Υλοποίηση, οπτικοποίηση και πειραματική μελέτη αλγορίθμων για προβλήματα χρονοπρογραμματισμού

Θωμάς Ράμμος

Διπλωματική Εργασία

Επιβλέπων: Χρήστος Νομικός

Ιωάννινα, Φεβρουάριος, 2025

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Ευχαριστίες

Θα ήθελα να εκφράσω τις βαθύτατες ευχαριστίες μου στο Πανεπιστήμιο Ιωαννίνων και ειδικότερα στο τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής, για την πολύτιμη ευκαιρία που μου δόθηκε να σπουδάσω και να αποκτήσω τις γνώσεις και τις δεξιότητες που με οδήγησαν στο να ολοκληρώσω αυτή τη διπλωματική εργασία. Οι πόροι των εργαστηρίων και η υποστήριξη από τους καθηγητές του τμήματος ήταν καθοριστικοί για την εξέλιξή μου ως μηχανικός.

Ιδιαίτερες ευχαριστίες απευθύνω στον καθηγητή Χρήστο Νομικό, του οποίου η καθοδήγηση, η υποστήριξη και η βοήθεια στην κατανόηση της εργασίας υπήρξαν ανεκτίμητες. Χωρίς την αμέριστη στήριξή του και τις πολύτιμες συμβουλές του, η ολοκλήρωση αυτής της εργασίας δεν θα ήταν δυνατή.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου για την ηθική υποστήριξη και την κατανόηση καθ' όλη τη διάρκεια των σπουδών μου. Αν και δεν συνέβαλαν άμεσα στη σύνταξη της εργασίας, η στήριξή τους υπήρξε σημαντική για την επιτυχή ολοκλήρωση των σπουδών μου.

Σας ευχαριστώ όλους θερμά.

Θωμάς Ράμμος

Περίληψη

Στην παρούσα διπλωματική εργασία μελετούμε τρία διακριτά προβλήματα χρονοπρογραμματισμού και υλοποιούμε ισάριθμους αλγορίθμους, ο καθένας με διαφορετικές προδιαγραφές και στόχους. Αρχικά, επικεντρωνόμαστε στο πρόβλημα εκτέλεσης εργασιών σε ακριβώς τρεις αφιερωμένες μηχανές και προτείνουμε έναν προσεγγιστικό αλγόριθμο γραμμικού χρόνου, ο οποίος εγγυάται λόγο προσέγγισης 7/6 σε σχέση με το βέλτιστο makespan. Στη συνέχεια, εξετάζουμε μια πιο γενική εκδοχή του προβλήματος, όπου ο αριθμός των μηχανών είναι μεγαλύτερος, και υλοποιούμε τον αλγόριθμο Longest First (LF). Ο LF εκτελείται σε χρόνο τάξης O(dmlog n) και εξασφαλίζει σταθερό λόγο προσέγγισης, που στην πιο γενική περίπτωση δεν υπερβαίνει το τετραπλάσιο του βέλτιστου, με βελτιώσεις σε ειδικές κατηγορίες γραφημάτων. Τέλος, αντιμετωπίζουμε ένα εντελώς διαφορετικό πρόβλημα, το οποίο στοχεύει στην ελαχιστοποίηση του αριθμού των βαθμονομήσεων (calibrations). Για αυτό το ζήτημα, υλοποιούμε τον αλγόριθμο Preemptive Lazy Binning (PLB), ο οποίος επιλύει ακριβώς το πρόβλημα σε πολυωνυμικό χρόνο ($O(n^2)$), δίχως να αποτελεί προσεγγιστική μέθοδο. Παρουσιάζουμε την πολυπλοκότητα και τις θεωρητικές ιδιότητες κάθε αλγορίθμου και, επιπλέον, εκτελούμε εκτενή πειραματική μελέτη σε ποικίλα σύνολα δεδομένων. Τα αποτελέσματά μας επιβεβαιώνουν ότι οι δύο πρώτοι αλγόριθμοι παρέχουν προσεγγιστικές λύσεις με αποδεδειγμένα χαμηλή απόκλιση από το βέλτιστο, ενώ ο τρίτος αλγόριθμος επιλύει με ακρίβεια το πρόβλημα βαθμονόμησης, επιβεβαιώνοντας τις θεωρητικές προβλέψεις.

Λέξεις Κλειδιά: χρονοπρογραμματισμός, προσεγγιστικοί αλγόριθμοι, makespan, Longest First, Preemptive Lazy Binning, πολυπλοκότητα, βαθμονομήσεις, πειραματική μελέτη

Abstract

In this thesis, we study three distinct scheduling problems and implement three corresponding algorithms, each with different specifications and objectives. Initially, we focus on the problem of executing tasks on exactly three dedicated machines and propose a linear-time approximation algorithm that guarantees an approximation ratio of 7/6 compared to the optimal makespan. Subsequently, we examine a more general version of the problem, where the number of machines is greater, and we implement the Longest First (LF) algorithm. LF runs in O(dmlog n) time and ensures a constant approximation ratio, which, in the most general case, does not exceed four times the optimal, with improvements in specific classes of graphs. Finally, we address an entirely different problem that aims to minimize the number of calibrations. For this problem, we implement the Preemptive Lazy Binning (PLB) algorithm, which solves the problem exactly in polynomial time ($O(n^2)$) without being an approximation method. We present the complexity and theoretical properties of each algorithm and additionally conduct an extensive experimental study on various datasets. Our results confirm that the first two algorithms provide approximate solutions with provably low deviation from the optimal, while the third algorithm precisely solves the calibration problem, validating the theoretical predictions.

Keywords: scheduling, approximation algorithms, makespan, Longest First, Preemptive Lazy Binning, complexity, calibrations, experimental study

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1. Εισαγωγή

1.1 Κίνητρα και Στόχοι της Εργασίας

Στην εργασία μας ασχολούμαστε με την υλοποίηση και τη βασική σύγκριση τριών προσεγγιστικών αλγορίθμων που έχουν προταθεί στη βιβλιογραφία για προβλήματα χρονοπρογραμματισμού. Επιλέξαμε αυτό το αντικείμενο επειδή ο χρονοπρογραμματισμός εμφανίζει ιδιαίτερο ενδιαφέρον σε εφαρμογές με αυξημένη υπολογιστική πολυπλοκότητα, όπου η εύρεση ακριβώς βέλτιστης λύσης δεν είναι εφικτή σε εύλογο χρόνο. Οι προσεγγιστικοί αλγόριθμοι έρχονται να καλύψουν αυτό το κενό, επιτυγχάνοντας λύσεις αρκετά κοντά στο βέλτιστο, με σαφή θεωρητικά όρια και μειωμένο χρόνο εκτέλεσης.

Γνωρίζουμε ότι πολλά προβλήματα χρονοπρογραμματισμού είναι NP-πλήρη και ως εκ τούτου απαιτούν μεθόδους που ανταλλάσσουν