεσμίες και χωρίς προθεσμίες σε ενιαία διάταξη συγκαταλέγονται στην κατηγορία των ευρηματικών μεθόδων για το πρόβλημα ροϊκής παραγωγής. Η απόδοση της μεθόδου προγραμματισμού εργασιών με παρεμβολή δεν έχει μελετηθεί θεωρητικά, όπως αυτή των αλγορίθμων CDS και Dannenbring της παραγράφου 2.3 [NS89, NS91]. Αν και ο αλγόριθμος παρεμβολής της παραγράφου 2.3.3 παρουσιάζει τα καλύτερα αποτελέσματα μεταξύ των ευρηματικών αλγορίθμων που κατά καιρούς προτάθηκαν (συμπεριλαμβανομένων και των CDS και Dannenbring) [Naw83, Lei90], παράγει λύσεις που απέχουν πιθανώς σημαντικά από το βέλτιστο. Συγκεκριμένα, από τις δοκιμές που πραγματοποιήθηκαν και παρουσιάζονται στο κεφάλαιο 7, αποκτούμε μια γενική εικόνα της ποιότητας των λύσεων. Αν $C_{max}(Nawaz)$ είναι ο χρόνος εκτέλεσης του προγράμματος που παράγει ο αλγόριθμος για κάποιο πρόβλημα και LB το κάτω φράγμα για το βέλτιστο χρόνο εκτέλεσης, που υπολογίζεται από τον τύπο (10) του 2.2, ορίζουμε το σχετικό σφάλμα του αλγορίθμου, $relerr_{Naw}$, ως εξής:

$$relerr_{Naw} = \frac{C_{max}(Nawaz) - LB}{LB}$$

Αν και για το μεγαλύτερο μέρος των τυχαίων προβλημάτων που λύθηκαν το σχετικό σφάλμα του αλγορίθμου κυμαίνονταν σε τιμές κάτω του 0.07, σε ορισμένες περιπτώσεις ξεπέρασε το 0.15 (θα προσδιορίσουμε τους παράγοντες που καθιστούν ”δύσκολο” ένα πρόβλημα ροϊκής παραγωγής στο κεφάλαιο 7). Στο κεφάλαιο αυτό θ’ αναλύσουμε ορισμένες περιπτώσεις εμφάνισης καθυστερήσεων στην εκτέλεση του προγράμματος και θα περιγράψουμε μια τροποποιημένη διαδικασία παρεμβολής που βελτιώνει σημαντικά την ποιότητα των λύσεων.

Η μέθοδος της παρεμβολής εργασιών κατασκευάζει σταδιακά το τελικό πρόγραμμα σε N βήματα. Κύρια χαρακτηριστικά της είναι η διατήρηση της σχετικής διάταξης των ήδη προγραμματισμένων εργασιών και η μυωπική φύση της. Στο i βήμα τοποθετείται η i εργασία από την ουρά προγραμματισμού στην μερική διάταξη μήκους i-1 με επιλογή του προγράμματος με το μικρότερο χρόνο εκτέλεσης μεταξύ όλων των δυνατών μερικών προγραμμάτων. Δηλαδή αποφασίζεται η σχετική θέση της i εργασίας ως προς τις ήδη προγραμματισμένες σε προηγούμενα βήματα. Η απόφαση γίνεται με ”μυωπικά” κριτήρια μια και δεν είναι δυνατόν να γνωρίζουμε εκ των προτέρων τις διαταραχές που μπορούν

να προκαλέσουν οι παρεμβολές των υπολοίπων εργασιών στη συνέχεια της διαδικασίας. Αρα, ενδέχεται ο αλγόριθμος να οδηγηθεί σε κάποιο τοπικό βέλτιστο για την απόσταση του οποίου από το ολικό βέλτιστο δεν υπάρχει γνωστό φράγμα.

Χρονοι επεξεργασιας

εργ	M1	M2	M3	M4
1	0	10	1	5
2	8	2	5	5
3	10	8	8	3
4	9	8	10	0

Χρονοι εξαρμωσης

εργ	M1	M2	M3	M4
1	2	0	2	0
2	0	1	0	2
3	0	2	0	2
4	0	3	0	2

Αλγοριθμος NAWAZ

Ουρα Προγραμματισμου : 3, 4, 2, 1

Βημα 2 :		Βημα 3 :		Βημα 4 :	
Διαταξεις	Cmax	Διαταξεις	Cmax	Διαταξεις	Cmax
3-4	39	2–4–3	46	1–4–3–2	49
4-3	38 →	4–2–3	46	4–1–3–2	55
		4–3–2	45 →	4–3–1–2	50
				4–3–2–1	50

Λυση NAWAZ : 1 - 4 - 3 - 2 με Cmax = 49

Βελτιστη λυση : 1 – 3 – 2 – 4 με Cmax = 45

Σχήμα 7: Μυωπική λειτουργία του αλγορίθμου παρεμβολής

Στο σχήμα 7 μπορούμε να παρακολουθήσουμε την απώλεια του ολικού βελτίστου στη διαδικασία παρεμβολής για το πρόβλημα $N/M/F, P, S_{seq-ind}/C_{max}$. Αφού η αρχική ουρά προγραμματισμού είναι η 3,4,2,1, στο δεύτερο στάδιο του αλγορίθμου παρεμβολής σχηματίζονται οι μερικές ακολουθίες 3-4 και 4-3 με χρόνους εκτέλεσης 39 και 38 αντίστοιχα. Μεταξύ των δύο επιλέγεται η 4-3 ως αποδοτικότερη στο τρέχον βήμα και η τελική λύση είναι το πρόγραμμα 1-4-3-2 με $C_{max} = 49$. Η επιλογή του δεύτερου σταδίου για τη σχετική σειρά εκτέλεσης 4-3 διατηρείται και στο τελικό πρόγραμμα. Το ολικό βέλτιστο για το συγκεκριμένο πρόβλημα είναι το πρόγραμμα 1-3-2-4 με $C_{max} = 45$, στο οποίο η σχετική σειρά εκτέλεσης των εργασιών 3 και 4 είναι 3-4. Επειδή όμως ο αλγόριθμος στο δεύτερο βήμα του επέλεξε με μυωπικά κριτήρια τη σειρά 4-3, είναι

αδύνατο να την αναθεωρήσει στη συνέχεια και να εντοπίσει τη βέλτιστη λύση.

Η αιτία για την απώλεια του βελτίστου κατά τη διαδικασία παρεμβολής είναι η αδυναμία της να αναθεωρήσει αποφάσεις προηγούμενων βημάτων. Αν ήταν δυνατό να επιστρέφει και να αλλάζει τη σχετική θέση εργασιών που έχουν ήδη προγραμματιστεί, η τρέχουσα εργασία ίσως να μπορούσε να τοποθετηθεί ώστε να παράγει μερικό πρόγραμμα με μικρότερο C_{max}.

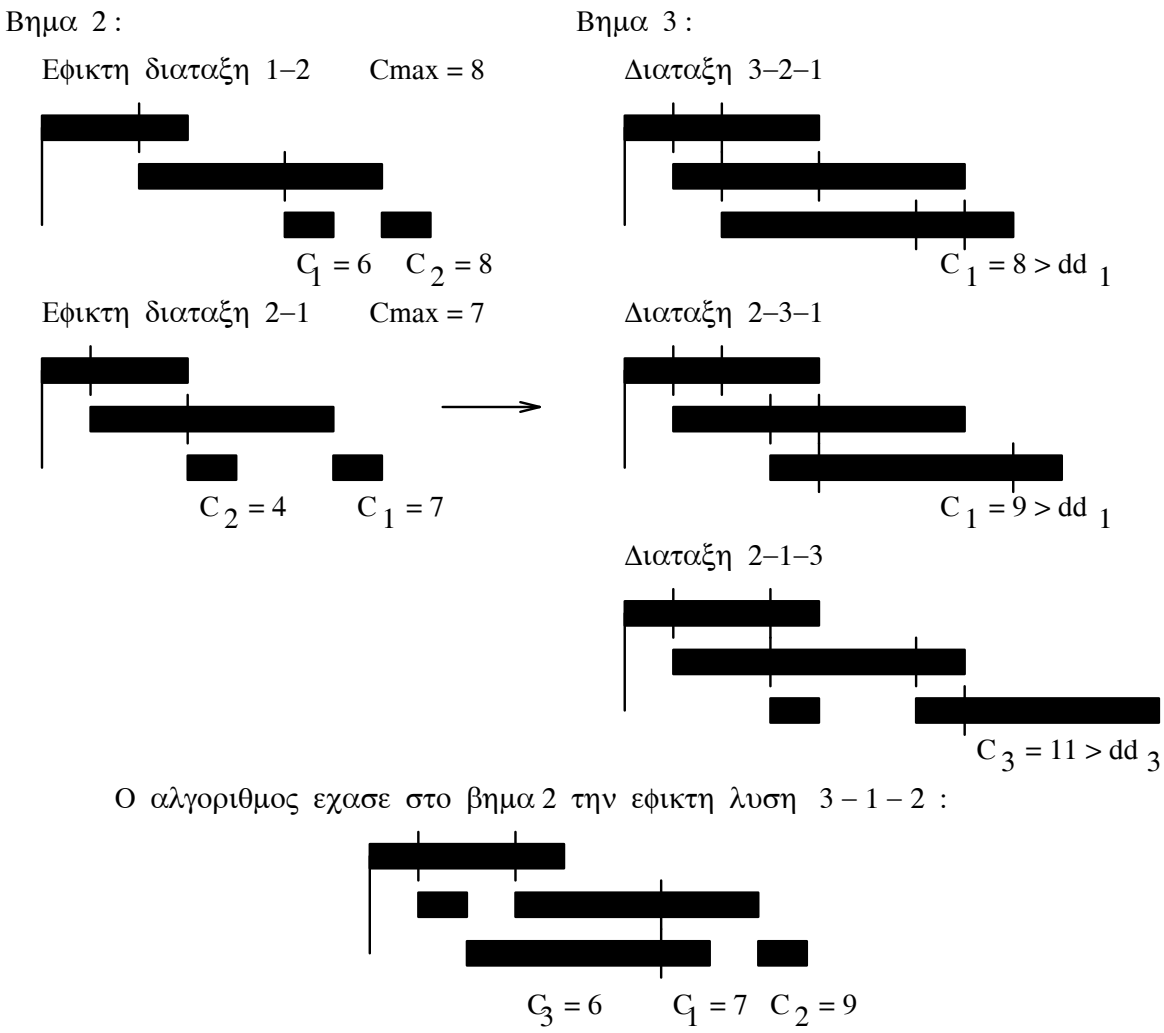
Αντίστοιχοι είναι οι λόγοι για την απώλεια εφικτών λύσεων κατά τον προγραμματισμό με παρεμβολή εργασιών με προθεσμίες. Σε κάθε βήμα της διαδικασίας εφικτής παρεμβολής τοποθετούμε την τρέχουσα εργασία σε τέτοια θέση της μερικής διάταξης ώστε το πρόγραμμα που προκύπτει να είναι το εφικτό με το μικρότερο χρόνο εκτέλεσης. Αρα, το κριτήριο του καλού χρόνου εκτέλεσης είναι δυνατό να τοποθετήσει μια εργασία όσο πιο αργά γίνεται, με αποτέλεσμα να ολοκληρώνεται αμέσως πριν την προθεσμία της. Ετσι, ενώ θα ήταν δυνατή η τοποθέτησή της σε θέση με μεγαλύτερη χρονική χαλαρότητα, δεν υπάρχουν περιθώρια σε εργασίες που προγραμματίζονται σε επόμενα στάδια να παρεμβληθούν πριν από αυτήν στη μερική διάταξη. Αρα, η διαδικασία περιορίζει ουσιαστικά τα πιθανά εφικτά προγράμματα και, ίσως, σε επόμενο βήμα να χάσει το εφικτό της λύσης, χωρίς να μπορεί να το προβλέψει. Αυτό θα συμβεί σε περίπτωση πολύ κοντινών προθεσμιών για τις αμέσως επόμενες εργασίες. Οταν διαπιστωθεί από τον αλγόριθμο παρεμβολής η απώλεια του εφικτού της λύσης, δεν υπάρχει τρόπος αναθεώρησης των αποφάσεων προηγούμενων βημάτων και η εφικτή λύση είναι αδύνατο να προσδιοριστεί.

Η περίπτωση φαίνεται στο σχήμα 8. Στο δεύτερο βήμα του αλγορίθμου οι δυνατές μερικές διατάξεις είναι οι 1-2 και 2-1, οι οποίες είναι και οι δύο εφικτές. Μεταξύ των δύο επιλέγεται η 2-1 μια και έχει μικρότερο C_{max}. Στο βήμα 3 ο αλγόριθμος θα οδηγηθεί σε μη εφικτό πρόγραμμα. Ολα τα παραγόμενα προγράμματα είναι μη εφικτά, αφού στο 3-2-1 και στο 2-3-1 η εργασία 1 είναι καθυστερημένη, και στο 2-1-3 η εργασία 3 είναι καθυστερημένη. Παρόλ' αυτά υπάρχει εφικτή διάταξη για τις τρεις εργασίες, η 3-1-2. Ο αλγόριθμος δεν θα την εξετάσει ποτέ, μια και στο δεύτερο βήμα του επιλέγοντας τη μερική διάταξη 2-1 απέκλεισε κάθε τελικό πρόγραμμα στο οποίο η σχετική σειρά εκτέλεσης των δύο εργασιών είναι 1-2. Ετσι, η απόφαση του δεύτερου βήματος πρέπει απαραίτητα να αναθεωρηθεί προκειμένου να εντοπίσουμε το εφικτό τελικό πρόγραμμα.

Χρονοι επεξεργασιας				Προθεσμιες
εργ	M1	M2	M3	
1	2	3	1	$dd_1 = 7$
2	1	2	1	$dd_2 = 9$
3	1	1	4	$dd_3 = 10$

Αλγοριθμος εφικτης παρεμβολης

Ουρα προγραμματισμου : 1, 2, 3



Ο αλγοριθμος εχασε στο βημα 2 την εφικτη λυση 3 – 1 – 2 :

Σχήμα 8: Η διαδικασία εφικτής παρεμβολής δεν εντοπίζει την εφικτή λύση 3-1-2.

Τα παραδείγματα που αναφέραμε έδειξαν ότι η σειρά προγραμματισμού των εργασιών και το μυωπικό κριτήριο του χρόνου εκτέλεσης της τρέχουσας μερικής ακολουθίας οδηγούν τη διαδικασία σε χώρους έρευνας στους οποίους δεν υπάρχει εγγύηση για το βέλτιστο ή το εφικτό των λύσεων. Η βασική αιτία γι' αυτό είναι οι

απρόβλεπτες διαταραχές που μπορεί να εισαγάγουν στο πρόγραμμα οι εισερχόμενες εργασίες κατά τη λειτουργία του αλγορίθμου και η εμφάνιση νεκρών χρόνων στις μηχανές. Μια άλλη αιτία για το πρόβλημα που μελετάμε στην παρούσα εργασία είναι οι χρόνοι εξάρμωσης των μηχανών. Επειδή το κριτήριο αρχικής διάταξης στην ουρά προγραμματισμού είναι ο συνολικός χρόνος επεξεργασίας και οι προθεσμίες παράδοσης για τον αλγόριθμο παρεμβολής εργασιών και τον αλγόριθμο εφικτής παρεμβολής αντίστοιχα, στην ουρά προγραμματισμού οι ομοειδείς εργασίες δεν είναι γειτονικές εν γένει. Αρα, δύο ομοειδείς εργασίες θα προγραμματιστούν σε βήματα που ενδέχεται να απέχουν πολύ μεταξύ τους. Ετσι, η ομαδοποίησή τους δεν είναι δυνατό να αξιολογηθεί πριν από τη στιγμή παρεμβολής της δεύτερης από αυτές. Ακόμα είναι φανερό ότι η ομαδοποίησή τους, αν συνδυαστεί και με άλλες ανακατατάξεις στη δομή του προγράμματος, μπορεί να βελτιώσει ακόμα περισσότερο το C_{max} ή να οδηγήσει σε εφικτά προγράμματα. Κατά συνέπεια, οι υπό συνθήκες ομαδοποίηση εργασιών συνηγορεί στην ύπαρξη ενός μηχανισμού αναθεώρησης αποφάσεων προηγούμενων βημάτων.

5.2 Αποθήκευση συνόλου μερικών διατάξεων ανά στάδιο

Η μέθοδος της παρεμβολής εργασιών κατασκευάζει σταδιακά το τελικό πρόγραμμα σε μια σειρά Ν βημάτων. Αρα, η άμεση αναθεώρηση παρεμβολών που πραγματοποιήθηκαν σε προηγούμενα στάδια είναι αδύνατη. Θα προτείνουμε σ' αυτό το σημείο την ενσωμάτωση ενός μηχανισμού αποθήκευσης πολλών μερικών διατάξεων ανά στάδιο και θα δείξουμε ότι ο μηχανισμός αυτός επιτρέπει την έμμεση αναθεώρηση αποφάσεων για τις σχετικές θέσεις εργασιών.

Στους αλγορίθμους παρεμβολής που εξετάσαμε μέχρι στιγμής η τρέχουσα εργασία τοποθετείται σε κάθε βήμα σε όλες τις δυνατές θέσεις στη μερική ακολουθία που έχει προκύψει από το προηγούμενο στάδιο. Μεταξύ των παραγόμενων προγραμμάτων επιλέγεται αυτό με το μικρότερο χρόνο εκτέλεσης (για τη διαδικασία εφικτής παρεμβολής απαιτούμε επιπλέον το εφικτό του) και αποτελεί τη βάση για την παρεμβολή σε επόμενα στάδια. Υποθέτοντας ότι δεν κρατάμε μόνο το καλύτερο πρόγραμμα ανά στάδιο, αλλά τα W καλύτερα προγράμματα (ίσως απαιτούμε να είναι και εφικτά), στο αμέσως επόμενο βήμα οι επιλογές για τις πιθανές παρεμβολές αυξάνονται και κατά συνέπεια η ποιότητα των λύσεων βελτιώνεται. Το σύνολο των W προγραμμάτων που αποθηκεύουμε διατηρεί συνεχώς προγράμματα με διαφορετική δομή και αφήνει περιθώρια για παραγωγή περισσότερων μερικών ακολουθιών κατά τη διάρκεια της διαδικασίας που ενδεχομένως να οδηγήσουν σε αποδοτικότερα προγράμματα. Σημαντικό

είναι το γεγονός ότι επιλέγουμε σε κάθε στάδιο τα W καλύτερα προγράμματα, άρα κινούμαστε προς περιοχές έρευνας με καλές τιμές του κριτηρίου βελτίστου. Η διεύρυνση του χώρου έρευνας και η ποιότητα των ενδιάμεσων διατάξεων είναι τα στοιχεία που κάνουν αποτελεσματικό το μηχανισμό αποθήκευσης συνόλου προγραμμάτων. Το W είναι το μέγεθος του συνόλου μερικών διατάξεων που αποθηκεύουμε ανά βήμα και αποτελεί παράμετρο για το μηχανισμό. Στο κεφάλαιο 7 θα μελετηθεί η απόδοση του μηχανισμού για διάφορες τιμές του W και θα φανεί η αποτελεσματικότητά του ακόμα και για μικρές τιμές του W (της τάξεως του 10) σε προβλήματα μεγάλου μεγέθους (πολλές εργασίες).

Εστω, ότι η διαδικασία παρεμβολής βρίσκεται στο i στάδιο της. Από την παρεμβολή της i εργασίας της ουράς προγραμματισμού θα προκύψουν W προγράμματα, στα οποία η i βρίσκεται σε διαφορετική θέση. Στο επόμενο βήμα η εργασία i+1 της ουράς προγραμματισμού θα παρεμβληθεί σε όλες τις δυνατές θέσεις κάθε προγράμματος από τα W του προηγούμενου σταδίου. Αρα, τα προγράμματα που παράγει το i+1 στάδιο είναι (i+1)∗W και σε καθένα απ' αυτά η σχετική θέση των εργασιών i και i+1 είναι διαφορετική. Από αυτά θα επιλεγούν τα W καλύτερα και στο επόμενο βήμα της η διαδικασία θεωρεί W προγράμματα με διαφορετική δομή. Επεκτείνοντας επαγωγικά τον συλλογισμό, διαπιστώνουμε πως στο τελικό σύνολο αποθηκευμένων προγραμμάτων έχουμε W λύσεις, όπου οι σχετικές θέσεις κάποιων εργασιών είναι διαφορετικές. Μεταξύ τους μπορούμε να επιλέξουμε αυτήν με το μικρότερο C_{max} ή αυτήν που οδηγεί σε εφικτές τελικές λύσεις.

Θα εξετάσουμε με ποιο τρόπο ο μηχανισμός αποθήκευσης συνόλου μερικών διατάξεων επιτρέπει έμμεσα την αναθεώρηση αποφάσεων και βελτιώνει τις τελικές λύσεις στα παραδείγματα της προηγούμενης παραγράφου. Αρχικά, θεωρούμε το παράδειγμα του σχήματος 7, όπου ο αλγόριθμος παρεμβολής έδωσε ως τελική λύση το πρόγραμμα 3-4-2-1 με $C_{max} = 49$. Θέτοντας W=2 για το νέο μηχανισμό η παρεμβολή παίρνει τη μορφή του σχήματος 9. Μετά το δεύτερο βήμα του αλγορίθμου οι αποθηκευμένες διατάξεις είναι οι 4-3 και 3-4. Στο τρίτο βήμα εξετάζονται 6 διατάξεις, εκ των οποίων αποθηκεύονται και πάλι οι δύο καλύτερες, 4-3-2 και 3-2-4. Παρατηρούμε ότι η σχετική σειρά εκτέλεσης των εργασιών 3 και 4 είναι διαφορετική για τα δύο προγράμματα και άρα η απόφαση του δευτέρου σταδίου δεν είναι τελική και ενδέχεται να αναθεωρηθεί. Πράγματι, στο τέταρτο βήμα που εξετάζει 8 προγράμματα, η βέλτιστη λύση 1-3-2-4 παράγεται από την ακολουθία 3-2-4 και η απόφαση του βήματος 2 ουσιαστικά αναθεωρείται.

Χρονοι επεξεργασιας

εργ	M1	M2	M3	M4
1	0	10	1	5
2	8	2	5	5
3	10	8	8	3
4	9	8	10	0

Χρονοι εξαρμωσης

εργ	M1	M2	M3	M4
1	2	0	2	0
2	0	1	0	2
3	0	2	0	2
4	0	3	0	2

Αλγοριθμος παρεμβολης σε συνολο μερικων διαταξεων

Αριθμος αποθηκευομενων διαταξεων : W = 2

ουρα προγραμματισμου : 3, 4, 2, 1

Βημα 2 :

Διαταξεις	Cmax
3-4	39
4-3	38

Συνολο WS : [4-3 , 3-4] ⟶

Βημα 3 :

Διαταξεις	Cmax
2–4–3	46
4–2–3	46
4–3–2	45
2–3–4	47
3–2–4	45
3–4–2	49

Συνολο WS : [4–3–2 , 3–2–4] ⟶

Βημα 4 :

Διαταξεις	Cmax
1–4–3–2	49
4–1–3–2	55
4–3–1–2	50
4–3–2–1	50
1–3–2–4	45
3–1–2–4	52
3–2–1–4	52
3–2–4–1	53

τελικη λυση : 1–3–2–4

με Cmax = 45

(ολικο βελτιστο)

Σχήμα 9: Λειτουργία του μηχανισμού παρεμβολής σε σύνολο μερικών διατάξεων (παράδειγμα σχήματος 7).

Αντίστοιχα, στο σχήμα 10 φαίνεται ο εντοπισμός της εφικτής λύσης για το πρόβλημα που τέθηκε στο σχήμα 8. Μετά το τέλος του δευτέρου βήματος, διατηρούμε και τις δύο δυνατές διατάξεις εργασιών 2-1 και 1-2, μια και W=2. Στο επόμενο βήμα θα εντοπιστεί η εφικτή λύση 3-1-2 που παράγεται από τη μερική ακολουθία 1-2. Αν δεν υπήρχε η δυνατότητα αναθεώρησης του προγράμματος 2-1, το οποίο φαίνεται ως η καλύτερη διάταξη στο βήμα 2, η διαδικασία θα οδηγούνταν σε μη εφικτή λύση (σχήμα 8).

Χρονοι επεξεργασιας				Προθεσμιες
εργ	M1	M2	M3	$dd_1 = 7$
1	2	3	1	$dd_2 = 9$
2	1	2	1	$dd_3 = 10$
3	1	1	4	

Αλγοριθμος εφικτης παρεμβολης σε συνολο μερικων διαταξεων (W = 2)

Ουρα προγραμματισμου : 1, 2, 3

Βημα 2 :

Διαταξεις	Cmax	Εφικτες
1–2	8	ναι
2–1	7	ναι

Συνολο WS : [2-1 , 1-2] ⟶

Βημα 3 :

Διαταξεις	Cmax	Εφικτες
3–2–1	8	οχι
2–3–1	9	οχι
2–1–3	11	οχι
3–1–2	9	ναι
1–3–2	11	οχι
1–2–3	12	οχι

Ο αλγοριθμος εντοπισε την εφικτη λυση : 3 – 1 –2

Σχήμα 10: Εφικτή παρεμβολή σε σύνολο μερικών διατάξεων (παράδειγμα σχήματος 8).

5.3 Αλγόριθμος παρεμβολής εργασιών σε σύνολο μερικών διατάξεων

Ο νέος αλγόριθμος παρεμβολής που υλοποιεί την ιδέα της αποθήκευσης πολλών μερικών διατάξεων ανά στάδιο περιγράφεται παρακάτω για το πρόβλημα $N/M/F, P, S_{seq-ind}/C_{max}$.

Αλγόριθμος παρεμβολής σε σύνολο μερικών διατάξεων

Βήμα 1 Για κάθε εργασία i, με i = 1,2,..,N, υπολόγισε :

$$T_i = \sum_{j=1}^{M} p_{ij}$$

Βήμα 2 Σχημάτισε την αρχική ουρά προγραμματισμού με φθίνουσα διάταξη των εργασιών κατά T_i.

Βήμα 3 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία S^1 με μήκος 1.
Θέσε k = 2 και $WS^1 = \{ S^1 \}$ όπου WS^h το σύνολο των μερικών διατάξεων που παράγονται στο στάδιο h.

Βήμα 4 Για κάθε $S_i^{k-1} \epsilon WS^{k-1}$ με i=1,..,W, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας S_i^{k-1} μήκους k-1. Από τα W*k παραγόμενα προγράμματα διάλεξε τα W με το μικρότερο C_{max}, ταξινόμησέ τα ως προς το C_{max} και ονόμασέ τα S_i^k, όπου i = 1,..,W. Τα S_i^k με i = 1,..,W, συγκροτούν το σύνολο WS^k.

Βήμα 5 Αν k < N, θέσε k = k + 1 και εκτέλεσε το βήμα 4. Αλλιώς, πήγαινε στο βήμα 6.

Βήμα 6 Δώσε ως τελική λύση, το S_1^N.

Ο συνολικός αριθμός μερικών διατάξεων που εξετάζει η παραπάνω διαδικασία είναι:

$$P = W \frac{N(N+1)}{2}$$

Στο k βήμα του αλγορίθμου η μερική διάταξη S_1^k μήκους k αντιπροσωπεύει τη λύση με το μέχρι στιγμής καλύτερο C_{max}. Ετσι, αν η μερική διάταξη S_1^{k+1} μήκους k+1 προκύπτει από παρεμβολή της εργασίας J_{k+1} στο πρόγραμμα S_1^k, η διαδικασία κατασκευάζει τη λύση διατηρώντας τη σχετική θέση της εργασίας J_k στο μερικό πρόγραμμα S_1^k. Αν όμως η μερική διάταξη S_1^{k+1} παράγεται από κάποια S_i^k με i=2,..,W, τότε η θέση στην οποία τοποθετήθηκε η εργασία J_k στην καλύτερη διάταξη του σταδίου k δεν διατηρείται στην καλύτερη διάταξη του σταδίου k+1. Αρα, η απόφαση του σταδίου k για τη βέλτιστη τοποθέτηση της εργασίας J_k αναθεωρείται και ο αλγόριθμος ουσιαστικά οπισθοδρομεί προκειμένου να παραγάγει αποδοτικότερη λύση. Αν επεκτείνουμε την παραπάνω ανάλυση σε περισσότερα βήματα του αλγορίθμου, διαπιστώνουμε ότι θεωρητικά είναι δυνατή η αναθεώρηση αποφάσεων που ελήφθησαν οσαδήποτε βήματα πίσω. Αυτό οφείλεται στη δυνατότητα που έχει ο μηχανισμός αποθήκευσης συνόλου λύσεων να συντηρεί προγράμματα με διαφορετική δομή και σχετική σειρά εκτέλεσης εργασιών, παρόλο που ο αριθμός των ανά βήμα αποθηκευόμενων διατάξεων είναι πεπερασμένος και σταθερός. Στο κεφάλαιο 7, όπου αναλύεται η συμπεριφορά του αλγορίθμου για διάφορες τιμές του W, θα φανεί η σημαντική βελτίωση της ποιότητας των λύσεων ακόμα και για μικρό μέγεθος του συνόλου μερικών διατάξεων (τιμές 5-10).

5.4 Αλγόριθμος εφικτής παρεμβολής εργασιών με προθεσμίες σε σύνολο μερικών διατάξεων

Ο μηχανισμός αποθήκευσης συνόλου μερικών διατάξεων επεκτείνεται άμεσα και εφαρμόζεται στο πρόβλημα $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$. Παρακάτω περιγράφεται ο αλγόριθμος που προκύπτει από την ενσωμάτωση του μηχανισμού αυτού στη διαδικασία εφικτής παρεμβολής εργασιών όπως παρουσιάστηκε στην παράγραφο 4.4. Ο μηχανισμός λειτουργεί αποτελεσματικά κατά τη διάρκεια και των 2 φάσεων του αλγορίθμου. Στην πρώτη φάση ενισχύει την ικανότητα του αλγορίθμου να εντοπίζει εφικτά προγράμματα με καλό C_{max} για τις εργασίες με προθεσμίες. Στη δεύτερη φάση διευρύνει το πεδίο έρευνας του αλγορίθμου για διατάξεις εργασιών χωρίς προθεσμίες με ελάχιστο C_{max}.

Αλγόριθμος παρεμβολής ΕΜΠ και ΕΧΠ σε σύνολο μερικών διατάξεων

Φάση 1: Εφικτή παρεμβολή των εργασιών με προθεσμία

Βήμα 1 Διάταξε τις εργασίες με προθεσμία κατά αύξουσα τάξη προθεσμιών. Εστω D ο αριθμός τους.

Βήμα 2 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία A_1^1 με μήκος 1. Αν η εργασία περατώνεται πριν την προθεσμία της, τότε FEASIBLE_FLAG = TRUE, αλλιώς FEASIBLE_FLAG = FALSE. Θέσε k = 2 και $WSA^1 = \{ \mathrm{A}_1^1 \}$.

Βήμα 3 Αν FEASIBLE_FLAG = TRUE, Για κάθε $A_i^{k-1} \epsilon WSA^{k-1}$ με i=1,..,W, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας A_i^{k-1} μήκους k-1. Από τα W∗k παραγόμενα προγράμματα διάλεξε τα W εφικτά με το μικρότερο C_{max}, ταξινόμησέ τα ως προς το C_{max} και ονόμασέ τα A_i^k με i = 1,..,W. Τα A_i^k με i = 1,..,W, συγκροτούν το σύνολο WSA^k. Αν δεν υπάρχει εφικτό πρόγραμμα, θέσε FEASIBLE_FLAG = FALSE και

$$\mathrm{A}_1^k = \mathrm{A}_1^{k-1} J_k$$

Στην περίπτωση αυτή το πρόγραμμα A_1^k αποτελεί το μοναδικό στοιχείο του συνόλου WSA^k.

Αν FEASIBLE_FLAG = FALSE, θέσε

$$\mathrm{A}_1^k = \mathrm{A}_1^{k-1} J_k$$

Στην περίπτωση αυτή το πρόγραμμα A_1^k αποτελεί το μοναδικό στοιχείο του συνόλου WSA^k.

Βήμα 4 Αν k $< D$, θέσε k = k + 1 και εκτέλεσε το βήμα 4. Αλλιώς, θέσε $WSA^{opt} = WSA^k$ και $A_1^{opt} = A_1^k$ και πήγαινε στο βήμα 5.

Φάση 2: Παρεμβολή των εργασιών χωρίς προθεσμία

Βήμα 5 Για κάθε εργασία i χωρίς προθεσμία, υπολόγισε :

$$T_i = \sum_{j=1}^{M} p_{ij}$$

Βήμα 6 Διάταξε τις εργασίες χωρίς προθεσμία κατά φθίνουσα τάξη των T_i. Εστω F ο αριθμός τους. Θέσε k = 1. Αν FEASIBLE_FLAG = TRUE, θέσε $WSB^0 = WSA^{opt}$. Αν FEASIBLE_FLAG = FALSE, θέσε $WSB^0 = \{\}$.

Βήμα 7 Αν FEASIBLE_FLAG = TRUE, για κάθε B_i^{k-1} ϵ WSB^{k-1} με i=1,..,W, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις D+k δυνατές θέσεις της μερικής ακολουθίας B_i^{k-1} μήκους D+k-1. Από τα W∗(D+k) παραγόμενα προγράμματα διάλεξε τα W εφικτά με το μικρότερο C_{max}, ταξινόμησέ τα ως προς το C_{max} και ονόμασέ τα B_i^k με i = 1,..,W. Τα B_i^k με i = 1,..,W, συγκροτούν το σύνολο WSB^k.

Αν FEASIBLE_FLAG = FALSE, για κάθε B_i^{k-1} ϵ WSB^{k-1} με i=1,..,W, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας B_i^{k-1} μήκους k-1. Από τα W∗k παραγόμενα προγράμματα διάλεξε τα W για τα οποία η διάταξη A_1^{opt} B_i^k έχει το μικρότερο C_{max}, ταξινόμησέ τα ως προς το C_{max} και ονόμασέ τα B_i^k με i = 1,..,W. Τα B_i^k με i = 1,..,W, συγκροτούν το σύνολο WSB^k.

Βήμα 8 Αν k $< F$, θέσε k = k + 1 και εκτέλεσε το βήμα 7. Αλλιώς, θέσε $WSB^{opt} = WSB^k$ και πήγαινε στο βήμα 9.

Βήμα 9 Αν FEASIBLE_FLAG = TRUE, δώσε ως τελική λύση το πρόγραμμα B_1^{opt}. Αν FEASIBLE_FLAG = FALSE, δώσε ως τελική λύση το A_1^{opt} B_1^{opt}.

Η συμπεριφορά του αλγορίθμου σε περιπτώσεις απώλειας του εφικτού των διατάξεων υπαγορεύεται από το κριτήριο της ελάχιστης μέγιστης καθυστέρησης. Κατά συνέπεια, αν στο k στάδιο δεν εντοπιστεί εφικτή μερική διάταξη, τότε η μοναδική διάταξη που αποθηκεύεται στο σύνολο WSA^k είναι αυτή που προκύπτει από εκτέλεση της J_k αμέσως μετά της τέλος της WSA_1^{k-1}. Στη συνέχεια, όλες οι επόμενες εργασίες προγραμματίζονται στο τέλος του προγράμματος προκειμένου να μην μεγαλώσει περισσότερο η μέγιστη

καθυστέρηση. Επειδή, λοιπόν, αποθηκεύεται μόνο μια μη εφικτή μερική διάταξη η φάση 2 θα ξεκινήσει με μοναδική αρχική μερική διάταξη και θα τοποθετήσει τις ΕΧΠ μετά το τέλος της.

Ο μέγιστος συνολικός αριθμός μερικών διατάξεων που θεωρεί ο αλγόριθμος είναι:

$$P = W \frac{N(N+1)}{2}$$

6 Επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού

6.1 Απόδοση του κριτηρίου αρχικής διάταξης

Στην αρχική μορφή του αλγορίθμου παρεμβολής εργασιών για το πρόβλημα $N/M/F, S_{seq-ind}/C_{max}$, ο Nawaz προτείνει τη διάταξη των εργασιών στην ουρά προγραμματισμού κατά φθίνοντες συνολικούς χρόνους επεξεργασίας [Naw83]. Εργασίες με μεγάλο συνολικό χρόνο επεξεργασίας αποκτούν μεγαλύτερη προτεραιότητα και προγραμματίζονται στην αρχή της διαδικασίας. Ομως, η επιλογή του συγκεκριμένου κριτηρίου αρχικής διάταξης δεν τεκμηριώνεται από αναλυτική μελέτη της διαδικασίας παρεμβολής. Η υπόθεση ότι εργασίες που επιβαρύνουν το σύστημα των μηχανών περισσότερο είναι καλό να προγραμματίζονται πρώτες, δεν αιτιολογείται θεωρητικά.

Η απόδοση του αλγορίθμου παρεμβολής αποδεικνύεται πειραματικά βελτιωμένη έναντι αυτής των άλλων ευρηματικών μεθόδων της κατηγορίας (εξαιρούνται γενικές μέθοδοι βελτιστοποίησης, όπως η μέθοδος διαμερισμού και φραγής και οι επαναληπτικές μέθοδοι τοπικής αναζήτησης, οι οποίες απαιτούν πολύ περισσότερο υπολογιστικό χρόνο). Από το παράδειγμα του σχήματος 11 φαίνεται η κρισιμότητα του κριτηρίου αρχικής διάταξης για την αποτελεσματικότητα της διαδικασίας. Θεωρώντας τα δεδομένα του παραδείγματος της 5.1 παρατηρούμε ότι η αρχική ουρά προγραμματισμού με διάταξη κατά φθίνοντες χρόνους επεξεργασίας παράγει την τελική λύση 1-4-3-2 με $C_{max} = 49$. Η βέλτιστη διάταξη 1-3-2-4, με $C_{max} = 45$, είναι δυνατό να παραχθεί με τη διαδικασία παρεμβολής, αλλά με διαφορετική αρχική διάταξη των εργασιών στην ουρά προγραμματισμού. Στην προκειμένη περίπτωση μια από τις ουρές προγραμματισμού που οδηγούν στο βέλτιστο πρόγραμμα με τη διαδικασία παρεμβολής είναι και η 3,2,1,4 (σχήμα 11), χωρίς να είναι φανερό το κριτήριο διάταξης των εργασιών σ' αυτή.

Διαφορετικές αρχικές διατάξεις παράγουν λύσεις που διαφέρουν ριζικά μεταξύ τους. Αυτό δικαιολογείται από τον τρόπο σταδιακής κατασκευής της τελικής λύσης και τη μυωπική λειτουργία του κριτηρίου επιλογής των ενδιάμεσων μερικών διατάξεων. Κατά συνέπεια, παρουσιάζει ενδιαφέρον η εξέταση της υπόθεσης του Nawaz σύμφωνα με την οποία το κριτήριο του συνολικού χρόνου εκτέλεσης είναι το αποδοτικότερο για τη διαδικασία παρεμβολής, όπως και η αναζήτηση νέου κριτηρίου αρχικής διάταξης στην ουρά προγραμματισμού αν αυτό οδηγεί σε βελτιωμένες λύσεις.

.

Χρονοι επεξεργασιας

εργ	M1	M2	M3	M4
1	0	10	1	5
2	8	2	5	5
3	10	8	8	3
4	9	8	10	0

Χρονοι εξαρμωσης

εργ	M1	M2	M3	M4
1	2	0	2	0
2	0	1	0	2
3	0	2	0	2
4	0	3	0	2

Αλγοριθμος παρεμβολης με νεα ουρα προγραμματισμου : 3, 2, 1, 4

Βημα 2 :			Βημα 3 :			Βημα 4 :	
Διαταξεις	Cmax		Διαταξεις	Cmax		Διαταξεις	Cmax
3-2	36	→	1–3–2	38	→	4–1–3–2	55
2-3	37		3–1–2	41		1–4–3–2	49
			3–2–1	41		1–3–4–2	51
						1–3–2–4	45

Λυση = 1 – 3 – 2 – 4 με Cmax = 45 (ολικο βελτιστο)

Σχήμα 11: Η ουρά προγραμματισμού 3-2-1-4 οδηγεί στο ολικό βέλτιστο.

6.2 Τυχαία διάταξη εργασιών στην ουρά προγραμματισμού

Με σκοπό να ερευνήσουμε πειραματικά την αποτελεσματικότητα της διάταξης εργασιών στην ουρά προγραμματισμού σύμφωνα με τους χρόνους επεξεργασίας, υλοποιήσαμε τον αλγόριθμο παρεμβολής εργασιών χωρίς προθεσμίες με τυχαία αρχική διάταξη των εργασιών στην ουρά προγραμματισμού. Οι δοκιμές κάλυψαν, τόσο τεχνητά προβλήματα με γνωστή βέλτιστη λύση, όσο και μικρό αριθμό προβλημάτων με τυχαία δεδομένα. Ο παραλλαγμένος αλγόριθμος περιγράφεται στη συνέχεια:

Αλγόριθμος παρεμβολής εργασιών με τυχαία ουρά προγραμματισμού

Βήμα 1 Σχημάτισε την ουρά προγραμματισμού με τυχαία δίαταξη των εργασιών.

Βήμα 2 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία S^1_{opt} με μήκος 1.

Θέσε k = 2.

Βήμα 3 Πάρε την k εργασία της ουράς προγραμματισμού και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας S_{opt}^{k-1} μήκους k-1. Από τα k παραγόμενα προγράμματα S_i^k με i = 1,..,k (στη μερική ακολουθία S_i^k με μήκος k, η τρέχουσα εργασία βρίσκεται στην i θέση), διάλεξε αυτό με το μικρότερο C_{max} και ονόμασέ το S_{opt}^k.

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

Βήμα 5 Δώσε ως τελική λύση, το S_{opt}^N.

Ο παραπάνω αλγόριθμος συγκρίθηκε απ' ευθείας με τον αλγόριθμο Nawaz που παρουσιάσαμε στην παράγραφο 3.3.3. Συγκεκριμένα, οι δύο αλγόριθμοι εφαρμόστηκαν σ' ένα σύνολο 20 περίπου κατασκευασμένων προβλημάτων που κάλυπταν ορισμένες περιπτώσεις ανισοκατανομής των χρόνων κατεργασίας, ώστε να εμφανίζονται νεκροί χρόνοι στις μηχανές. Επίσης, εφαρμόστηκαν και σε προβλήματα με τυχαία αριθμητικά δεδομένα, ο αριθμός των οποίων ήταν αρκετά μεγαλύτερος. Η ποιότητα των τελικών λύσεων των δύο αλγορίθμων διέφερε γενικά σε μεγάλο βαθμό για όλα τα προβλήματα που λύθηκαν. Κατά την εκτέλεση των δοκιμών μετρήθηκε ο αριθμός των περιπτώσεων που ο ένας αλγόριθμος παρήγαγε λύση με μικρότερο C_{max} από τον άλλον. Το ενδιαφέρον αποτέλεσμα που προέκυψε από τη σύγκριση ήταν η ισοκατανομή των περιπτώσεων καλύτερων λύσεων για τους δύο αλγορίθμους. Κανένας από τους δύο δεν υπερείχε σημαντικά εν γένει, ενώ δεν ήταν φανερός από τα δεδομένα ο λόγος για την περιστασιακή υπεροχή του ενός έναντι του άλλου. Αν και ο αριθμός των προβλημάτων που λύθηκαν δεν ήταν αρκετά μεγάλος ώστε να υπάρχει απόλυτη βεβαιότητα για την αξιοπιστία των αποτελεσμάτων, το κριτήριο διάταξης των εργασιών κατά φθίνοντα συνολικό χρόνο επεξεργασίας, που πρότεινε ο Nawaz, δεν φάνηκε αποτελεσματικό σε όλες τις περιπτώσεις.

6.3 Επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού

Από την ανάλυση της απόδοσης του κριτηρίου αρχικής διάταξης εργασιών χωρίς προθεσμίες στην ουρά προγραμματισμού προέκυψαν δύο ενδιαφέροντα συμπεράσματα. Το πρώτο είναι η σπουδαιότητα της σειράς παρεμβολής των εργασιών κατά τη διαδικασία προγραμματισμού για την παραγωγή τελικών προγραμμάτων με ελαχιστοποιημένο C_{max}. Το δεύτερο είναι η διαφαινόμενη αδυναμία ανάδειξης ενός κριτηρίου αρχικής διάταξης με σημαντική υπεροχή έναντι των άλλων.

Σ' αυτό το σημείο θα παρουσιάσουμε μια διαδικασία παρεμβολής που δεν χρησιμοποιεί κανένα κριτήριο αρχικής διάταξης στην ουρά προγραμματισμού. Οι εργασίες χωρίς προθεσμίες τοποθετούνται τυχαία στην ουρά προγραμματισμού και η σειρά παρεμβολής τους δεν παρέχει καμιά εγγύηση για την ποιότητα της τελικής λύσης. Ομως, η διαδικασία εκτελείται επαναληπτικά R φορές με διαφορετική τυχαία ουρά προγραμματισμού κάθε φορά. Μεταξύ των διαφορετικών λύσεων που παράγονται στην περίπτωση αυτή, επιλέγεται αυτή με το μικρότερο συνολικό χρόνο εκτέλεσης. Αναμένεται ότι κάποια από τις διαφορετικές αρχικές διατάξεις θα οδηγήσει σε καλό τελικό πρόγραμμα και ότι η ποιότητα των λύσεων θα βελτιώνεται με την αύξηση του αριθμού (R) των εκτελέσεων της διαδικασίας. Η υπόθεση αυτή δεν τεκμηριώνεται με πιθανολογική ανάλυση για την ποιότητα μιας λύσης που προέρχεται από τυχαία αρχική διάταξης. Αυτό προϋποθέτει το συσχετισμό δομής της ουράς προγραμματισμού και δομής του βέλτιστου προγράμματος, κάτι που δεν είναι δυνατό για το πρόβλημα, όπως μελετήθηκε στην παρούσα εργασία.

Αλγόριθμος επαναληπτικής παρεμβολής με τυχαίες ουρές προγραμματισμού

Βήμα 1 θέσε r=1 και $OptSol = \{\ \}$.

Βήμα 2 Σχημάτισε την ουρά προγραμματισμού με τυχαία διάταξη των εργασιών.

Βήμα 3 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία S^1_{opt} με μήκος 1.
Θέσε k = 2.

Βήμα 4 Πάρε την k εργασία της ουράς προγραμματισμού και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας S^{k-1}_{opt} μήκους k-1. Από τα k παραγόμενα προγράμματα S^k_i με i = 1,..,k (στη μερική ακολουθία S^k_i με μήκος k, η τρέχουσα εργασία βρίσκεται στην i θέση), διάλεξε αυτό με το μικρότερο C_{max} και ονόμασέ το S^k_{opt}.

Βήμα 5 Αν k $<$ N, θέσε k = k + 1 και εκτέλεσε το βήμα 4. Αλλιώς, πήγαινε στο βήμα 6.

Βήμα 6 Αν $C_{max}(S^N_{opt}) < C_{max}(OptSol)$, τότε $OptSol = S^N_{opt}$.

Βήμα 7 Αν $r < R$, θέσε r = r + 1 και εκτέλεσε το βήμα 2. Αλλιώς, πήγαινε στο βήμα 8.

Βήμα 8 Δώσε ως τελική λύση την $OptSol$.

Για τον αριθμό των μερικών διατάξεων που εξετάζει ο αλγόριθμος ισχύει:

$$P = R\,\frac{N(N+1)}{2}$$

Συγκριτική μελέτη των αποτελεσμάτων του αλγορίθμου επαναληπτικής παρεμβολής με τυχαίες ουρές προγραμματισμού, του αλγορίθμου του Nawaz και του αλγορίθμου παρεμβολής σε σύνολο μερικών διατάξεων γίνεται στο κεφάλαιο 7. Γενικά, η επανάληψη της διαδικασίας παρεμβολής με διαφορετικές αρχικές ουρές προγραμματισμού βελτίωσε σε μεγάλο βαθμό την ποιότητα των παραγόμενων λύσεων. Το κόστος για τη βελτίωση αυτή ήταν η σημαντική αύξηση του υπολογιστικού χρόνου για τη διαδικασία.

6.4 Επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού σε σύνολο μερικών διατάξεων

Η αποτελεσματικότητα των αλγορίθμων παρεμβολής εξαρτάται, όπως αναφέρθηκε, από δύο κρίσιμα στοιχεία: τη διάταξη των εργασιών στην ουρά προγραμματισμού και τη δυνατότητα αναθεώρησης αποφάσεων προηγούμενων σταδίων για τις σχετικές θέσεις των εργασιών. Στο κεφάλαιο αυτό προτάθηκε η επαναληπτική εκτέλεση της παρεμβολής με τυχαίες ουρές προγραμματισμού με σκοπό την βελτίωση της απόδοσης της διαδικασίας. Στο προηγούμενο κεφάλαιο παρουσιάστηκε ο μηχανισμός αποθήκευσης συνόλου μερικών διάταξεων και φάνηκε πώς αυτός επιτρέπει την έμμεση αναθεώρηση των σχετικών θέσεων των εργασιών.

Σ' αυτό το σημείο θα περιγράψουμε ένα νέο αλγόριθμο παρεμβολής που χρησιμοποιεί και τους δύο μηχανισμούς για τον εντοπισμό προγραμμάτων εργασιών χωρίς προθεσμίες με μικρό συνολικό χρόνο εκτέλεσης. Ο αλγόριθμος πραγματοποιεί παρεμβολή σε σύνολο μερικών διατάξεων, ενώ η διαδικασία παρεμβολής επαναλαμβάνεται με διαφορετική κάθε φορά τυχαία ουρά προγραμματισμού. Ετσι, μεταξύ των λύσεων που παράγονται, επιλέγεται εκείνη με το μικρότερο C_{max}. Η διατύπωση του αλγορίθμου παράγεται απευθείας από συνδυασμό του αλγορίθμου παρεμβολής εργασιών σε σύνολο μερικών διάταξεων της 5.3, και του αλγορίθμου επαναληπτικής παρεμβολής της 6.3 :

Αλγόριθμος επαναληπτικής παρεμβολής με τυχαίες ουρές προγραμματισμού σε σύνολο μερικών διατάξεων

Βήμα 1 θέσε r=1 και $OptSol = \{\}$.

Βήμα 2 Σχημάτισε την ουρά προγραμματισμού με τυχαία διάταξη των εργασιών.

Βήμα 3 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία S^1 με μήκος 1.
Θέσε k = 2 και $WS^1 = \{ S^1 \}$ όπου WS^h το σύνολο των μερικών διατάξεων που παράγονται στο στάδιο h.

Βήμα 4 Για κάθε $S_i^{k-1} \epsilon WS^k$ με i=1,..,W, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας S_i^{k-1} μήκους k-1. Από τα W∗k παραγόμενα προγράμματα διάλεξε τα W με το μικρότερο C_{max}, ταξινόμησέ τα ως προς το C_{max} και ονόμασέ τα S_i^k με i = 1,..,W. Τα S_i^k με i = 1,..,W, συγκροτούν το σύνολο WS^k.

Βήμα 5 Αν k $<$ N, θέσε k = k + 1 και εκτέλεσε το βήμα 4. Αλλιώς, πήγαινε στο βήμα 6.

Βήμα 6 Αν $C_{max}(S_1^N) < C_{max}(OptSol)$, τότε $OptSol = S_1^N$.

Βήμα 7 Αν $r < R$, θέσε r = r + 1 και εκτέλεσε το βήμα 2. Αλλιώς, πήγαινε στο βήμα 8.

Βήμα 8 Δώσε ως τελική λύση την $OptSol$.

Ο συνολικός αριθμός μερικών διατάξεων που εξετάζει η παραπάνω διαδικασία είναι:

$$P = R\,W\,\frac{N(N+1)}{2}$$

Είναι φανερό ότι μεγάλες τιμές των παραμέτρων R και W ενδέχεται να αυξήσουν σημαντικά το χρόνο εκτέλεσης του αλγορίθμου, αν ο αριθμός των εργασιών είναι μεγάλος. Στο κεφάλαιο 7 θα μελετήσουμε τη συμπεριφορά του αλγορίθμου σχετικά με όλες τις παραμέτρους του προβλήματος και θα δείξουμε ότι αρκούν μικρές τιμές για τα R και W προκειμένου να βελτιωθεί η ποιότητα των λύσεων των αλγορίθμων παρεμβολής.

6.5 Επαναληπτική εκτέλεση της παρεμβολής ΕΧΠ σε σύνολο μερικών διατάξεων

Ο μηχανισμός επαναληπτικής παρεμβολής με τυχαίες ουρές προγραμματισμού προτάθηκε για εργασίες για την περάτωση των οποίων δεν υπάρχει χρονικός περιορισμός και ενσωματώθηκε στη διαδικασία παρεμβολής εργασιών χωρίς προθεσμίες. Η εφαρμογή του για τη διαδικασία εφικτής παρεμβολής εργασιών με προθεσμίες δεν είναι δυνατή. Ο λόγος έγκειται στην κρισιμότητα του κριτηρίου αρχικής διάταξης εργασιών με προθεσμίες στην ουρά προγραμματισμού κατά αύξουσα τάξη προθεσμιών. Οπως

δείξαμε στην 3.2, οποιαδήποτε άλλη σειρά προγραμματισμού ΕΜΠ συνεπάγεται σε πολλές περιπτώσεις απώλεια του εφικτού των λύσεων. Κατά συνέπεια δεν θα είχε νόημα η επαναληπτική εφικτή παρεμβολή ΕΜΠ με τυχαίες ουρές προγραμματισμού, μια και πρωταρχική μέριμνα του προγραμματισμού ΕΜΠ είναι η κατασκευή εφικτών προγραμμάτων και, όταν αυτό δεν είναι δυνατό, προγραμμάτων με μικρές καθυστερήσεις εργασιών.

Επειδή στο πλαίσιο της παρούσας εργασίας υποθέτουμε ότι προγραμματίζουμε μίγμα εργασιών με προθεσμίες και χωρίς προθεσμίες, θα χρησιμοποιήσουμε την επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού κατά τη φάση παρεμβολής των ΕΧΠ. Ετσι, ο μηχανισμός ενσωματώνεται στη φάση 2 του αλγορίθμου εφικτής παρεμβολής εργασιών σε σύνολο μερικών διατάξεων της 5.4. Κατά τη φάση 1 παράγεται μια τελική διάταξη για τις ΕΜΠ και στη συνέχεια οι ΕΧΠ διατάσσονται επαναληπτικά σε τυχαίες ουρές προγραμματισμού και παρεμβάλλονται, είτε μεταξύ των ΕΜΠ, αν αυτό δεν συνεπάγεται απώλεια του εφικτού των λύσεων, είτε μετά από αυτές στο τελικό πρόγραμμα.

Αλγόριθμος παρεμβολής ΕΜΠ και ΕΧΠ σε σύνολο μερικών διατάξεων

Φάση 1: Εφικτή παρεμβολή των εργασιών με προθεσμία

Βήμα 1 Διάταξε τις ΕΜΠ στην ουρά προγραμματισμού κατά αύξουσα τάξη προθεσμιών. Εστω D ο αριθμός τους.

Βήμα 2 Τοποθέτησε την πρώτη εργασία της ουράς προγραμματισμού στην αρχή του προγράμματος για να σχηματίσεις τη μερική ακολουθία A_1^1 με μήκος 1. Αν η εργασία περατώνεται πριν την προθεσμία της, τότε FEASIBLE_FLAG = TRUE, αλλιώς FEASIBLE_FLAG = FALSE.
Θέσε k = 2 και $WSA^1 = \{ A_1^1 \}$.

Βήμα 3 Αν FEASIBLE_FLAG = TRUE, Για κάθε $A_i^{k-1} \epsilon WSA^{k-1}$ με i=1,..,W, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας A_i^{k-1} μήκους k-1. Από τα W*k παραγόμενα προγράμματα διάλεξε τα W εφικτά με το μικρότερο C_{max}, ταξινόμησέ τα ως προς το C_{max} και ονόμασέ τα A_i^k με i = 1,..,W. Τα A_i^k με i = 1,..,W, συγκροτούν το σύνολο WSA^k. Αν δεν υπάρχει εφικτό πρόγραμμα, θέσε FEASIBLE_FLAG = FALSE και

$$A_1^k = A_1^{k-1} J_k$$

Στην περίπτωση αυτή το πρόγραμμα A_1^k αποτελεί το μοναδικό στοιχείο του συνόλου WSA^k.

Αν FEASIBLE_FLAG = FALSE, θέσε

$$\mathrm{A}_1^k = \mathrm{A}_1^{k-1}\, J_k$$

Στην περίπτωση αυτή το πρόγραμμα A_1^k αποτελεί το μοναδικό στοιχείο του συνόλου WSA^k.

Βήμα 4 Αν k $< D$, θέσε k = k + 1 και εκτέλεσε το βήμα 3.
Αλλιώς, θέσε $WSA^{opt} = WSA^k$ και $A_1^{opt} = A_1^k$ και πήγαινε στο βήμα 5.

Φάση 2: Παρεμβολή των εργασιών χωρίς προθεσμία

Βήμα 5 θέσε r=1 και $OptSol = \{\ \}$.

Βήμα 6 Διάταξε τις εργασίες χωρίς προθεσμία με τυχαία σειρά στην ουρά προγραμματισμού. Εστω F ο αριθμός τους. Θέσε k = 1.
Αν FEASIBLE_FLAG = TRUE, θέσε $WSB^0 = WSA^{opt}$.
Αν FEASIBLE_FLAG = FALSE, θέσε $WSB^0 = \{\}$.

Βήμα 7 Αν FEASIBLE_FLAG = TRUE, για κάθε $B_i^{k-1}\ \epsilon\ WSB^{k-1}$ με i=1,..,W, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις D+k δυνατές θέσεις της μερικής ακολουθίας B_i^{k-1} μήκους D+k-1. Από τα W∗(D+k) παραγόμενα προγράμματα διάλεξε τα W εφικτά με το μικρότερο C_{max}, ταξινόμησέ τα ως προς το C_{max} και ονόμασέ τα B_i^k με i = 1,..,W. Τα B_i^k με i = 1,..,W, συγκροτούν το σύνολο WSB^k.

Αν FEASIBLE_FLAG = FALSE, για κάθε $B_i^{k-1}\ \epsilon\ WSB^{k-1}$ με i=1,..,W, πάρε την k εργασία της ουράς προγραμματισμού J_k και παρέμβαλέ την διαδοχικά στις k δυνατές θέσεις της μερικής ακολουθίας B_i^{k-1} μήκους k-1. Από τα W∗k παραγόμενα προγράμματα διάλεξε τα W για το οποία η διάταξη $A_1^{opt}\ B_i^k$ έχει το μικρότερο C_{max}, ταξινόμησέ τα ως προς το C_{max} και ονόμασέ τα B_i^k με i = 1,..,W. Τα B_i^k με i = 1,..,W, συγκροτούν το σύνολο WSB^k.

Βήμα 8 Αν k $< F$, θέσε k = k + 1 και εκτέλεσε το βήμα 7.
Αλλιώς, θέσε $WSB^{opt} = WSB^k$ και πήγαινε στο βήμα 9.

Βήμα 9 Αν FEASIBLE_FLAG = TRUE, τότε $CurSol = B_1^{opt}$.
Αν FEASIBLE_FLAG = FALSE, τότε $CurSol = A_1^{opt}\ B_1^{opt}$.

Βήμα 10 Αν $C_{max}(CurSol) < C_{max}(OptSol)$, τότε $OptSol = CurSol$.

Βήμα 11 Αν $r < R$, θέσε r = r + 1 και εκτέλεσε το βήμα 6. Αλλιώς, πήγαινε στο βήμα 12.

Βήμα 12 Δώσε ως τελική λύση την $OptSol$.

Για το συνολικό αριθμό μερικών διατάξεων που εξετάζει η παραπάνω διαδικασία ισχύει:

$$P = W(1 + .. + D + R(D+1) + .. + RN) = W\ (\frac{D(D+1)}{2} + R\ \frac{F(F+1)}{2} + R\ F\ D)$$

7 Πειραματική αξιολόγηση αλγορίθμων

Οι αλγόριθμοι που θεωρούμε στα πλαίσια αυτής της εργασίας ανήκουν στην κατηγορία των ευρηματικών μεθόδων για το πρόβλημα ροϊκής παραγωγής. Επειδή ο σχεδιασμός τους δεν προέκυψε από αναλυτική μελέτη των ιδιοτήτων των βέλτιστων λύσεων, δεν μπορούμε να προσδιορίσουμε φράγματα για την απόδοση τους. Ο μοναδικός τρόπος αξιολόγησης των επιδόσεων τους είναι η οργάνωση συστηματικών δοκιμών πάνω σε μεγάλο αριθμό τυχαίων προβλημάτων. Οι δοκιμές που παρουσιάζουμε στο κεφάλαιο αυτό αφορούν και τα δύο προβλήματα που εξετάζουμε: το $N/M/F, P, S_{seq-ind}/C_{max}$, όπου επιδιώκουμε την ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης προγράμματος, και το $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$, όπου αναζητούμε εφικτά ως προς τις προθεσμίες των εργασιών προγράμματα με μικρό χρόνο εκτέλεσης ή, όταν αυτό δεν είναι δυνατό, προγράμματα με μικρή καθυστέρηση των εργασιών με προθεσμίες. Καταρχήν θα ορίσουμε τα κριτήρια επιδόσεων των αλγορίθμων, στη συνέχεια θα αιτιολογήσουμε τις επιλογές που αφορούσαν στα διάφορα μεγέθη προβλημάτων και στις τιμές των παραμέτρων και θα μελετήσουμε τη συμπεριφορά των αλγορίθμων χωριστά για κάθε πρόβλημα. Στο τέλος του κεφαλαίου θα παρακολουθήσουμε τη μεταβολή του απαιτούμενου υπολογιστικού χρόνου εκτέλεσης των αλγορίθμων σε σχέση με τις παραμέτρους έλέγχου και το μέγεθος των προβλημάτων.

Πρόβλημα $N/M/F, P, S_{seq-ind}/C_{max}$

Για το προκείμενο πρόβλημα σκοπός των πειραμάτων ήταν η εκτίμηση των βελτιώσεων που συνεπάγεται η ενσωμάτωση στη διαδικασία παρεμβολής των δύο νέων μηχανισμών: της αποθήκευσης συνόλου μερικών διατάξεων και της επαναληπτικής εκτέλεσης της διαδικασίας με τυχαίες ουρές προγραμματισμού. Ετσι, υλοποιήσαμε και συγκρίναμε τους εξής αλγορίθμους:

- Αλγόριθμο παρεμβολής του Nawaz (υποκεφάλαιο 3.3.3). Θα τον αναφέρουμε ως **Nawaz-Algo**.
- Αλγόριθμο παρεμβολής σε σύνολο μερικών διατάξεων (υποκεφάλαιο 5.3) με παράμετρο ελέγχου τον αριθμό των αποθηκευόμενων ανά στάδιο μερικών διατάξεων (W). Θα τον αναφέρουμε ως **W-Interpol**.
- Αλγόριθμο επαναληπτικής παρεμβολής με τυχαίες ουρές προγραμματισμού (υποκεφάλαιο 6.3) με παράμετρο ελέγχου τον αριθμό των επαναλήψεων (R). Θα τον αναφέρουμε ως **R-Interpol**.

- Αλγόριθμο επαναληπτικής παρεμβολής με τυχαίες ουρές προγραμματισμού σε σύνολο μερικών διατάξεων (υποκεφάλαιο 6.4) με παραμέτρους ελέγχου τον αριθμό των αποθηκευόμενων ανά στάδιο μερικών διατάξεων (W) και τον αριθμό των επαναλήψεων (R). Θα τον αναφέρουμε ως **WR-Interpol**.

Πρόβλημα $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$

Αφού στη σχετική με τη ροϊκή παραγωγή βιβλιογραφία δεν αναφέρεται αλγόριθμος επίλυσης του προβλήματος, παραθέτουμε απλώς τα πειραματικά αποτελέσματα που αφορούν το εφικτό των λύσεων, τη μέγιστη καθυστέρηση των εργασιών και το συνολικό χρόνο εκτέλεσης των τελικών προγραμμάτων. Επίσης, αξιολογείται η βελτίωση της διαδικασίας εφικτής παρεμβολής με τους δύο νέους μηχανισμούς που παρουσιάσαμε σε προηγούμενα κεφάλαια. Συγκεκριμένα, συγκρίναμε τους παρακάτω αλγορίθμους:

- Αλγόριθμο εφικτής παρεμβολής (υποκεφάλαιο 4.3). Θα τον αναφέρουμε ως **Feas-Interpol**.
- Αλγόριθμο εφικτής παρεμβολής σε σύνολο μερικών διατάξεων (υποκεφάλαιο 5.4) με παράμετρο ελέγχου τον αριθμό των αποθηκευόμενων ανά στάδιο μερικών διατάξεων (W). Θα τον αναφέρουμε ως **W-Feas-Interpol**.
- Αλγόριθμο εφικτής παρεμβολής σε σύνολο μερικών διατάξεων με επαναληπτική εκτέλεση της παρεμβολής ΕΧΠ (υποκεφάλαιο 6.5) με παραμέτρους ελέγχου τον αριθμό των αποθηκευόμενων ανά στάδιο μερικών διατάξεων (W) και τον αριθμό των επαναλήψεων (R) της φάσης παρεμβολής των ΕΧΠ. Θα τον αναφέρουμε ως **WR-Feas-Interpol**.

7.1 Κριτήρια Επιδόσεων

Στο $N/M/F, P, S_{seq-ind}/C_{max}$ μοναδικό κριτήριο βελτίστου είναι ο συνολικός χρόνος εκτέλεσης προγράμματος. Συμβολίζοντας με $C_{max}(A)$ το συνολικό χρόνο εκτέλεσης της λύσης που παράγει ο αλγόριθμος Α για κάποιο πρόβλημα και με LB το κάτω φράγμα του βέλτιστου χρόνο εκτέλεσης για το ίδιο πρόβλημα όπως υπολογίζεται από τον τύπο (10) της 2.2, ορίζουμε τα παρακάτω κριτήρια επιδόσεων:

- απόλυτο σφάλμα αλγορίθμου Α, $error_A$, με:

$$error_A = C_{max}(A) - LB$$

- σχετικό σφάλμα αλγορίθμου Α, $relerror_A$, με:

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

- σχετική βελτίωση αλγορίθμου Α ως προς αλγόριθμο Β, $imprv_{A,B}$, με:

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

Στο $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$ πραγματοποιούμε δικριτηριακό προγραμματισμό ως προς τη μέγιστη καθυστέρηση των εργασιών με προθεσμίες και το συνολικό χρόνο εκτέλεσης του προγράμματος. Θεωρώντας ως πρωτεύον κριτήριο το πρώτο, θα αξιολογήσουμε τον αλγόριθμο Α με βάση τις παρακάτω μετρήσεις:

- αριθμός περιπτώσεων που ο αλγόριθμος Α εντόπισε εφικτές λύσεις
- αριθμός περιπτώσεων που ο αλγόριθμος Α μείωσε τη μέγιστη καθυστέρηση που παρουσίασε η λύση του αλγορίθμου Feas-Interpol
- σχετική βελτίωση χρόνου εκτέλεσης της λύσης του αλγορίθμου Α ως προς τη λύση του αλγορίθμου Β, $imprv_{A,B}$, με:

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

Στο σημείο αυτό πρέπει να σημειώσουμε ότι επειδή το σφάλμα υπολογίζεται ως η απόσταση από το κάτω φράγμα για το βέλτιστο χρόνο εκτέλεσης και όχι από το ίδιο το βέλτιστο, ενδέχεται να απέχει σημαντικά από το πραγματικό σφάλμα. Η εκτίμηση για το κάτω φράγμα δεν συμπεριλαμβάνει τυχόντες νεκρούς χρόνους στην εκτέλεση και αποτελεί ιδιαίτερα χαλαρή προσέγγιση της πραγματικής τιμής του βελτίστου. Ετσι, τα συμπεράσματά μας θα αφορούν κυρίως στις διακυμάνσεις των απολύτων και σχετικών σφαλμάτων και στις ποσοστιαίες βελτιώσεις τους, χωρίς οι τιμές των επιδόσεων να αποτελούν ασφαλές μέτρο της απόδοσης των αλγορίθμων.

7.2 Δοκιμές για το $N/M/F, P, S_{seq-ind}/C_{max}$

Η δημιουργία και επίλυση τυχαίων προβλημάτων ροϊκής παραγωγής πραγματοποιήθηκε με δύο βασικές επιδιώξεις: αφενός την εκτίμηση της απόδοσης των αλγορίθμων παρεμβολής για τα διάφορα μεγέθη προβλημάτων και αφετέρου τη διερεύνηση των βελτιωμένων αλγορίθμων που προτείνουμε στην εργασία για διάφορες τιμές των

παραμέτρων ελέγχου της παρεμβολής (W και R). Ετσι, τα μεταβαλλόμενα μεγέθη των δοκιμών ήταν συνολικά τέσσερα:

1. N : ο αριθμός των εργασιών προς προγραμματισμό.
   Κατά τις διάφορες συγκριτικές δοκιμές το N πήρε τιμές στο σύνολο {6,8,10,20,30,40,50}. Προβλήματα με 6,8 ή 10 εργασίες θεωρούνται μικρού μεγέθους, ενώ εκείνα με 20,30,40 ή 50 εργασίες, μεγάλου μεγέθους. Στη σχετική βιβλιογραφία ο μέγιστος αριθμός εργασιών των προβλημάτων που λύνονται είναι συνήθως 30-40 [CGD91, CB92, Lei90, WH89].

2. M : ο αριθμός των μηχανών.
   Ο αριθμός των σταδίων παραγωγής των συστημάτων ροϊκής παραγωγής που συναντάμε στην πράξη δεν ξεπερνά συνήθως το 10. Συστήματα με μεγαλύτερο αριθμό μηχανών θεωρούνται πολύπλοκα. Στα πλαίσια της εργασίας, ορίστηκε το M να παίρνει τιμές στο σύνολο {4,8,12,16}.

3. W : ο αριθμός των αποθηκευόμενων ανά στάδιο μερικών διατάξεων.
   Προκειμένου ο απαιτούμενος υπολογιστικός χρόνος για την εκτέλεση των αλγορίθμων να είναι της τάξης του λεπτού ακόμα και για μεγάλα προβλήματα (40-50) εργασίες, ως μέγιστη τιμή για το W ορίστηκε το 40 (λεπτομερής πίνακας για τον υπολογιστικό χρόνο των αλγορίθμων δίδεται στην παράγραφο 7.4).

4. R : ο αριθμός των εκτελέσεων της διαδικασίας παρεμβολής.
   Επειδή ο χρόνος εκτέλεσης των αλγορίθμων αναμένεται περίπου ανάλογος του γινομένου των W και R (βλ. πίνακα υπολογιστικού χρόνου του 7.4), η μέγιστη τιμή για το R ορίστηκε το 40 για να εξασφαλίζεται δίκαια σύγκριση των βελτιώσεων για τις διάφορες τιμές των δύο παραμέτρων ελέγχου.

Για κάθε ζεύγος τιμών (N,M) δημιουργήθηκαν και λύθηκαν 30 τυχαία προβλήματα. Τα κριτήρια επιδόσεων που αξιολογήθηκαν αφορούσαν πάντα μέσους όρους τιμών για τα 30 διαφορετικά προβλήματα. Η εξάρτηση των επιδόσεων από τα αριθμητικά δεδομένα του προβλήματος ήταν η αιτία για τη μεγάλη διακύμανση στα αποτελέσματα των αλγορίθμων παρεμβολής. Με την επίλυση μεγάλου αριθμού διαφορετικών τυχαίων προβλημάτων ανά κατηγορία, εξομαλύνθηκαν ενδεχόμενες αποκλίσεις και ενισχύθηκε η αξιοπιστία των αποτελεσμάτων. Οι χρόνοι επεξεργασίας και εξάρμωσης για τα προβλήματα ήταν τυχαίοι ακέραιοι ομοιόμορφα κατανεμημένοι στο διάστημα [0,199] και [0,49] αντίστοιχα. Η επιλογή μεγάλου διαστήματος τιμών έγινε με σκοπό τη συχνή

παραγωγή περιπτώσεων ανομοιόμορφων χρόνων επεξεργασίας, οι οποίοι, όπως είδαμε στο κεφάλαιο 2, συνεπάγονται νεκρούς χρόνους στις μηχανές και καθυστερήσεις στην εκτέλεση του προγράμματος. Αν και η σχέση 4:1 στο εύρος των τιμών των χρόνων επεξεργασίας και εξάρμωσης ορίζει μια γενική αναλογία, είναι φανερό ότι υπήρξαν περιπτώσεις στις οποίες οι χρόνοι εξάρμωσης ξεπερνούσαν τους αντίστοιχους χρόνους επεξεργασίας. Κατά τη διάρκεια των 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 επιβλήθηκε προκειμένου να μπορούν οι επιδόσεις των αλγορίθμων να συγκριθούν ως προς το κάτω φράγμα του χρόνου εκτέλεσης προγράμματος όπως υπολογίζεται στην παράγραφο 2.2. Παρτιδοποίηση των εργασιών θα σήμαινε σημαντική μείωση του χρόνου εκτέλεσης, χωρίς να υπάρχει τρόπος υπολογισμού κάτω φράγματος στην περίπτωση αυτή.

Στην παράγραφο 7.3.1 συγκρίνουμε τις βελτιώσεις των δύο μηχανισμών που ενσωματώθηκαν στους αλγορίθμους W-Interpol και R-Interpol. Στη συνέχεια, διερευνούμε τη συμπεριφορά του αλγορίθμου WR-Interpol για διάφορες τιμές των παραμέτρων ελέγχου (παράγραφος 7.3.2) και αμέσως μετά μελετούμε συγκριτικά τους W-Interpol, R-Interpol και WR-Interpol με σκοπό να προσδιορίσουμε τον αποδοτικότερο. Στην παράγραφο 7.3.4 παρουσιάζουμε τα αποτελέσματα της επίλυσης προβλημάτων με διαφορετικές κατανομές στους χρόνους επεξεργασίας και εξάρμωσης. Σ' όλες τις περιπτώσεις θεωρούμε ως αλγόριθμο αναφοράς τον Nawaz-Algo, αφού αφενός υλοποιεί τη βασική διαδικασία παρεμβολής και αφετέρου εμφανίζεται με τις καλύτερες επιδόσεις μεταξύ των ευρηματικών αλγορίθμων από προηγούμενες δοκιμές [Lei90, Naw83, WH89].

7.2.1 Σύγκριση των μηχανισμών βελτίωσης

Η πρώτη ομάδα δοκιμών οργανώθηκε με κύρια επιδίωξη την αξιολόγηση των βελτιώσεων που επιτυγχάνουν οι δύο νέοι μηχανισμοί της διαδικασίας παρεμβολής: η αποθήκευση συνόλου μερικών ακολουθιών και η επαναληπτική εκτέλεση με τυχαίες ουρές προγραμματισμού. Ετσι, καθένα από τα τυχαία προβλήματα λύθηκε 9 φορές: με τον Nawaz-Algo, με το W-Interpol για αριθμό αποθηκευόμενων μερικών ακολουθιών W=5,10,20,40 και με τον R-Interpol για αριθμό επαναλήψεων R=5,10,20,40. Για κάθε πρόβλημα μετρήθηκε το σφάλμα του κάθε αλγορίθμου ως προς το κάτω φράγμα για το χρόνο εκτέλεσης, όπως υπολογίζεται από τον τύπο (10) της 2.2.

Πίνακας 1: Απόλυτο σφάλμα για Nawaz-Algo, W-Interpol και R-Interpol.

NxM	NAW	W=5	W=10	W=20	W=40	R=5	R=10	R=20	R=40
6x8	198.1	179.7	178.7	178.5	178.5	183.6	179.8	179.4	178.5
8x8	233.4	211.2	203.9	202.6	198.1	211.1	203.6	201.4	198.7
10x8	223.4	200.8	196.4	193.5	188.6	198.2	197.3	189.3	184.7
20x8	240.5	215.3	203.1	186.9	182.3	209.1	197.1	183.7	179.8
30x8	192.5	157.5	154.4	135.4	142.0	168.1	149.3	141.0	132.0
40x8	196.7	179.1	170.5	167.1	151.5	180.5	160.3	145.9	136.5
50x8	175.0	152.8	151.3	162.9	144.5	172.6	142.5	132.6	126.4
20x4	42.1	40.2	39.3	36.9	37.0	37.9	34.0	32.4	28.2
20x8	240.5	215.3	203.1	186.9	182.3	209.1	197.1	183.7	179.8
20x12	445.6	393.1	378.3	362.0	354.4	398.2	388.0	378.3	362.0
20x16	643.3	572.5	553.9	542.7	528.3	596.4	581.1	560.3	544.8

Στον πίνακα 1 παραθέτουμε τους μέσους όρους σφάλματος των αλγορίθμων για τα 30 προβλήματα που λύθηκαν για κάθε μέγεθος. Στην πρώτη στήλη παρακολουθούμε το μέγεθος του προβλήματος (NxM). Αρχικά, μεταβάλλεται ο αριθμός των εργασιών για 8 μηχανές, ενώ στη συνέχεια μεταβάλλεται ο αριθμός μηχανών για 20 εργασίες. Ετσι, τα μεγέθη προβλημάτων που λύθηκαν είναι: 6x8, 8x8, 10x8, 20x8, 30x8, 40x8, 50x8, 20x4, 20x12, 20x16. Για να παρακολουθούμε ευκολότερα τη διακύμανση του σφάλματος με τις παραμέτρους M και N, τα αποτελέσματα για την κατηγορία προβλημάτων 20x8 επαναλαμβάνονται στις γραμμές 4 και 9. Στη δεύτερη στήλη παρουσιάζεται το σφάλμα του Nawaz-Algo. Στις στήλες 3 έως και 6 δίδεται το σφάλμα του αλγορίθμου W-Interpol για τις διάφορες τιμές του W, ενώ στις στήλες 7 έως 10 το σφάλμα το R-Interpol για τις διάφορες τιμές του R. Παρόμοια έκθεση του μέσου σχετικού σφάλματος του κάθε αλγορίθμου γίνεται στον πίνακα 2. Στον πίνακα 3 φαίνεται η βελτίωση του μέσου σφάλματος των αλγορίθμων ως προς τον Nawaz-Algo.

Από τους παραπάνω πίνακες προκύπτουν καταρχήν μερικά χρήσιμα συμπεράσματα για την απόδοση του Nawaz-Algo. Το σφάλμα του αλγορίθμου παρουσιάζεται σχετικά ανεξάρτητο της μεταβολής του N, ενώ εξαιρετικά ευαίσθητο στη μεταβολή του M. Το σχήμα 12 παρουσιάζει τις διακυμάνσεις του σφάλματος του Nawaz-Algo για τα διάφορα μεγέθη προβλημάτων. Παρατηρούμε ότι όσο ο αριθμός των μηχανών είναι 8, το σφάλμα κινείται σε μια περιοχή του 200.

Πίνακας 2: Σχετικό σφάλμα για Nawaz-Algo, W-Interpol και R-Interpol.

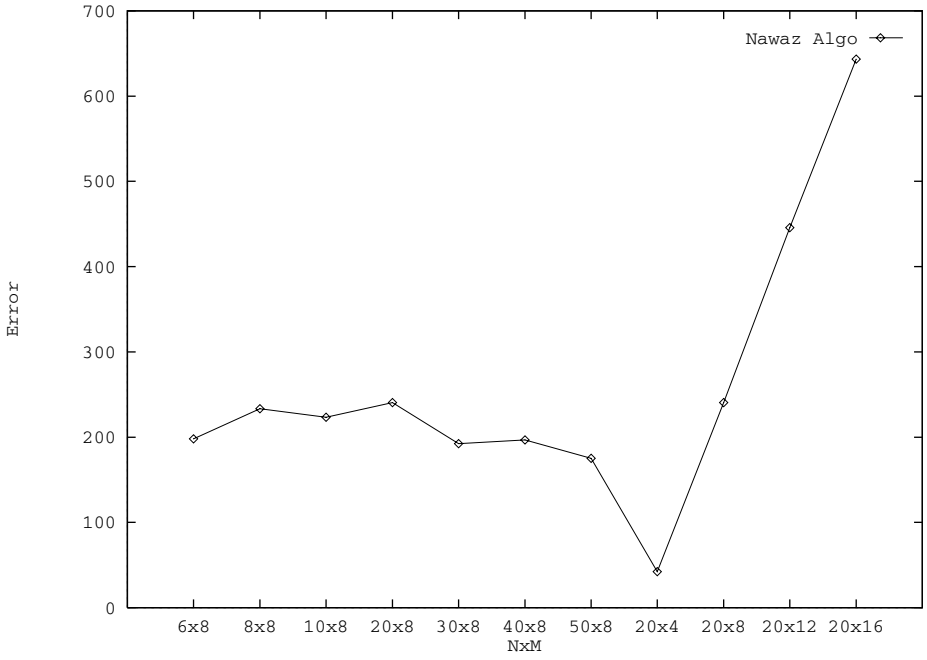
NxM	NAW	W=5	W=10	W=20	W=40	R=5	R=10	R=20	R=40
6x8	14.6	13.3	13.2	13.2	13.2	13.6	13.3	13.2	13.2
8x8	14.5	13.2	12.7	12.6	12.4	13.2	12.7	12.6	12.4
10x8	12.1	10.9	10.7	10.5	10.2	10.8	10.7	10.2	10.0
20x8	7.6	6.8	6.4	5.9	5.7	6.6	6.3	5.8	5.7
30x8	4.4	3.6	3.5	3.1	3.2	3.8	3.4	3.2	3.0
40x8	3.4	3.1	3.0	2.9	2.6	3.1	2.8	2.5	2.3
50x8	2.5	2.2	2.1	2.3	2.0	2.5	2.1	1.9	1.8
20x4	1.5	1.4	1.4	1.3	1.3	1.4	1.2	1.1	1.0
20x8	7.6	6.8	6.4	5.9	5.7	6.6	6.3	5.8	5.7
20x12	12.8	11.3	10.9	10.4	10.2	11.4	11.1	11.0	10.4
20x16	16.4	14.6	14.1	13.9	13.5	15.2	14.9	14.3	13.9

Πίνακας 3: Ποσοστιαία βελτίωση (%) για W-Interpol και R-Interpol ως προς τον Nawaz-Algo.

NxM	W=5	W=10	W=20	W=40	R=5	R=10	R=20	R=40
6x8	9.3	9.8	9.9	9.9	7.3	9.2	9.4	9.8
8x8	9.5	12.6	13.1	15.1	9.5	12.7	13.7	14.8
10x8	10.1	12.1	13.4	15.6	11.2	11.7	15.3	17.3
20x8	10.4	15.5	22.2	24.2	13.1	18.1	23.6	25.2
30x8	18.1	20.0	29.7	26.2	12.7	22.4	26.8	31.4
40x8	8.9	13.3	15.0	23.0	8.2	18.5	25.8	30.6
50x8	12.7	13.5	7.0	17.4	1.4	18.6	24.2	27.7
20x4	4.5	6.7	12.3	12.1	9.9	19.2	23.0	33.0
20x8	10.4	15.5	22.2	24.2	13.1	18.1	23.6	25.2
20x12	11.7	15.1	18.8	20.4	10.6	12.9	15.1	18.7
M.O.	10.6	13.4	16.3	18.7	9.4	15.5	19.4	22.6

Επειδή οι χρόνοι επεξεργασίας των εργασιών είναι τυχαίοι ακέραιοι στο [0,199] με μέση τιμή το 100, μπορούμε να ισχυρισθούμε ότι ο χρόνος εκτέλεσης του προγράμματος απέχει από το κάτω φράγμα του βελτίστου απόσταση που αντιστοιχεί στην εκτέλεση 2 επιπλέον εργασιών, πάντα για M=8. Για αριθμό μηχανών μεγαλύτερο, η τάξη μεγέθους του σφάλματος αλλάζει. Για M=12, το σφάλμα αντιστοιχεί σε 4 περίπου εργασίες, ενώ

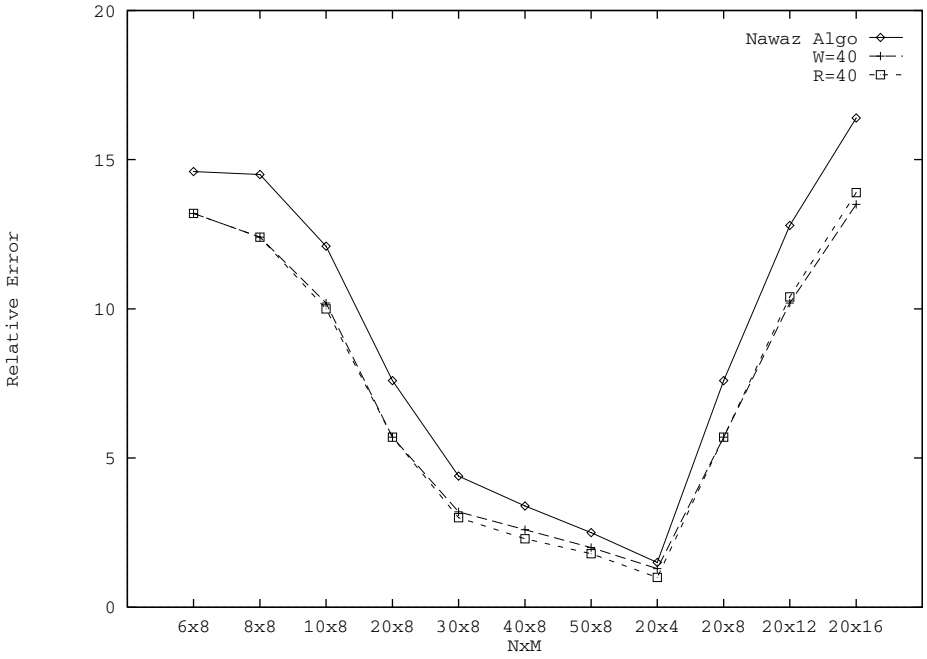
για M=16, σε 6 εργασίες. Αντίθετα, για M=4 το σφάλμα είναι πολύ μικρό (καθυστέρηση εκτέλεσης μισής εργασίας). Επειδή, η διακύμανση αυτή ισχύει σε γενικές γραμμές και για τους αλγορίθμους W-Interpol και R-Interpol, μπορούμε να ισχυρισθούμε ότι κρίσιμη παράμετρος για την ποιότητα των λύσεων των αλγορίθμων παρεμβολής είναι ο αριθμός των μηχανών M και όχι ο αριθμός των εργασιών N. Η υπόθεση αυτή είναι σημαντική και επιβεβαιώνεται από τα αποτελέσματα των υπόλοιπων δοκιμών. Η αιτιολόγησή της πρέπει ν' αναζητηθεί στο γεγονός ότι με την αύξηση του αριθμού μηχανών του συστήματος αυξάνεται και η πιθανότητα εμφάνισης ανομοιόμορφων χρόνων επεξεργασίας και, άρα, νεκρών χρόνων σε περισσότερα από ένα σημεία της αλυσίδας παραγωγής.



Σχήμα 12: Μεταβολή απολύτου σφάλματος Nawaz-Algo με το μέγεθος προβλήματος.

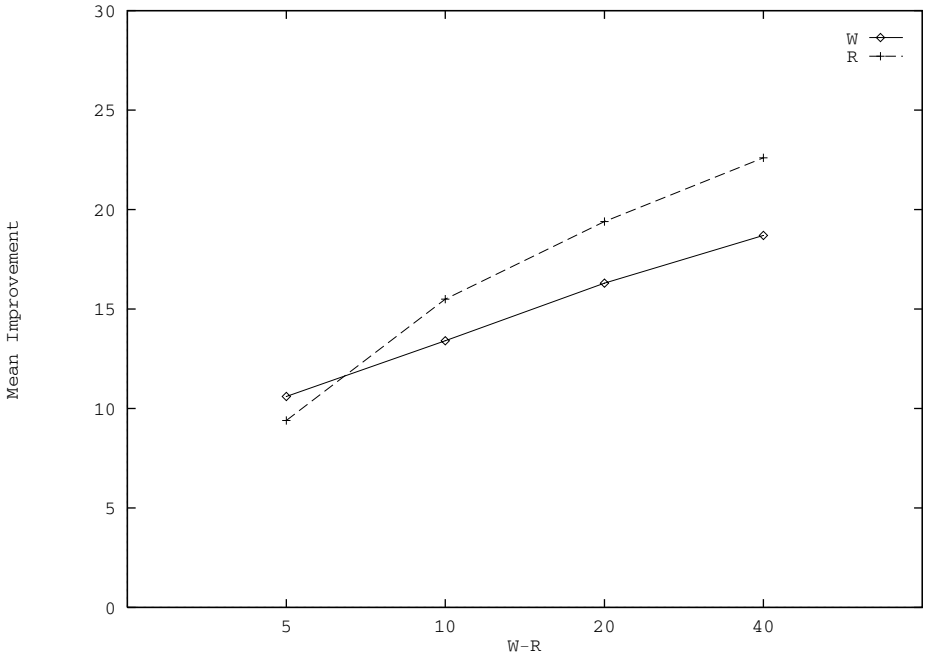
Σε προβλήματα προγραμματισμού εργασιών το σχετικό σφάλμα θεωρείται πάντα χαρακτηριστικότερο κριτήριο επίδοσης από το απόλυτο σφάλμα. Αποτελεί ένα καλό μέτρο της δυσκολίας του προβλήματος, μια και εκφράζει την απόσταση από το βέλτιστο χρόνο εκτέλεσης σε σχέση με την ίδια την τιμή του βελτίστου. Από τον πίνακα 2 παρατηρούμε ότι μεγάλο σχετικό σφάλμα εμφανίζουν τα προβλήματα με μικρό αριθμό εργασιών (6x8, 8x8, 10x8) και εκείνα με μεγάλο αριθμό μηχανών (20x12, 20x16). Για τα προβλήματα με μικρό N, το μεγάλο σχετικό σφάλμα δικαιολογείται από τη σταθερότητα του απολύτου σφάλματος και τη μικρή τιμή του φράγματος για το βέλτιστο χρόνο εκτέλεσης (ο αριθμητής στο κλάσμα υπολογισμού του σχετικού σφάλματος παραμένει σταθερός, ενώ ο παρονομαστής κινείται σε χαμηλά επίπεδα). Για τα προβλήματα με

μεγάλο M η αιτιολόγηση δεν μπορεί να είναι παρόμοια, αφού ο χρόνος εκτέλεσης προγράμματος είναι σημαντικά μεγάλος για πολλά στάδια παραγωγής. Το μεγάλο σχετικό σφάλμα στην περίπτωση αυτή αποτελεί μέτρο της αυξημένης πολυπλοκότητας της συγκεκριμένης κατηγορίας προβλημάτων. Επιχειρώντας μια ταξινόμηση των προβλημάτων σύμφωνα με το σχετικό σφάλμα τους, θα μπορούσαμε να θεωρήσουμε "δύσκολα" τα προβλήματα με μικρό N ή μεγάλο M (6x8, 8x8, 10x8, 20x12, 20x16) και περισσότερο "εύκολα" εκείνα με μεγάλο αριθμό εργασιών και μικρό αριθμό μηχανών (10x4, 20x8, 30x8, 40x8, 50x8). Στις επόμενες παραγράφους θα επιβεβαιώσουμε ότι το σχετικό σφάλμα για τους αλγορίθμους παρεμβολής μεταβάλλεται σε γενικές γραμμές όπως δείχνει το σχήμα 13.



Σχήμα 13: Συγκριτική παράσταση σχετικού σφάλματος των Nawaz-Algo, W-Interpol με W=40 και R-Interpol με R=40.

Το σχήμα 13 περιέχει επίσης τα σχετικά σφάλματα του αλγορίθμου W-Interpol με W=40 και R-Interpol με R=40. Παρατηρούμε ότι ο R-Interpol παρουσιάζει λίγο μικρότερα σχετικά σφάλματα από τον W-Interpol. Ο πίνακας 2 αποδεικνύει ότι αυτό ισχύει γενικά αν συγκρίνουμε τους δύο αλγορίθμους για ίσες τιμές των παραμέτρων R και W. Γενικά, και οι δύο μηχανισμοί επιτυγχάνουν μείωση του σχετικού σφάλματος κατά 1 έως 2 ποσοστιαίες μονάδες.



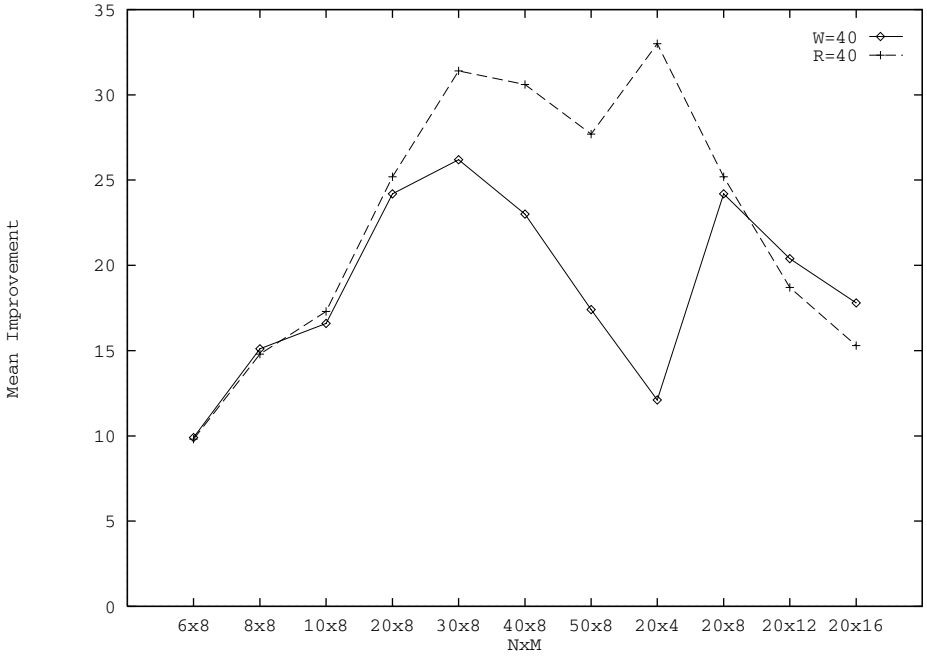
Σχήμα 14: Αύξηση της μέσης σχετικής βελτίωσης των W-Interpol και R-Interpol με μεταβολή των παραμέτρων W και R

Η ελαφρά υπεροχή του R-Interpol έναντι του W-Interpol αποδεικνύεται από τις σχετικές με τον Nawaz-Algo βελτιώσεις που επιτυγχάνουν (πίνακας 3). Κατά κανόνα για ίσες τιμές παραμέτρων W και R η επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού παρουσιάζεται αποδοτικότερη από την αποθήκευση συνόλου μερικών διατάξεων ανά στάδιο της παρεμβολής. Το σχήμα 14 παριστάνει συγκριτικά τη μέση βελτίωση για τις διάφορες τιμές των W και R. Εκτός από την περίπτωση W=5 και R=5, η μέση βελτίωση που πέτυχε ο R-Interpol είναι σαφώς μεγαλύτερη. Προκειμένου να παρακολουθήσουμε γραφικά τη διακύμανση της βελτίωσης και για τους δύο μηχανισμούς, παραθέτουμε συγκριτικά τις επιδόσεις των αλγορίθμων για W=40 και R=40 στο σχήμα 15. Η σχετική βελτίωση ως προς τον Nawaz-Algo εμφανίζεται μεγαλύτερη για τα ”εύκολα” προβλήματα (30x8, 40x8, 50x8, 20x4) και φτάνει μέχρι και 33%, ενώ αρκετά μικρότερη για τα υπόλοιπα (10% έως 20%).

7.2.2 Σύγχρονη λειτουργία των δύο μηχανισμών

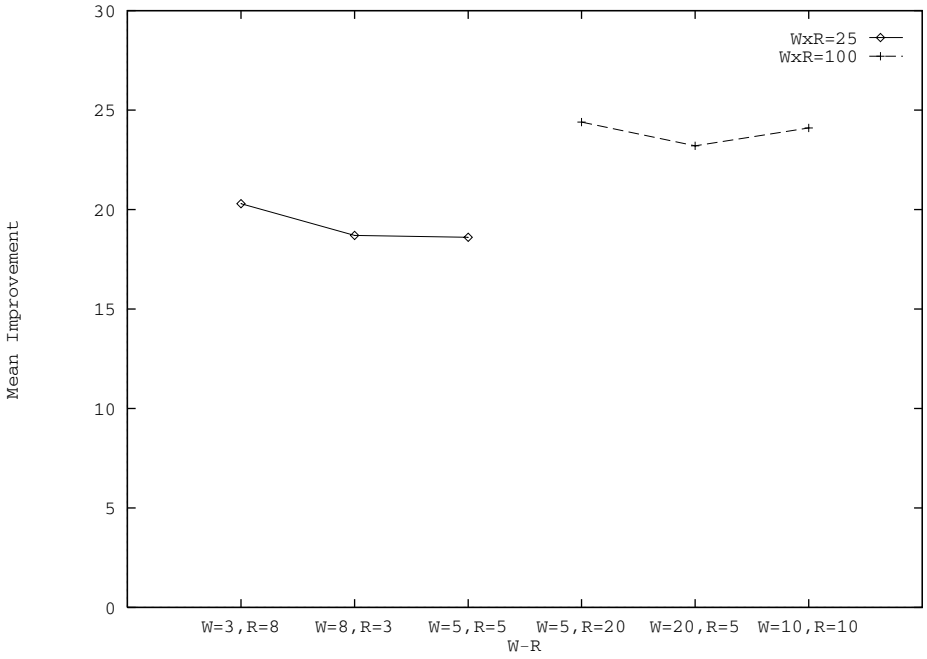
Η δεύτερη ομάδα δοκιμών διερεύνησε τον αλγόριθμο WR-Interpol για τις διάφορες τιμές των παραμέτρων ελέγχου W και R για να προσδιορίσει καταρχήν τα αποδοτικά μεγέθη τους και κατά δεύτερο λόγο την αναλογία τιμών των δύο παραμέτρων. Δημιουργήθηκαν

και λύθηκαν προβλήματα μεγέθους: 6x8, 8x8, 10x8, 20x8 30x8, 40x8, 50x8, 10x4, 10x12, 10x16 (30 προβλήματα ανά μέγεθος). Κάθε πρόβλημα λύθηκε με τον Nawaz-Algo και WR-Interpol για τα παρακάτω ζεύγη τιμών (W,R): (3,8)-(8,3)-(5,5) και (5,20)-(20,5)-(10,10). Για τα πρώτα 3 ζεύγη τιμών ισχύει WxR=25 περίπου, ενώ για τα επόμενα 3 ζεύγη WxR=100. Μ' αυτόν τον τρόπο η σύγκριση των αποτελεσμάτων γίνεται μεταξύ ζευγών τιμών με το ίδιο γινόμενο WxR για τον προσδιορισμό της κρισιμότερης παραμέτρου ελέγου και μεταξύ των δύο ομάδων για αξιολόγηση των βελτιώσεων με τη μεταβολή της τάξης μεγέθους των παραμέτρων.

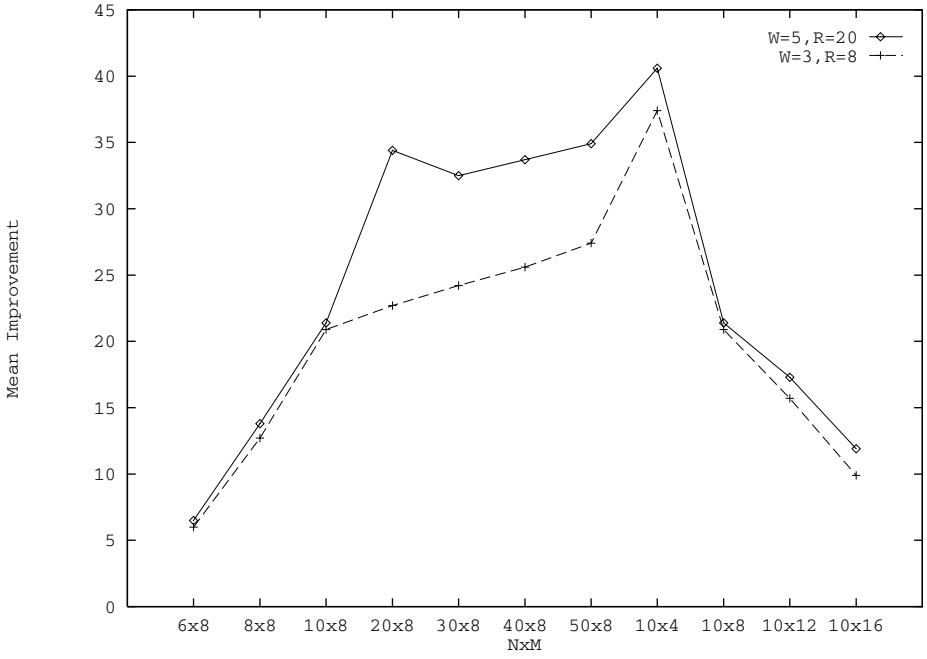


Σχήμα 15: Ποσοστιαία βελτίωση για W-Interpol με W=40 και R-Interpol με R=40

Οι πίνακες με το απόλυτο και σχετικό σφάλμα του αλγορίθμου για τα διάφορα μεγέθη προβλημάτων και παραμέτρων, καθώς και τη σχετική με τον Nawaz-Algo βελτίωση που επιτυγχάνει, παρατίθενται στο παράρτημα Α. Από τους πίνακες αυτούς επιβεβαιώνουμε καταρχήν τα συμπεράσματα της προηγούμενης παραγράφου γύρω από την απόδοση των αλγορίθμων παρεμβολής. Το απόλυτο σφάλμα εμφανίζεται ανεξάρτητο του Ν και μεταβάλλεται με το Μ. Για Μ=8 αντιστοιχεί κατά κανόνα σε εκτέλεση δύο επιπλέον εργασιών, ενώ για Μ=4, Μ=12 και Μ=16 είναι της τάξης της μισής, των τεσσάρων και των έξι εργασιών αντίστοιχα. Το σχετικό σφάλμα παρουσιάζεται σημαντικό για τα προβλήματα λίγων εργασιών ή πολλών μηχανών (6x8, 8x8, 10x8, 10x12, 10x16), και μικρό για τα υπόλοιπα (20x8, 30x8, 40x8, 50x8).



Σχήμα 16: Μέση βελτίωση του WR-Interpol για τις διάφορες τιμές των παραμέτρων ελέγχου



Σχήμα 17: Διακύμανση της ποσοστιαίας βελτίωσης του WR-Interpol

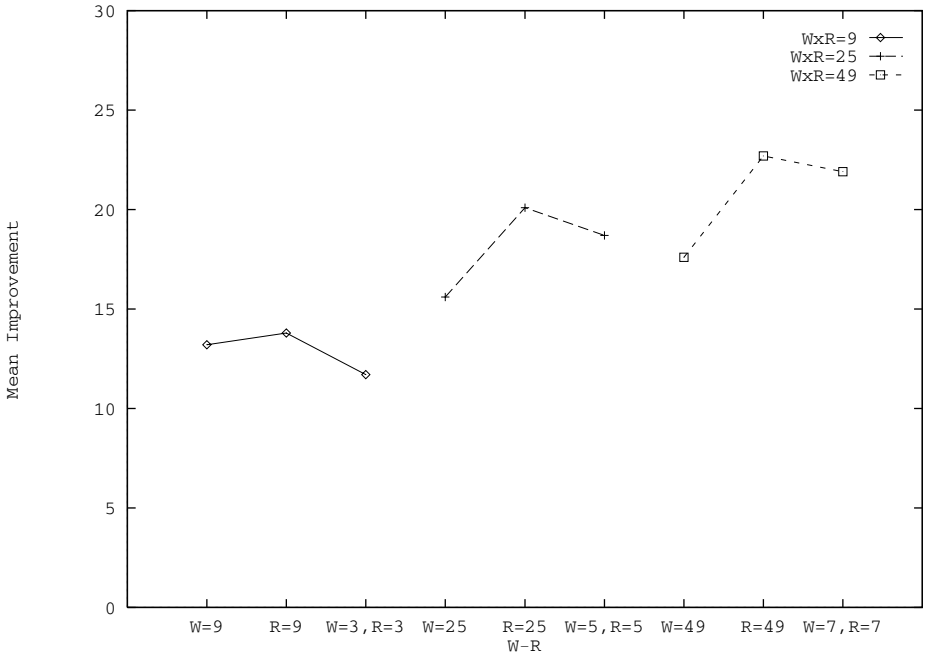
Οι βελτιώσεις που πέτυχε ο αλγόριθμος φαίνονται στο σχήμα 16. Η μέση βελτίωση πάνω σε όλα τα μεγέθη προβλημάτων που λύθηκαν ήταν της τάξης του 20% για WxR=25

και της τάξης του 25% για WxR=100. Παρατηρούμε επίσης ελαφρά υπεροχή για τα ζεύγη τιμών (W,R) στα οποία το R είναι μεγαλύτερο ((3,8),(5,20)). Η υπεροχή αυτή, αν και δεν είναι συντριπτική, επιβεβαιώνει ότι ο μηχανισμός επαναληπτικής παρεμβολής με τυχαίες ουρές προγραμματισμού παράγει αποδοτικότερες λύσεις από το μηχανισμό αποθήκευσης συνόλου μερικών διατάξεων. Στο σχήμα 17 παρακολουθούμε τη μεταβολή της βελτίωσης με το μέγεθος των προβλημάτων για ζεύγη τιμών των παραμέτρων ελέγχου (3,8) και (5,20). Η βελτίωση φτάνει μέχρι και 40% για μεγάλο N, ενώ είναι σημαντικά μειωμένη για μικρό N ή μεγάλο M. Αυτό συμβαδίζει με τα αποτέλεσματα της προηγούμενης παραγράφου, όπου η βελτίωση για τα διάφορα μεγέθη προβλημάτων ακολουθούσε παρόμοια πορεία. Επιπλέον, ουσιαστική διαφοροποίηση των επιδόσεων του WR-Interpol για τα ζεύγη παραμέτρων (3,8) και (5,20) παρατηρείται μόνο στα προβλήματα με μεγάλο N.

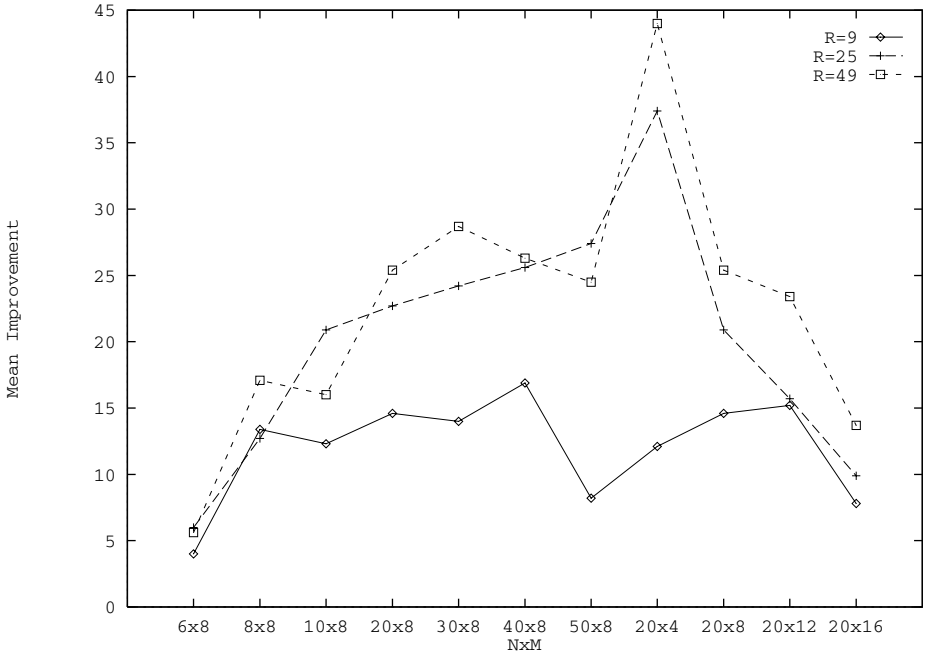
7.2.3 Υπεροχή του μηχανισμού επαναληπτικής παρεμβολής

Προκειμένου να διασταυρώσουμε την υπόθεση ότι η επαναληπτική παρεμβολή με τυχαίες ουρές προγραμματισμού οδηγεί σε καλύτερα αποτελέσματα, οργανώσαμε την τρίτη ομάδα δοκιμών. Προβλήματα μεγέθους 6x8, 8x8, 10x8, 20x8, 30x8, 40x8, 50x8, 20x4, 20x12, 20x16 λύθηκαν με τους αλγορίθμους Nawaz-Algo, W-Interpol με W=9,25,49, R-Interpol με R=9,25,49 και WR-Interpol με (W,R)=(3,3),(5,5),(7,7). Οι πίνακες με το απόλυτο και σχετικό σφάλμα για τους αλγορίθμους βρίσκονται στο παράρτημα Α. Στο ίδιο σημείο παραθέτουμε και τις σχετικές βελτιώσεις των αλγορίθμων ως προς τον αλγόριθμο Nawaz.

Οι δοκιμές επιβεβαίωσαν την εξάρτηση του απολύτου σφάλματος από τον αριθμό των μηχανών, το μεγάλο σχετικό σφάλμα για προβλήματα με λίγες εργασίες ή πολλές μηχανές και την υπεροχή του αλγορίθμου R-Interpol στις περισσότερες περιπτώσεις. Στο σχήμα 18 φαίνεται η μέση βελτίωση (πάνω σε όλα τα μεγέθη προβλημάτων) για κάθενα από τους αλγορίθμους και τις διάφορες τιμές των W και R. Γενικά, η βελτίωση για WxR=9 είναι γύρω στο 13% για WxR=25, στο 19% και για WxR=49 στο 22%. Παρατηρούμε ότι σε κάθεμια από τις 3 ομάδες τιμών γινομένου WxR την καλύτερη βελτίωση παρουσιάζει ο R-Interpol, ακολουθεί πολύ κοντά ο WR-Interpol, ενώ ο W-Interpol απέχει σημαντικά από τους προηγούμενους για μεγάλες τιμές των W και R. Η βελτίωση σε συνάρτηση με το μέγεθος προβλήματος δίδεται στο σχήμα 19 για τον αλγόριθμο R-Interpol με R=9,25,49. Ενώ οι βελτιώσεις διαφέρουν κατά πολύ μεταξύ R=9 και R=25, ειδικά όταν ο αριθμός των εργασιών είναι μεγάλος και ο αριθμός των μηχανών μικρός, δεν υπάρχει αντίστοιχη διαφορά μεταξύ R=25 και R=49. Η παρατήρηση αυτή ισχύει γενικά για τη σχετική βελτίωση που καταγράφηκε και στις προηγούμενες ομάδες δοκιμών.



Σχήμα 18: Μεταβολή της μέσης βελτίωσης με το γινόμενο WxR



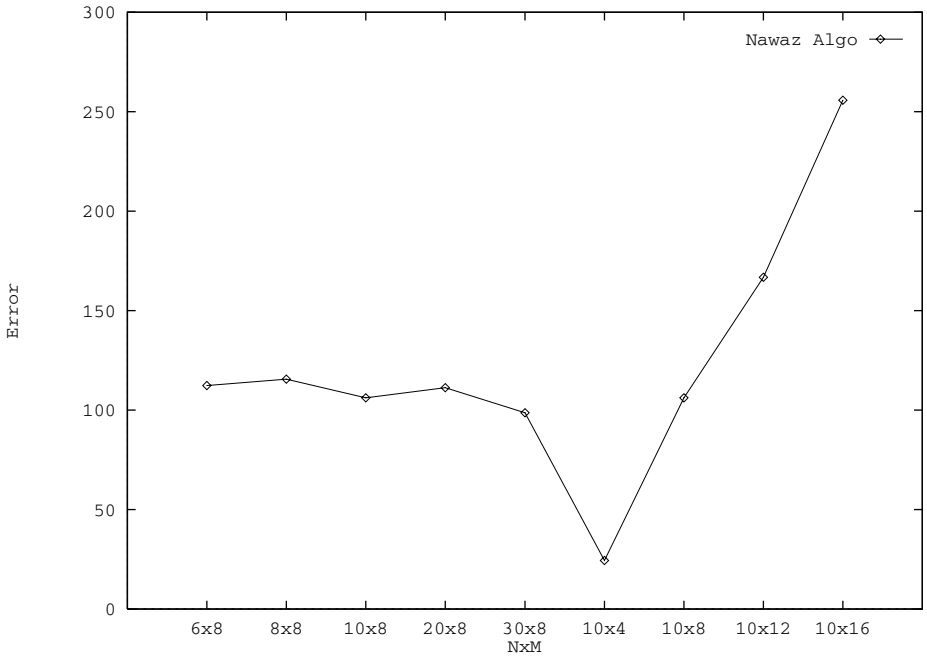
Σχήμα 19: Διακύμανση της ποσοστιαίας βελτίωσης του R-Interpol (αποδοτικότερος αλγόριθμος) για R=9,25,49.

Για τιμές παραμέτρων με WxR=25 και μεγαλύτερο, η βελτίωση δεν αυξάνεται με το γρήγορο ρυθμό που ακολουθούσε μέχρι το σημείο αυτό, αν και υπάρχει αξιοπρόσεχτη

διαφοροποίηση (στην 7.2.2 για WxR=25 η μέση βελτίωση ήταν γύρω στο 20%, ενώ για WxR=100 μεταφέρθηκε στο 25%).

7.2.4 Αλλαγή κατανομών

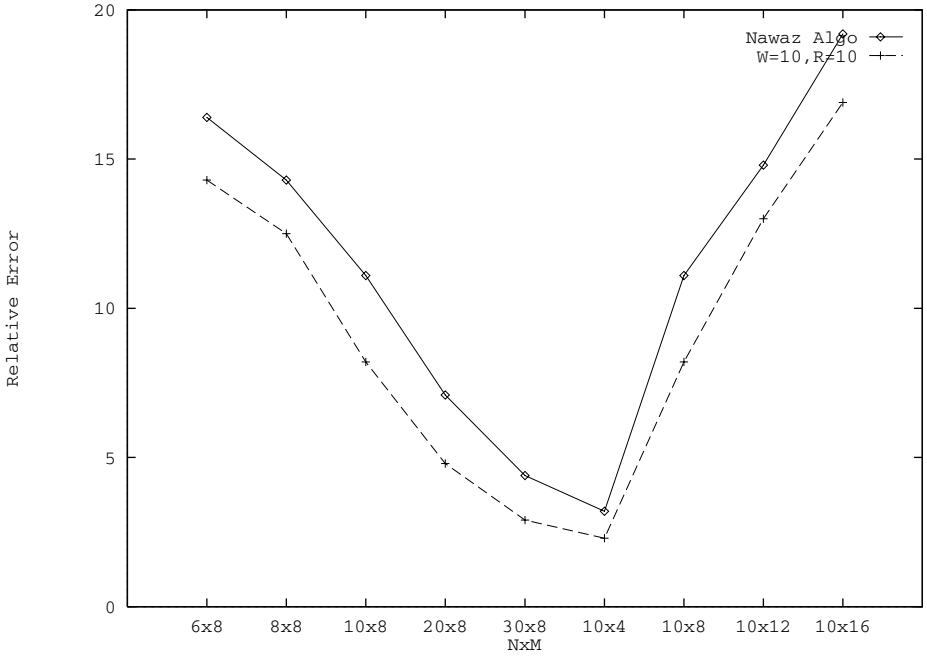
Στην τέταρτη ομάδα δοκιμών οι χρόνοι επεξεργασίας και εξάρμωσης ήταν τυχαίοι ακέραιοι ομοιόμορφα κατανεμημένοι στο διάστημα [0,99] και [0,24] αντίστοιχα. Η γενική αναλογία 4:1 συντηρείται, με αλλαγή όμως των ορίων των διαστημάτων. Για καθε μέγεθος προβλήματος (6x8, 8x8, 10x8, 20x8, 30x8, 10x4, 10x8, 10x12, 10x16) λύθηκαν 30 προβλήματα με τους αλγορίθμους Nawaz-Algo και WR-Interpol με (W,R) να παίρνουν τιμές (3,8)-(8,3)-(5,5) και (5,20)-(20,5)-(10,10).



Σχήμα 20: Απόλυτο σφάλμα αλγορίθμων για χρόνους επεξεργασίας στο [0,99].

Οι πίνακες με το απόλυτο και σχετικό σφάλμα παρατίθενται στο παράρτημα Α. Το μέγεθος του απολύτου σφάλματος παραμένει ανεξάρτητο του αριθμού εργασιών και μεταβάλλεται με τον αριθμό των μηχανών (το σχήμα 20 παριστάνει γραφικά τη μεταβολή του σφάλματος με το μέγεθος προβλήματος). Θεωρώντας ως μέσο χρόνο κατεργασίας το 50, η τάξη μεγέθους του σφάλματος αντιστοιχεί σε μισή εργασία για M=4, σε δύο εργασίες για M=8, σε τρεις εργασίες για M=12 και σε πέντε εργασίες για M=16. Η παραπάνω παρατήρηση συμβαδίζει σε γενικές γραμμές με τα αποτελέσματα της 7.2.1. Αντιστοιχία εμφανίζεται και στο σχετικό σφάλμα των αλγορίθμων. Το σχήμα 21

αποδεικνύει ότι η διακύμανση του σχετικού σφάλματος με το μέγεθος προβλήματος είναι παρόμοια μ' αυτές που παρουσίασαν οι προηγούμενες ομάδες δοκιμών με διαφορετική κατανομή χρόνων επεξεργασίας και εξάρμωσης. Στο ίδιο σχήμα παρακολουθούμε και τη μείωση του σχετικού σφάλματος από τον Nawaz-Algo στον WR-Interpol με (W,R)=(10,10), η οποία είναι της τάξης των δύο ποσοστιαίων μονάδων. Ο πίνακας των σχετικών βελτιώσεων ως προς τον Nawaz-Algo δίδεται επίσης στο παράρτημα Α και επιβεβαιώνει τα συμπεράσματα των προηγούμενων παραγράφων.



Σχήμα 21: Μετά την αλλαγή κατανομών η μείωση του σχετικού σφάλματος παρέμεινε στο επίπεδο του 2%.

7.3 Δοκιμές για το $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$

Η διερεύνηση του προβλήματος του δικριτηριακού προγραμματισμού εργασιών με προθεσμίες όπου η ελαχιστοποίηση της μέγιστης καθυστέρησης προέχει έναντι της ελαχιστοποίησης του συνολικού χρόνου εκτέλεσης, πραγματοποιήθηκε με εντελώς διαφορετικά κριτήρια. Επειδή στη βιβλιογραφία δεν αναφέρεται αλγόριθμος για το πρόβλημα, η διαδικασία εφικτής παρεμβολής που προτείναμε στην εργασία δεν ήταν δυνατό να αξιολογηθεί ως προς την ποιότητα των λύσεων που παράγει. Παρακολουθήσαμε απλώς την απόδοσή της σε σχέση με τις παραμέτρους του προβλήματος και τη βελτίωση που πέτυχαν οι μηχανισμοί αποθήκευσης συνόλου μερικών διατάξεων και επαναληπτικής παρεμβολής των ΕΧΠ.

Τα μεταβλητά μεγέθη των δοκιμών ήταν συνολικά 5:

1. N: ο αριθμός των εργασιών προς προγραμματισμό.
2. M: ο αριθμός των μηχανών του συστήματος.
3. W: ο αριθμός των αποθηκευόμενων ανά στάδιο μερικών διατάξεων.
4. R: ο αριθμός των επαναλήψεων της φάσης παρεμβολής των ΕΧΠ.
5. ddper: το ποσοστό των εργασιών με προθεσμίες επί του συνόλου των εργασιών.

Η παράμετρος ddper ήλεγχε την αναλογία εργασιών υψηλής προτεραιότητας (ΕΜΠ) και εργασιών χωρίς περιορισμούς στο χρόνο περάτωσης (ΕΧΠ). Προβλήματα με μεγάλο ποσοστό ΕΜΠ αναμένονται σαφώς δυσκολότερα, μια και τα περιθώρια της διαδικασίας εφικτής παρεμβολής για εφικτές ή ικανοποιητικές λύσεις στενεύουν κατά πολύ. Οι τιμές που δόθηκαν στην παράμετρο ddper κατά τις δοκιμές ήταν 50%, 60%, 80% και 100%. Για κάθεμιά από τις 4 ομάδες δοκιμών που δημιουργήθηκαν, λύθηκαν προβλήματα μεγέθους NxM: 6x8, 8x8, 10x4, 10x8, 10x12, 10x16, 20x4, 20x8, 20x12, 20x16, 30x8 και 40x8 (30 προβλήματα για κάθε μέγεθος). Οι χρόνοι επεξεργασίας και εξάρμωσης ήταν τυχαίοι ακέραιοι ομοιόμορφα κατανεμημένοι στο διάστημα [0,199] και [0,49] αντίστοιχα. Σε κάθε εργασία δίδονταν προθεσμία παράδοσης με πιθανότητα ίση με το ddper για την ομάδα δοκιμών. Η προθεσμία εργασίας i που ανήκε στην κατηγορία ΕΜΠ ήταν τυχαίος ακέραιος που ακολουθούσε ομοιόμορφη κατανομή στο διάστημα $[TotP_i, LB]$, όπου $Totp_i = \sum_{j=1} M(p_{ij} + s_{ij})$ (εκφράζει το συντομότερο χρόνο περάτωσης της εργασίας στο σύστημα) και LB κάτω φράγμα για το συνολικό χρόνο εκτέλεσης ολόκληρου του προγράμματος. Οπως αποδείχθηκε από τις δοκιμές οι προθεσμίες των εργασιών ήταν εξαιρετικά "σφικτές", μια και ο χρόνος εκτέλεσης προγράμματος απήχε σημαντικά από το κάτω φράγμα LB.

Κάθε πρόβλημα λύθηκε από τους αλγορίθμους: Feas-Interpol, W-Feas-Interpol με W=25,100 και WR-Feas-Interpol με (W,R)=(8,3),(3,8),(5,5) και (20,5),(5,20),(10,10). Προκειμένου να εκτιμηθεί η βελτίωση του κάθε αλγορίθμου σε σχέση με τον Feas-Interpol, που αποτελεί την αρχική διαδικασία εφικτής παρεμβολής χωρίς κανένα μηχανισμό βελτίωσης, μετρήθηκαν τα παρακάτω:

- ΕΛ: αριθμός προβλημάτων για τα οποία ο αλγόριθμος εντόπισε εφικτή λύση (επί συνόλου 30 προβλημάτων).

- ΒΚ: αριθμός προβλημάτων για τα οποία ο αλγόριθμος εντόπισε μη εφικτή λύση με μικρότερη καθυστέρηση από αυτήν της μη εφικτής λύσης που παρήγαγε ο Feas-Interpol για το ίδιο πρόβλημα.
- ΠΒΧ: ποσοστό βελτίωσης (%) του χρόνου εκτέλεσης του τελικού προγράμματος του αλγορίθμου ως προς το χρόνο εκτέλεσης εκείνου του Feas-Interpol.

Είναι φανερό ότι επειδή η επαναληπτική παρεμβολή ΕΧΠ με τυχαίες ουρές προγραμματισμού εφαρμόζεται κατά τη δεύτερη φάση των αλγορίθμων, αποκλείεται να επηρεάσει το εφικτό των λύσεων ή τις καθυστερήσεις που εμφανίζονται. Αρα, η συζήτηση γύρω από τις μετρήσεις των ΕΛ και ΒΚ αφορά μόνο την τιμή της παραμέτρου W του μηχανισμού αποθήκευσης συνόλου μερικών διατάξεων ανά στάδιο. Οι δοκιμές των αλγορίθμων έδειξαν καταρχήν ότι τα ΕΛ και ΒΚ των αλγορίθμων δεν ακολουθούν τη μεταβολή του W. Στις περισσότερες περιπτώσεις, όταν υπήρχαν περιθώρια για τον εντοπισμό λύσεων με μικρότερη καθυστέρηση, αυτό έγινε και με μικρές τιμές του W. Αντίθετα, για εργασίες με πολύ "σφικτές" προθεσμίες δεν ήταν δυνατή η βελτίωσή της τελικής καθυστέρησης ακόμα και για μεγάλες τιμές του W. Ετσι, θεωρήσαμε περιτό να εκθέσουμε τις μετρήσεις για όλες τις τιμές των παραμέτρων και περιοριζόμαστε στα αποτελέσματα των αλγορίθμων Feas-Interpol, W-Interpol για W=100 και WR-Interpol για (W,R)=(20,5),(5,20) και (10,10).

Ενδεικτικά, θα ασχοληθούμε με την ομάδα δοκιμών για ddper=60%. Ο πίνακας 4 δείχνει τα αποτελέσματα για την περίπτωση αυτή, ενώ οι αντίστοιχοι πίνακες για ddper=50%, ddper=80% και ddper=100% δίνονται στο παράρτημα Β. Στην πρώτη στήλη του πίνακα 4 παρακολουθούμε το μέγεθος προβλήματος και στη δεύτερη το ΕΛ για τον Nawaz-Algo. Οι 3 επόμενες στήλες αντιστοιχούν στα ΕΛ, ΒΚ και ΠΒΧ για τον Feas-Interpol με W=100. Ακολουθούν με παρόμοιο τρόπο οι μετρήσεις για τον WR-Feas-Interpol με (W,R)=(20,5),(5,20) και (10,10). Για παράδειγμα, στη γραμμή 20x16 καταγράφονται τα εξής: ο Feas-Interpol παρήγαγε 9 φορές εφικτή λύση σε σύνολο 30 προβλημάτων. Ο W-Interpol με W=100 εντόπισε εφικτή λύση 2 φορές παραπάνω (συνολικά 11) και από τις υπόλοιπες 19 περιπτώσεις κατάφερε να βελτιώσει το $T_m max$ του Feas-Interpol 13 φορές, με παράλληλη μέση μείωση του χρόνου εκτέλεσης προγράμματος κατά 9.4%. Για τα ίδια προβλήματα, ο WR-Feas-Interpol με (W,R)=(20,5) εντόπισε 10 φορές εφικτή λύση, από τις υπόλοιπες 20 περιπτώσεις οδηγήθηκε σε μικρότερο T_{max} τις 13 φορές με μέσο χρόνο εκτέλεσης βελτιωμένο κατά 8.7%. Αντίστοιχα ισχύουν και για τις επόμενες 6 στήλες του WR-Feas-Interpol με (W,R)=(5,20) και (10,10). Στη γραμμή αυτή φαίνεται ότι ο W-Interpol με W=100 παρουσιάζει αναμφισβήτητα τα καλύτερα αποτελέσματα, έστω

κι αν οι διαφορές είναι μικρές. Οι μικρότερες βελτιώσεις που εμφανίζονται είναι για τον WR-Feas-Interpol με (W,R)=(5,20). Μελετώντας ολόκληρο τον πίνακα, αλλά και τους υπόλοιπους του παραρτήματος Β, παρατηρούμε ότι η υπεροχή του W-Feas-Interpol είναι φανερή τόσο ως προς το T_{max}, όσο και ως προς το C_{max}.

Πίνακας 4: Επιδόσεις αλγορίθμων για ποσοστό ΕΜΠ 60

	NAW	W=100			W=20 R=5			W=5 R=20			W=10 R=10		
NxM	ΕΛ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ
6x8	7	7	1	1.0	7	1	1.0	7	1	1.0	7	1	1.0
8x8	9	10	1	3.4	10	1	3.4	10	1	3.4	10	1	3.4
10x4	16	18	2	14.2	18	2	14.2	17	2	12.5	18	2	14.2
10x8	9	9	8	5.7	9	8	5.5	9	8	5.5	9	8	5.5
10x12	8	10	4	4.7	10	4	4.7	10	4	4.6	10	4	4.7
10x16	5	7	9	0.5	7	9	0.5	7	9	0.6	7	9	0.5
20x4	20	20	1	13.7	20	1	25.6	20	1	19.1	20	1	22.0
20x8	19	20	3	16.9	20	3	17.4	20	2	14.9	20	2	14.8
20x12	9	12	9	13.1	12	9	11.5	11	9	8.9	11	9	10.8
20x16	9	11	13	9.4	10	13	8.7	10	11	6.1	10	13	7.3
30x8	17	21	7	19.5	21	7	24.5	21	7	20.9	21	7	22.4
40x8	19	20	7	19.4	20	7	18.8	20	6	17.8	20	7	21.4

Η διαφορά δικαιολογείται από το γεγονός ότι ο μηχανισμός αποθήκευσης συνόλου μερικών διατάξεων βελτιώνει τόσο τη φάση προγραμματισμού ΕΜΠ, όσο και τη φάση προγραμματισμού ΕΧΠ. Ειδικά για μεγάλο ποσοστό ΕΜΠ (ddper=80%, ddper=100%), η δεύτερη φάση προγραμματίζει πολύ μικρό αριθμό εργασιών και δεν μπορεί να επηρεάσει ουσιαστικά τις επιδόσεις του αλγορίθμου. Αντίθετα, για μικρό ποσοστό εργασιών (ddper=50%) παρατηρήθηκε ουσιαστική βελτίωση του ΠΒΧ για μεγάλες τιμές του R στον αλγόριθμο WR-Interpol.

Κατά τη μεταβολή του ddper, ο αριθμός των περιπτώσεων που οι αλγόριθμοι εντόπισαν εφικτή λύση παρουσίασε σημαντική διακύμανση. Με τον Feas-Interpol, για ddper=50% ο μέσος όρος για το ΕΛ ήταν 18.6, για ddper=60% έπεσε στο 12.3, για ddper=80% στο 3.5 και για ddper=100% δεν εντοπίστηκαν εφικτές λύσεις σε καμιά περίπτωση. Ο λόγος γι' αυτό ήταν οι πολύ σφικτές προθεσμίες παράδοσης των εργασιών. Οπως εξηγήσαμε για τις κατανομές των δεδομένων, καμιά προθεσμία δεν ξεπερνούσε το LB για το αντίστοιχο πρόβλημα, ενώ ο χρόνος εκτέλεσης προγράμματος ήταν πολύ μεγαλύτερος.

Ο προγραμματισμός με πρωτεύον κριτήριο το T_{max} οδήγησε σε χρόνους εκτέλεσης που απήχαν σημαντικά από τους αντίστοιχους των λύσεων των αλγορίθμων παρεμβολής όταν μοναδικό κριτήριο ήταν το C_{max}.

Σημαντικό χαρακτηριστικό της διαδικασίας εφικτής παρεμβολής είναι η κακή απόδοσή της όταν οι πρώτες προθεσμίες παράδοσης είναι υπερβολικά σφικτές. Σε περιπτώσεις στις οποίες οι εργασίες που βρίσκονταν στην αρχή της ουράς προγραμματισμού δεν περατώνονταν έγκαιρα, χάναμε πολύ γρήγορα το εφικτό του όλου προγράμματος και η διάταξη των υπόλοιπων ΕΜΠ στην τελική λύση γινόταν κατά αύξουσες προθεσμίες (EDD). Αυτό συνεπάγεται ουσιαστικά εμπλοκή της διαδικασίας και παύση της μεθόδου προγραμματισμού με παρεμβολή. Το αποτέλεσμα ήταν συχνά κακή απόδοση, όχι μόνο ως προς το κριτήριο C_{max}, αλλά και ως προς το κριτήριο T_{max}, μια και η διάταξη κατά αύξουσες προθεσμίες δεν εγκειάται ελαχιστοποίησή του σ' όλες τις περιπτώσεις. Μια αναθεώρηση των αλγορίθμων για τις ειδικές περιπτώσεις γρήγορης απώλειας του εφικτού ενδέχεται να οδηγούσε σε καλύτερες επιδόσεις. Το ενδεχόμενο θα συζητηθεί στο επόμενο κεφάλαιο, μεταξύ των προτεινόμενων επεκτάσεων.

7.4 Υπολογιστικός χρόνος

Ο αλγόριθμος του Nawaz συγκαταλέγεται στις ευρηματικές μεθόδους για το πρόβλημα ροϊκής παραγωγής και ο απαιτούμενος υπολογιστικός χρόνος για την εκτέλεσή του είναι πολύ μικρός. Οι νέοι αλγόριθμοι που παρουσιάσαμε, αν και προγραμματίζουν τις εργασίες με τη μέθοδο της παρεμβολής, χρησιμοποιούν τους μηχανισμούς αποθήκευσης συνόλου μερικών διατάξεων και επαναληπτικής εκτέλεσης της παρεμβολής με τυχαίες ουρές προγραμματισμού, οι οποίοι ανεβάζουν αισθητά τον απαιτούμενο υπολογιστικό χρόνο.

Στους αλγορίθμους W-Interpol, R-Interpol και WR-Interpol ο συνολικός αριθμός μερικών διατάξεων που εξετάζονται είναι περίπου ανάλογος των W, R και WxR αντίστοιχα. Κατ' επέκταση, το ίδιο μπορούμε να υποθέσουμε και για τον απαιτούμενο χρόνο εκτέλεσής τους. Στον πίνακα 5 παραθέτουμε ενδεικτικά μετρήσεις που επιβεβαιώνουν την υπόθεση αυτή. Οι μετρήσεις πραγματοποιήθηκαν σε SPARCstation ELC, ενώ οι αλγόριθμοι είχαν υλοποιηθεί σε γλώσσα προγραμματισμού C και είχαν μεταγλωτιστεί με βελτιστοποίηση κώδικα. Η αύξηση του υπολογιστικού χρόνου παρουσιάζεται ανάλογη του γινομένου WxR για τον WR-Interpol και εντελώς αντίστοιχη για τους W-Interpol και R-Interpol για τις ίδιες τιμές των W και R. Είναι φανερή επίσης η

δυνατότητα για παραπάνω αύξηση των τιμών των παραμέτρων, ειδικά για προβλήματα λίγων εργασιών. Για μεγάλα προβλήματα όμως (40 και 50 εργασίες) ο υπολογιστικός χρόνος ενδέχεται να μεγαλώσει υπερβολικά και να ξεπεράσει το λεπτό σε περίπτωση σημαντικής αύξησης των τιμών των W και R. Γι' αυτόν τον λόγο οι δοκιμές που παρουσιάσαμε στο κεφάλαιο αυτό διερεύνησαν τη συμπεριφορά των αλγορίθμων για μικρές τιμές των παραμέτρων ελέγχου.

Πίνακας 5: Υπολογιστικός χρόνος (CPU secs) εκτέλεσης των αλγορίθμων.

Αλγόριθμος	Παράμετροι	10x8	20x8	30x8	40x8
W-Interpol	W=10	0.0	0.4	1.4	3.5
	W=40	0.2	2.0	6.4	14.9
R-Interpol	R=10	0.0	0.4	1.4	3.3
	R=40	0.2	1.6	5.7	13.4
WR-Interpol	W=5,R=5	0.1	1.6	3.6	8.4
	W=10,R=10	0.5	4.4	14.5	34.2
	W=20,R=5	0.6	4.5	14.8	35.2
	W=5,R=20	0.5	4.2	14.3	33.7

Για τους αλγορίθμους προγραμματισμού εργασιών με προθεσμίες (Feas-Interpol, W-Feas-Interpol, WR-Feas-Interpol) δεν θα παραθέσουμε μετρήσεις απαιτούμενου υπολογιστικού χρόνου. Επειδή, η διαδικασία εφικτής παρεμβολής λειτουργεί όπως ακριβώς η διαδικασία παρεμβολής μόνο στην ακραία περίπτωση πολύ χαλαρών προθεσμιών, οι μερικές διατάξεις που εξετάζει είναι κατά μέσο όρο σημαντικά πιο λίγες. Ετσι, ο πίνακας 5 με τους υπολογιστικούς χρόνους για τους αλγορίθμους παρεμβολής παρέχει το μέγιστο υπολογιστικό χρόνο για τους αντίστοιχους αλγορίθμους εφικτής παρεμβολής.

8 Επίλογος

8.1 Απόδοση αλγορίθμων παρεμβολής για το πρόβλημα ροϊκής παραγωγής

Το πρόβλημα ροϊκής παραγωγής συγκατελέγεται μεταξύ των δύσκολων προβλημάτων στο χώρο του προγραμματισμού παραγωγής. Η συνδυαστική πλοκή και η εξάρτηση από τα δεδομένα αποτελούν χαρακτηριστικά δομικά στοιχεία του. Στην εργασία αυτή μας απασχόλησαν δύο μορφές προβλημάτων: το $N/M/F, P, S_{seq-ind}/C_{max}$ και το $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$. Η επίλυσή τους από τους αλγορίθμους παρεμβολής που παρουσιάσαμε οδήγησε σε ουσιαστικές παρατηρήσεις γύρω από τη φύση του προβλήματος και τις εγγενείς δυσκολίες που παρουσιάζει.

Θεωρώντας το σχετικό σφάλμα περισσότερο δίκαιο και αξιόπιστο κριτήριο επίδοσης των αλγορίθμων που στοχεύουν στην ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης προγράμματος, προσδιορίσαμε καταρχήν το βαθμό δυσκολίας του προβλήματος σε σχέση με το μέγεθός του. Από την πειραματική αξιολόγηση των αλγορίθμων παρεμβολής του κεφαλαίου 7 αποδείχθηκε ότι τα προβλήματα με μικρό αριθμό εργασιών (6,8,10) ή μεγάλο αριθμό μηχανών (12,16) παρουσιάζουν σταθερά μεγαλύτερο σχετικό σφάλμα από εκείνα με μεγάλο αριθμό εργασιών (20,30,40,50) και μικρό αριθμό μηχανών (4,8).

Η αυξημένη δυσκολία των προβλημάτων με μεγάλο αριθμό μηχανών αιτιολογείται από την ύπαρξη σημαντικών νεκρών χρόνων κατά την εκτέλεση. Τα πολλά στάδια παραγωγής μεγαλώνουν την πιθανότητα ανομοιομορφίας των χρόνων επεξεργασίας και, κατά συνέπεια, της εμφάνισης νεκρών διαστημάτων σε παραπάνω της μιας μηχανές. Επειδή η διαδικασία παρεμβολής λειτουργεί κατασκευαστικά και φροντίζει συνεχώς για το μικρό χρόνο εκτέλεσης της τρέχουσας μερικής διάταξης, αδυνατεί να εξαλείφει όλους τους νεκρούς χρόνους στα διαφορετικά στάδια παραγωγής, ειδικά όταν έχει ήδη αποφασίσει τη σχετική θέση εργασιών που έχουν προγραμματιστεί νωρίτερα στη διαδικασία. Μ' αυτόν τον τρόπο η απόσταση από το κάτω φράγμα για το βέλτιστο αυξάνει σημαντικά με αποτέλεσμα το σχετικό σφάλμα να παρουσιάζεται μεγάλο. Στην περίπτωση αυτή θα είχε ενδιαφέρον η σύγκριση των λύσεων που παράγουν οι αλγόριθμοι παρεμβολής με το ολικό βέλτιστο και όχι με κάποιο φράγμα γι' αυτό. Μόνο αυτή η σύγκριση θα απαντούσε στο ερώτημα κατά πόσο το μεγάλο σχετικό σφάλμα στην περίπτωση πολλών μηχανών οφείλεται πραγματικά στην κακή απόδοση των αλγορίθμων παρεμβολής ή στον τρόπο υπολογισμού του κάτω φράγματος για το βέλτιστο, που γίνεται ακόμα χαλαρότερο στην περίπτωση αυτή λόγω των περισσότερων νεκρών χρόνων.

Αντίστοιχα, όταν ο αριθμός των εργασιών είναι μικρός, το μεγάλο σχετικό σφάλμα δεν πρέπει να αποδωθεί αποκλειστικά στη λειτουργία του μηχανισμού παρεμβολής. Επειδή ο συνολικός χρόνος εκτέλεσης προγράμματος είναι μικρός σε απόλυτη τιμή, εμφάνιση έστω και μικρών νεκρών χρόνων συνεπάγεται αυτόματα θεαματική αύξηση του σχετικού σφάλματος. Οι αλγόριθμοι παρεμβολής αδυνατούν να εξαλείψουν αυτούς τους μικρούς νεκρούς χρόνους, αφού παράγουν λύση σε πολύ λίγα βήματα και δεν υπάρχουν περιθώρια ομαλοποίησης της εκτέλεσης. Ενδέχεται, όμως, να μην είναι δυνατόν να εξαλειφθούν αυτοί οι νεκροί χρόνοι ούτε στο ολικό βέλτιστο. Αυτό θα το γνωρίζαμε μόνο αν το πρόγραμμα αναφοράς για τις επιδόσεις των αλγορίθμων ήταν η βέλτιστη λύση.

Πάντως, οι αλγόριθμοι παρεμβολής που προτάθηκαν στην προκείμενη εργασία, εμφανίζουν σταθερά καλύτερες επιδόσεις από τους υπόλοιπους ευρηματικούς αλγορίθμους για το πρόβλημα ροϊκής παραγωγής. Η σύγκρισή τους με τον αλγόριθμο του Nawaz, ο οποίος βρισκόταν στην κορυφή του καταλόγου επιδόσεων, απέδειξε την σταθερή υπεροχή τους. Οι μηχανισμοί αποθήκευσης συνόλου μερικών διατάξεων ανά στάδιο και επαναληπτικής εκτέλεσης της παρεμβολής με τυχαίες ουρές προγραμματισμού οδήγησαν σε βελτιώσεις ως προς τον αλγόριθμο Nawaz της τάξης του 20%-25%, ακόμα και για μικρές τιμές των παραμέτρων ελέγχου. Οι βελτιώσεις αυτές έφτασαν σε ορισμένες περιπτώσεις το 40%-45%. Για το πρόβλημα του δικριτηριακού προγραμματισμού παραγωγής με πρωτεύον κριτήριο την ελαχιστοποίηση της μέγιστης καθυστέρησης και δευτερεύον εκείνο του συνολικού χρόνου εκτέλεσης, δεν ήταν δυνατή αντίστοιχη αξιολόγηση των αλγορίθμων εφικτής παρεμβολής, μια καιτο πρόβλημα αντιμετωπίζεται για πρώτη φορά. Το ουσιαστικό συμπέρασμα των δοκιμών ήταν ότι οι μηχανισμοί βελτίωσης της εφικτής παρεμβολής αποδείχθηκαν αποδοτικοί και για το δεύτερο πρόβλημα ροϊκής παραγωγής που μελετήσαμε.

8.2 Προτάσεις βελτιώσεων

Από τις πειραματικές δοκιμές για το πρόβλημα $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$ εντοπίστηκε η αδυναμία παραγωγής καλών λύσεων σε περιπτώσεις που οι πρώτες προθεσμίες εργασίων τυχαίνει να είναι σφικτές. Τότε, το εφικτό του προγράμματος χάνεται από τα πρώτα βήματα της διαδικασίας παρεμβολής και όλες οι υπόλοιπες εργασίες διατάσσονται κατά αύξουσα τάξη προθεσμιών, προκειμένου να ελαχιστοποιηθεί η μέγιστη καθυστέρηση. Στην περίπτωση αυτή λοιπόν παύει η διαδικασία προγραμματισμού με παρεμβολή και τα περιθώρια βελτίωσης των λύσεων

στενεύουν, ακόμα κι αν οι επόμενες εργασίες έχουν χαλαρές ημερομηνίες και θα μπορούσαν να εκτελεστούν κατά αποδοτικότερο τρόπο. Η πολιτική λοιπόν που ακολουθούμε δεν φαίνεται ιδιαίτερα αποτελεσματική στην ειδική αυτή περίπτωση.

Μια διαφορετική πολιτική που θα μπορούσε να εφαρμοστεί όταν η διαδικασία οδηγείται σε απώλεια του εφικτού είναι η συνέχιση της παρεμβολής με αλλαγή του κριτηρίου βελτίστου ανά στάδιο. Οσο κινούμαστε μεταξύ εφικτών προγραμμάτων επιλέγουμε σε κάθε βήμα τις διατάξεις εκείνες που έχουν το μικρότερο χρόνο εκτέλεσης. Αν χάσουμε το εφικτό της λύσης σε κάποιο βήμα, μπορούμε να προγραμματίσουμε τις επόμενες εργασίες παρεμβάλλοντάς τις κατά τρόπο ώστε να ελαχιστοποιούμε τη μέγιστη καθυστέρηση σε κάθε στάδιο κι όχι το συνολικό χρόνο εκτέλεσης. Η πολιτική αυτή διατηρεί την απόλυτη προτεραιότητα των εργασιών με προθεσμίες έναντι των εργασιών χωρίς προθεσμίες και παράλληλα εξετάζει συνεχώς διατάξεις με μικρές καθυστερήσεις εργασιών. Μ' αυτόν τον τρόπο φροντίζουμε για την ελαχιστοποίηση της μέγιστης καθυστέρησης, χωρίς ουσιαστική διαφορά στο συνολικό χρόνο εκτέλεσης.

Ακόμα, όμως, κι αν χρησιμοποιούμε τους αλγορίθμους παρεμβολής στην τρέχουσα μορφή τους, υπάρχει τρόπος αντιμετώπισης της περίπτωσης γρήγορης απώλειας του εφικτού. Αν στη λύση των αλγορίθμων παρατηρηθεί ότι οι πρώτες εργασίες είναι καθυστερημένες και, κατά συνέπεια, η ποιότητα όλου του προγράμματος δεν είναι η αναμενόμενη, ο επαναπρογραμματισμός των εργασιών που δεν έχουν σφικτές προθεσμίες σε ανεξάρτητη διάταξη θα μπορούσε να οδηγήσει σε βελτιώσεις και των δύο κριτηρίων βελτίστου. Ουσιαστικά, αφαιρούμε από τις εργασίες προς προγραμματισμό τις πρώτες, οι οποίες παρουσίασαν γρήγορη καθυστέρηση, και επαναπρογραμματίζουμε τις υπόλοιπες με ορίζοντα προγραμματισμού μετακινημένο όσο διαρκεί η εκτέλεση των εργασιών που αφαιρέθηκαν. Αν οι προθεσμίες ήταν πραγματικά χαλαρές για τις υπόλοιπες εργασίες, θα παραμείνουν χαλαρές και στο νέο πρόβλημα. Οι αλγόριθμοι παρεμβολής θα λειτουργήσουν αποτελεσματικά τότε και το νέο πρόγραμμα θα εκτελεστεί αμέσως μετά τις εργασίες που είχαν εξαιρεθεί από τον επαναπρογραμματισμό.

8.3 Επεκτάσεις

Οι αλγόριθμοι προγραμματισμού εργασιών με παρεμβολή που παρουσιάσαμε στην εργασία αυτή εφαρμόστηκαν στο πρόβλημα ροϊκής παραγωγής, το οποίο ανήκει στην κατηγορία των δύσκολων συνδυαστικών προβλημάτων. Το ουσιαστικό πλεόνεκτημα της διαδικασίας παρεμβολής είναι ο κατασκευαστικός χαρακτήρας της που επιτρέπει

τον έλεγχο της ποιότητας της τρέχουσας μερικής διάταξης ανά βήμα. Το στοιχείο αυτό μπορεί να καταστήσει τη διαδικασία παρεμβολής κατάλληλο αλγοριθμικό σχήμα και για άλλα συνδυαστικά προβλήματα διάταξης εργασιών.

Μεταξύ των προβλημάτων προγραμματισμού εργασιών σε μια μηχανή, ιδιαίτερα δύσκολα θεωρούνται εκείνα στα οποία οι χρόνοι εξάρμωσης των μηχανών είναι εξαρτημένοι ακολουθίας. Σε περιπτώσεις πολύ μεγάλων χρόνων εξάρμωσης, η απόδοση οποιουδήποτε αλγόριθμου επίλυσής τους εξαρτάται από την κατάλληλη παρτιδοποίηση εργασιών με σύγχρονη μέριμνα για καλή απόδοση ως προς το κριτήριο βελτίστου, όποιο κι αν είναι αυτό. Οι αλγόριθμοι παρεμβολής επειδή λειτουργούν κατασκευαστικά και σε κάθε βήμα λαμβάνουν υπόψη τους εξαρτημένους ακολουθίας χρόνους εξάρμωσης, ενδέχεται να αποδώσουν καλά ανεξάρτητα από το κριτήριο βελτίστου. Η παραπάνω υπόθεση μπορεί να επεκταθεί και σε προβλήμματα προγραμματισμού εργασιών σε παράλληλες μηχανές, όπου υπάρχει η δυνατότητα εναλλακτικής εκτέλεσης κάθε εργασίας σε πολλές μηχανές. Η παρεμβολή της τρέχουσας εργασίας στην περίπτωση αυτή θα πραγματοποιοείται από την ουρά προγραμματισμού σε όλες τις δυνατές θέσεις των μερικών προγραμμάτων όλων των μηχανών.

Αν οι αλγόριθμοι παρεμβολής πρόκειται να εφαρμοστούν στα παραπάνω προβλήματα, ιδιαίτερα κρίσιμο στοιχείο τους είναι το κριτήριο αρχικής διάταξης των εργασιών στην ουρά προγραμματισμού. Επειδή η απόδοση του κάθε κριτηρίου διάταξης δεν εξαρτάται μόνο από το κριτήριο βελτίστου, αλλά και από τα εγγενή χαρακτηριστικά της γραμμής παραγωγής, είναι απαραίτητη η συστηματική διερεύνηση των επιδόσεων για διαφορετικά κριτήρια διάταξης στην ουρά προγραμματισμού. Πάντως, η εφαρμογή του μηχανισμού επαναληπτικής εκτέλεσης της παρεμβολής με τυχαίες ουρές προγραμματισμού που προτείναμε στην παρούσα εργασία, αποτελεί εγγύηση για την καλή απόδοση των αλγορίθμων υπο οποιεσδήποτε συνθήκες.

Ενα διαφορετικό πεδίο πιθανής επέκτασης των αλγορίθμων παρεμβολής είναι ο δικριτηριακός προγραμματισμός. Στην εργασία αυτή αντιμετωπίσαμε το δικριτηριακό προγραμματισμό ως προς τη μέγιστη καθυστέρηση και το συνολικό χρόνο εκτέλεσης προγράμματος. Ομως, η εφικτή παρεμβολή μπορεί να λειτουργήσει λαμβάνοντας υπόψη οποιοδήποτε περιορισμό ως προς το πρωτεύον κριτήριο και μεριμνώντας για την καλή απόδοση του δευτερεύοντος κριτηρίου. Πάντα βέβαια, απαιτείται η προσαρμογή του γενικού αλγοριθμικού σχήματος στις απαιτήσεις και τις ιδιομορφίες του κάθε προβλήματος για τη διατήρηση των προτεραιοτήτων μεταξύ των κριτηρίων και την εξισορρόπηση των αποκλίσεων από τους διαφορετικούς για κάθε κριτήριο στόχους.

Παραπομπές

[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--75, January-February 1975.

[BLK83] J. Blazewicz, J.K. Lenstra, and A.H.G. Rinnooy Kan. Scheduling Subject to Resource Constrains: Classification and Complexity. *Discrete Applied Mathematics*, 5:11--24, 1983.

[BPH82] J. Blackstone, D. Phillips, and G. Hogg. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *Int. Journal of Production Research*, 20(1):27--45, 1982.

[CB92] J. Cao and D. Bedworth. Flowshop Scheduling in Serial multi-product processes with Transfer and Setup Times. *International Journal of Production Research*, 30(8):1819--1830, 1992.

[CDS70] H.G. Campbell, R.A. Dudek, and N.L. Smith. A Heuristic Algorithm for the N-job, M-machine sequencing problem. *Management Science*, 16:B603--B637, 1970.

[CGD91] 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.

[Cha93] Κ. Χαριτωνίδης Δικριτηριακός Προγραμματισμός Μιας Μηχανής με Χρόνους Εξάρμωσης. Μεταπτυχιακή Εργασία, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, 1993.

[CK89] C.N.Potts and K.R.Baker. Flowshop Scheduling with lot streaming. *Operations Research Letters*, 8(6), 1989.

[Dan77] D.G. Dannenbring. An Evaluation of Flowshop Sequencing Heuristics. *Management Science*, 23:1174--1182, 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.

[Geo91] Θ. Γεωργίου Το Πρόβλημα της Μέγιστης Καθυστέρησης σε Μια Μηχανή με Χρόνους Εξάρμωσης. Μεταπτυχιακή Εργασία, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, 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.

[GR88] G.Parker and R.Raldin. *Discrete Optimization.* Acadenic Press, Inc., 1988.

[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.

[Gup71] J.N.D. Gupta. An Improved Combinatorial Algorithm for the Flowshop Scheduling Problem. *Operations Research*, 19:1753--1758, 1971.

[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.

[Hax78] A. C. Hax. Aggregate Production Planning, 1978.

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

[HH] A.C. Hax and H.C.Meal. Hierarchical Integration of Production Planning and Scheduling.

[IL65] E. Ignall and L.Schrage. Application of the Branch and Bound Technique to some Flowshop Scheduling Problems. *Operations Research*, 13:400--412, 1965.

[Jac55] J.R. Jackson. Scheduling a Production Line to Minimize Maximum Tardiness. Technical report, Management Sciences Research Report 43, University of California at Los Angeles, 1955.

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

[J.R85] J.R.Biggs. Priority Rules for Shop Floor Control in a Material Requirements Planning System under Various Levels of Capacity. *Int.J.Production Research*, 23(1):33--46, 1985.

[JW89] J.D.Yanney and W.Kuo. A Practical Approach to Scheduling a Multistage, Multiprocessor Flowshop. *Int.J.Production Research*, 27(10):1716--1733, October 1989.

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

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

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

[Man59] A. S. Manne. On the Job-Shop Scheduling Problem, 1959.

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

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

[NS89] E. Nowicki and C. Smutnicki. Worst-case Analysis of an Approximation Algorithm for Flow-shop Scheduling. *Operations Research Letters*, 8:171--177, 1989.

[NS91] E. Nowicki and C. Smutnicki. Worst-case Analysis of Dannenbring's Algorithm for Flow-shop scheduling. *Operations Research Letters*, 10:473--480, 1991.

[Ow85] Peng Si Ow. Focused Scheduling in Proportionate Flowshops. *Management Science*, 31(7), 1985.

[PB92] P.Pinto and B.Rao. Joint lot-sizing and scheduling for multi-stage multi-product flowshops. *Int.J.Production Research*, 30(5):1137--1152, 1992.

[PSW91] C. Potts, D. Shmoys, and D. Williamson. Permutation vs. Non-Permutation Flowshop Schedules. *Operations Research Letters*, 10:281--284, July 1991.

[RB92] R.G.Vickson and B.E.Alfredsson. Two and Three machine Flowshop Scheduling Problems with Equal Sized Transfer Batches. *Int.J.Production Research*, 30(7):1538--1551, July 1992.

[Sah77] S. Sahni. General Techniques for Combinatorial Approximation. *Operations Research*, 25(6):920--935, November-December 1977.

[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*, 15:71--82, 1967.

[SW76] S.S.Panwalkar and W.Iskander. A Survey of Scheduling Rules, 1976.

[Szw83] W. Szwarc. Flowshop problems with time lags. *Management Science*, 29(4):477--481, 1983.

[Tsa93] Ν. Τσατσάκης Το Πρόβλημα του Συνολικού Χρόνου Περάτωσης Ν Εργασιών σε Μια Μηχανή με Χρόνους Εξάρμωσης. Μεταπτυχιακή Εργασία, Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης, 1993.

[WH89] M. Widmer and A. Hertz. A New Heuristic Method for the Flowshop sequencing problem. *European Journal of Operational Research*, 41:186--193, 1989.

Α ΠΑΡΑΡΤΗΜΑ. Μετρήσεις για το $N/M/F, P, S_{seq-ind}/C_{max}$

Σύγκριση των μηχανισμών βελτίωσης (υποκεφάλαιο 7.2.1) : απόλυτο σφάλμα για τους Nawaz-Algo (στήλη 2), W-Interpol για W=5,10,20,40 (στήλες 3-6) και R-Interpol για R=5,10,20,40 (στήλες 7-10).

NxM	NAW	W=5	W=10	W=20	W=40	R=5	R=10	R=20	R=40
6x8	198.1	179.7	178.7	178.5	178.5	183.6	179.8	179.4	178.5
8x8	233.4	211.2	203.9	202.6	198.1	211.1	203.6	201.4	198.7
10x8	223.4	200.8	196.4	193.5	188.6	198.2	197.3	189.3	184.7
20x8	240.5	215.3	203.1	186.9	182.3	209.1	197.1	183.7	179.8
30x8	192.5	157.5	154.4	135.4	142.0	168.1	149.3	141.0	132.0
40x8	196.7	179.1	170.5	167.1	151.5	180.5	160.3	145.9	136.5
50x8	175.0	152.8	151.3	162.9	144.5	172.6	142.5	132.6	126.4
20x4	42.1	40.2	39.3	36.9	37.0	37.9	34.0	32.4	28.2
20x8	240.5	215.3	203.1	186.9	182.3	209.1	197.1	183.7	179.8
20x12	445.6	393.1	378.3	362.0	354.4	398.2	388.0	378.3	362.0
20x16	643.3	572.5	553.9	542.7	528.3	596.4	581.1	560.3	544.8

Σύγκριση των μηχανισμών βελτίωσης (υποκεφάλαιο 7.2.1) : σχετικό σφάλμα για τους Nawaz-Algo (στήλη 2), W-Interpol για W=5,10,20,40 (στήλες 3-6) και R-Interpol για R=5,10,20,40 (στήλες 7-10).

NxM	NAW	W=5	W=10	W=20	W=40	R=5	R=10	R=20	R=40
6x8	14.6	13.3	13.2	13.2	13.2	13.6	13.3	13.2	13.2
8x8	14.5	13.2	12.7	12.6	12.4	13.2	12.7	12.6	12.4
10x8	12.1	10.9	10.7	10.5	10.2	10.8	10.7	10.2	10.0
20x8	7.6	6.8	6.4	5.9	5.7	6.6	6.3	5.8	5.7
30x8	4.4	3.6	3.5	3.1	3.2	3.8	3.4	3.2	3.0
40x8	3.4	3.1	3.0	2.9	2.6	3.1	2.8	2.5	2.3
50x8	2.5	2.2	2.1	2.3	2.0	2.5	2.1	1.9	1.8
20x4	1.5	1.4	1.4	1.3	1.3	1.4	1.2	1.1	1.0
20x8	7.6	6.8	6.4	5.9	5.7	6.6	6.3	5.8	5.7
20x12	12.8	11.3	10.9	10.4	10.2	11.4	11.1	11.0	10.4
20x16	16.4	14.6	14.1	13.9	13.5	15.2	14.9	14.3	13.9

Σύγκριση των μηχανισμών βελτίωσης (υποκεφάλαιο 7.2.1) : ποσοστιαία βελτίωση (%) ως προς Nawaz-Algo για τους W-Interpol για W=5,10,20,40 (στήλες 2-5) και R-Interpol για R=5,10,20,40 (στήλες 6-9).

NxM	W=5	W=10	W=20	W=40	R=5	R=10	R=20	R=40
6x8	9.3	9.8	9.9	9.9	7.3	9.2	9.4	9.8
8x8	9.5	12.6	13.1	15.1	9.5	12.7	13.7	14.8
10x8	10.1	12.1	13.4	15.6	11.2	11.7	15.3	17.3
20x8	10.4	15.5	22.2	24.2	13.1	18.1	23.6	25.2
30x8	18.1	20.0	29.7	26.2	12.7	22.4	26.8	31.4
40x8	8.9	13.3	15.0	23.0	8.2	18.5	25.8	30.6
50x8	12.7	13.5	7.0	17.4	1.4	18.6	24.2	27.7
20x4	4.5	6.7	12.3	12.1	9.9	19.2	23.0	33.0
20x8	10.4	15.5	22.2	24.2	13.1	18.1	23.6	25.2
20x12	11.7	15.1	18.8	20.4	10.6	12.9	15.1	18.7
M.O.	10.6	13.4	16.3	18.7	9.4	15.5	19.4	22.6

Σύγχρονη λειτουργία των δύο μηχανισμών (υποκεφάλαιο 7.2.2) : απόλυτο σφάλμα για τους Nawaz-Algo (στήλη 2) και WR-Interpol για (W,R)=(3,8),(8,3),(5,5),(5,20),(20,5),(10,10) (στήλες 3-8).

NxM	NAW	W=3 R=8	W=8 R=3	W=5 R=5	W=5 R=20	W=20 R=5	W=10 R=10
6x8	191.1	179.6	178.8	179.3	178.7	178.7	178.7
8x8	213.0	186.0	186.0	184.9	183.6	183.5	183.8
10x8	239.0	189.1	191.1	191.6	187.9	186.7	188.2
20x8	226.0	174.7	171.9	181.0	148.3	153.2	156.4
30x8	217.6	164.9	165.2	171.5	146.8	147.6	140.6
40x8	211.2	157.2	163.4	170.0	140.1	141.3	140.5
50x8	187.5	136.0	155.2	143.0	122.0	135.6	125.9
10x4	46.8	29.3	30.8	30.2	27.8	29.6	27.4
10x8	239.0	189.1	191.1	191.6	187.9	186.7	188.2
10x12	388.7	327.8	332.9	325.7	321.2	320.4	322.9
10x16	529.0	476.7	474.7	474.0	466.2	467.1	466.5

Σύγχρονη λειτουργία των δύο μηχανισμών (υποκεφάλαιο 7.2.2) : σχετικό σφάλμα για τους Nawaz-Algo (στήλη 2) και WR-Interpol για (W,R)=(3,8),(8,3),(5,5),(5,20),(20,5),(10,10) (στήλες 3-8).

NxM	NAW	W=3 R=8	W=8 R=3	W=5 R=5	W=5 R=20	W=20 R=5	W=10 R=10
6x8	13.9	13.2	13.1	13.1	13.1	13.1	13.1
8x8	12.8	11.2	11.2	11.2	11.1	11.1	11.1
10x8	12.8	10.2	10.3	10.3	10.1	10.0	10.0
20x8	7.1	5.5	5.4	5.7	4.6	4.8	4.9
30x8	5.0	3.8	3.8	3.9	3.4	3.4	3.2
40x8	3.7	2.8	2.9	3.0	2.4	2.5	2.5
50x8	2.7	2.0	2.2	2.0	1.7	1.9	1.8
10x4	3.2	2.0	2.1	2.1	1.9	2.1	1.9
10x8	12.8	10.2	10.3	10.3	10.1	10.0	10.0
10x12	16.9	14.2	14.5	14.1	14.0	13.9	14.1
10x16	20.4	18.4	18.3	18.3	18.0	18.1	18.0

Σύγχρονη λειτουργία των δύο μηχανισμών (υποκεφάλαιο 7.2.2) : ποσοστιαία βελτίωση (%) ως προς Nawaz-Algo του WR-Interpol για (W,R)=(3,8),(8,3),(5,5),(5,20),(20,5),(10,10) (στήλες 2-7).

NxM	W=3 R=8	W=8 R=3	W=5 R=5	W=5 R=20	W=20 R=5	W=10 R=10
6x8	6.0	6.4	6.1	6.5	6.5	6.5
8x8	12.7	12.7	13.2	13.8	13.8	13.7
10x8	20.9	20.0	19.8	21.4	21.9	21.2
20x8	22.7	24.0	19.9	34.4	32.2	30.8
30x8	24.2	24.1	21.2	32.5	32.2	35.4
40x8	25.6	22.6	19.5	33.7	33.1	33.4
50x8	27.4	17.2	23.7	34.9	27.7	32.9
10x4	37.4	34.2	35.5	40.6	36.8	41.4
10x8	20.9	20.0	19.8	21.4	21.9	21.2
10x12	15.7	14.3	16.2	17.3	17.6	17.0
10x16	9.9	10.3	10.4	11.9	11.7	11.8
M.O.	20.3	18.7	18.6	24.4	23.2	24.1

Υπεροχή του μηχανισμού επαναληπτικής παρεμβολής (υποκεφάλαιο 7.2.3) : απόλυτο σφάλμα για τους Nawaz-Algo (στήλη 2) W-Interpol για W=9,25,49 (στήλες 3,6,9), R-Interpol για R=9,25,49 (στήλες 4,7,10) και WR-Interpol για (W,R)=(3,3),(5,5),(7,7) (στήλες 5,8,11).

NxM	NAW	W=9	R=9	W=3 R=3	W=25	R=25	W=5 R=5	W=49	R=49	W=7 R=7
6x8	219.0	205.9	210.3	209.0	205.9	211.3	207.4	205.9	206.8	205.9
8x8	213.9	178.5	185.2	182.6	176.5	178.5	178.1	176.5	177.4	176.8
10x8	243.6	213.4	213.6	216.3	207.0	205.2	206.2	205.6	204.6	202.7
20x8	246.0	210.4	210.1	200.5	207.5	193.1	198.3	196.9	183.6	174.9
30x8	221.9	179.9	190.9	193.5	166.9	171.4	171.2	170.5	158.2	167.9
40x8	210.8	178.7	175.1	197.7	174.3	154.6	168.6	161.5	155.3	159.9
50x8	193.5	170.5	177.5	189.1	162.6	151.7	169.6	158.8	146.1	154.8
20x4	45.5	40.0	31.0	39.4	39.2	28.13	29.1	39.1	25.5	30.2
20x8	246.0	210.4	210.1	200.5	207.5	193.1	198.3	196.9	183.6	174.9
20x12	412.0	354.9	349.2	339.8	347.0	326.7	315.8	331.6	315.7	314.1
20x16	619.8	561.5	571.5	565.2	533.7	539.1	526.9	518.2	534.6	516.2

Υπεροχή του μηχανισμού επαναληπτικής παρεμβολής (υποκεφάλαιο 7.2.3) : σχετικό σφάλμα για τους Nawaz-Algo (στήλη 2) W-Interpol για W=9,25,49 (στήλες 3,6,9), R-Interpol για R=9,25,49 (στήλες 4,7,10) και WR-Interpol για (W,R)=(3,3),(5,5),(7,7) (στήλες 5,8,11).

NxM	NAW	W=9	R=9	W=3 R=3	W=25	R=25	W=5 R=5	W=49	R=49	W=7 R=7
6x8	15.8	14.9	15.2	15.1	14.9	15.3	15.0	14.9	14.9	14.9
8x8	13.3	11.1	11.5	11.3	11.0	11.1	11.1	10.9	11.0	11.0
10x8	13.2	11.6	11.7	11.3	11.2	11.2	11.2	11.2	11.1	11.0
20x8	7.9	6.8	6.8	6.5	6.7	6.2	6.4	6.3	5.9	5.6
30x8	4.9	4.0	4.2	4.3	3.7	3.8	3.8	3.8	3.5	3.7
40x8	3.7	3.1	3.1	3.4	3.0	2.7	2.9	2.8	2.7	2.8
50x8	2.8	2.5	2.5	2.7	2.3	2.2	2.4	2.3	2.1	2.2
20x4	1.6	1.4	1.1	1.4	1.4	1.0	1.0	1.4	0.9	1.1
20x8	7.9	6.8	6.8	6.5	6.7	6.2	6.4	6.3	5.9	5.6
20x12	11.6	10.0	9.9	9.6	9.8	9.2	8.9	9.4	8.9	8.9
20x16	15.7	14.2	14.5	14.3	13.5	13.7	13.4	13.1	13.6	13.1

Υπεροχή του μηχανισμού επαναληπτικής παρεμβολής (υποκεφάλαιο 7.2.3) : ποσοστιαία βελτίωση ως προς Nawaz-Algo των W-Interpol για W=9,25,49 (στήλες 2,5,8), R-Interpol για R=9,25,49 (στήλες 3,6,9) και WR-Interpol για (W,R)=(3,3),(5,5),(7,7) (στήλες 4,7,10).

NxM	W=9	R=9	W=3 R=3	W=25	R=25	W=5 R=5	W=49	R=49	W=7 R=7
6x8	6.0	4.0	4.5	6.0	3.5	5.2	6.0	5.6	6.0
8x8	16.5	13.4	14.6	17.5	16.5	16.7	17.5	17.1	17.3
10x8	12.4	12.3	11.2	15.0	15.8	15.3	15.6	16.0	16.8
20x8	14.5	14.6	18.5	15.6	21.5	19.4	20.0	25.4	28.9
30x8	18.9	14.0	12.8	24.8	22.7	22.8	23.1	28.7	24.3
40x8	15.2	16.9	6.2	17.3	26.6	20.0	23.4	26.3	24.1
50x8	11.9	8.2	2.2	16.0	21.6	12.3	17.9	24.5	20.0
20x4	12.1	31.2	13.4	13.8	38.2	36.0	14.1	44.0	33.6
20x8	14.5	14.6	18.5	15.6	21.5	19.4	20.0	25.4	28.9
20x12	13.8	15.2	17.5	15.7	20.7	23.3	19.5	23.4	23.8
20x16	9.4	7.8	8.8	13.9	13.0	15.0	16.4	13.7	16.7
M.O.	13.2	13.8	11.7	15.6	20.1	18.7	17.6	22.7	21.9

Αλλαγή κατανομών (υποκεφάλαιο 7.2.4) : απόλυτο σφάλμα για τους Nawaz-Algo (στήλη 2) και WR-Interpol για (W,R)=(3,8),(8,3),(5,5),(5,20),(20,5),(10,10) (στήλες 3-8).

NxM	NAW	W=3 R=8	W=8 R=3	W=5 R=5	W=5 R=20	W=20 R=5	W=10 R=10
6x8	112.3	98.7	97.9	98.1	97.9	97.9	97.9
8x8	115.6	101.7	101.2	101.1	100.7	100.7	100.7
10x8	106.2	82.1	81.3	81.5	79.2	79.3	78.3
20x8	111.2	81.1	82.9	81.2	73.7	71.2	74.5
30x8	98.6	72.0	74.5	72.8	62.9	66.2	63.3
10x4	24.4	17.2	17.8	17.8	17.1	17.2	17.1
10x8	106.2	82.1	81.3	81.5	79.2	79.3	78.3
10x12	166.7	149.1	150.8	148.1	146.6	147.0	146.9
10x16	255.7	229.8	229.4	228.4	223.4	224.8	224.0

Αλλαγή κατανομών (υποκεφάλαιο 7.2.4) : σχετικό σφάλμα για τους Nawaz-Algo (στήλη 2) και WR-Interpol για (W,R)=(3,8),(8,3),(5,5),(5,20),(20,5),(10,10) (στήλες 3-8).

NxM	NAW	W=3 R=8	W=8 R=3	W=5 R=5	W=5 R=20	W=20 R=5	W=10 R=10
6x8	16.4	14.4	14.3	14.3	14.3	14.3	14.3
8x8	14.3	12.6	12.6	12.5	12.5	12.5	12.5
10x8	11.1	8.6	8.5	8.6	8.3	8.3	8.2
20x8	7.1	5.2	5.3	5.2	4.7	4.6	4.8
30x8	4.4	3.2	3.4	3.3	2.8	3.0	2.9
10x4	3.2	2.3	2.4	2.4	2.3	2.3	2.3
10x8	11.1	8.6	8.5	8.6	8.3	8.3	8.2
10x12	14.8	13.2	13.4	13.1	13.0	13.0	13.0
10x16	19.2	17.3	17.3	17.2	16.8	16.9	16.9

Αλλαγή κατανομών (υποκεφάλαιο 7.2.2) : ποσοστιαία βελτίωση (%) ως προς Nawaz-Algo του WR-Interpol για (W,R)=(3,8),(8,3),(5,5),(5,20),(20,5),(10,10) (στήλες 2-7).

NxM	W=3 R=8	W=8 R=3	W=5 R=5	W=5 R=20	W=20 R=5	W=10 R=10
6x8	12.1	12.8	12.6	12.8	12.8	12.8
8x8	12.0	12.5	12.5	12.9	12.9	12.9
10x8	22.7	23.4	23.6	25.4	25.3	26.3
20x8	27.1	25.5	27.0	33.7	36.0	33.0
30x8	27.0	24.4	26.2	36.2	32.9	35.8
10x4	29.5	27.0	27.0	29.9	29.5	29.9
10x8	22.7	23.4	23.6	25.4	25.3	26.3
10x12	10.5	9.5	11.1	12.1	11.8	11.9
10x16	10.1	10.0	10.7	12.6	12.1	12.4

Β ΠΑΡΑΡΤΗΜΑ. Μετρήσεις για το $N/M/F, P, S_{seq-ind}/T_{max}, C_{max}$

Συγκρινόμενοι αλγόριθμοι:

1. Feas-Interpol
2. W-Feas-Interpol
3. WR-Feas-Interpol

Κριτήρια επιδόσεων (στήλες πινάκων):

- ΕΛ: αριθμός προβλημάτων για τα οποία ο αλγόριθμος εντόπισε εφικτή λύση (επί συνόλου 30 προβλημάτων).
- ΒΚ: αριθμός προβλημάτων για τα οποία ο αλγόριθμος εντόπισε μη εφικτή λύση με μικρότερη καθυστέρηση από αυτήν της μη εφικτής λύσης που παρήγαγε ο Feas-Interpol για το ίδιο πρόβλημα.
- ΠΒΧ: ποσοστό βελτίωσης (%) του χρόνου εκτέλεσης του τελικού προγράμματος του αλγορίθμου ως προς το χρόνο εκτέλεσης εκείνου του Feas-Interpol.

Ποσοστό εργασιών με προθεσμίες ddper=50%

	NAW	W=100			W=20 R=5			W=5 R=20			W=10 R=10		
NxM	ΕΛ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ
6x8	18	18	1	10.0	18	1	10.0	18	1	9.9	18	1	10.0
8x8	19	20	1	7.7	20	1	7.7	20	1	7.5	20	1	7.5
10x4	22	23	1	15.2	23	1	12.3	23	1	14.5	23	1	15.3
10x8	19	19	3	12.9	19	3	12.7	19	3	10.8	19	3	12.3
10x12	16	17	3	11.0	17	3	10.5	17	3	10.5	17	3	10.7
10x16	12	14	5	4.8	14	5	4.6	14	5	4.0	14	5	4.8
20x4	22	23	1	16.8	23	1	22.4	23	1	22.1	23	1	23.9
20x8	18	19	2	17.2	19	2	17.8	19	1	17.1	19	2	18.5
20x12	20	21	4	15.6	21	4	15.3	21	4	14.5	21	4	14.9
20x16	14	16	10	11.7	16	10	13.5	16	10	11.7	16	10	13.1
30x8	22	23	2	21.3	23	2	21.4	23	2	19.2	23	2	20.7
40x8	22	23	1	18.0	23	1	18.7	23	1	14.4	23	1	15.1

Ποσοστό εργασιών με προθεσμίες ddper=60%

	NAW	W=100			W=20 R=5			W=5 R=20			W=10 R=10		
NxM	ΕΛ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ
6x8	7	7	1	1.0	7	1	1.0	7	1	1.0	7	1	1.0
8x8	9	10	1	3.4	10	1	3.4	10	1	3.4	10	1	3.4
10x4	16	18	2	14.2	18	2	14.2	17	2	12.5	18	2	14.2
10x8	9	9	8	5.7	9	8	5.5	9	8	5.5	9	8	5.5
10x12	8	10	4	4.7	10	4	4.7	10	4	4.6	10	4	4.7
10x16	5	7	9	0.5	7	9	0.5	7	9	0.6	7	9	0.5
20x4	20	20	1	13.7	20	1	25.6	20	1	19.1	20	1	22.0
20x8	19	20	3	16.9	20	3	17.4	20	2	14.9	20	2	14.8
20x12	9	12	9	13.1	12	9	11.5	11	9	8.9	11	9	10.8
20x16	9	11	13	9.4	10	13	8.7	10	11	6.1	10	13	7.3
30x8	17	21	7	19.5	21	7	24.5	21	7	20.9	21	7	22.4
40x8	19	20	7	19.4	20	7	18.8	20	6	17.8	20	7	21.4

Ποσοστό εργασιών με προθεσμίες ddper=80%

	NAW	W=100			W=20 R=5			W=5 R=20			W=10 R=10		
NxM	ΕΛ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ
6x8	3	4	2	1.5	4	2	1.5	4	2	1.5	4	2	1.5
8x8	3	3	2	0.1	3	2	0.1	3	2	0.1	3	2	0.1
10x4	8	8	1	1.8	8	1	2.0	8	1	2.0	8	1	2.0
10x8	5	5	4	3.5	5	4	3.5	5	4	1.1	5	4	3.8
10x12	2	3	8	4.6	3	8	4.6	3	7	3.8	3	8	4.6
10x16	0	1	5	0.1	1	5	0.1	1	5	0.1	1	5	0.1
20x4	6	6	7	8.1	6	7	6.7	6	5	5.9	6	7	7.5
20x8	3	6	12	7.6	6	12	6.8	3	8	4.3	5	11	6.0
20x12	0	2	12	3.5	1	11	3.1	1	8	2.3	1	8	2.2
20x16	0	3	16	5.4	3	16	5.2	0	15	3.7	2	16	4.5
30x8	3	3	11	6.9	3	10	7.0	3	10	5.6	3	9	6.4
40x8	6	9	14	13.2	8	9	8.8	8	8	6.9	8	8	8.6

Ποσοστό εργασιών με προθεσμίες ddper=100%

	NAW	W=100			W=20 R=5			W=5 R=20			W=10 R=10		
NxM	ΕΛ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ	ΕΛ	ΒΚ	ΠΒΧ
6x8	0	0	2	-0.1	0	2	-0.1	0	2	-0.1	0	2	-0.1
8x8	0	0	4	0.5	0	4	0.5	0	4	0.5	0	4	0.5
10x4	0	0	5	3.7	0	5	3.6	0	5	3.6	0	5	3.6
10x8	0	0	4	0.9	0	4	0.9	0	4	0.9	0	4	0.9
10x12	0	0	7	1.4	0	7	1.4	0	7	1.4	0	7	1.4
10x16	0	0	8	2.5	0	8	2.5	0	8	2.5	0	8	2.5
20x4	0	0	2	0.6	0	2	0.6	0	2	0.6	0	2	0.6
20x8	0	0	7	1.5	0	6	1.0	0	3	0.5	0	6	0.9
20x12	0	0	9	2.9	0	7	2.4	0	6	1.5	0	7	2.3
20x16	0	0	4	0.1	0	4	0.1	0	4	0.1	0	4	0.1
30x8	0	0	6	1.0	0	6	0.8	0	5	0.8	0	5	0.8
40x8	0	0	10	1.4	0	10	1.1	0	9	0.5	0	9	0.5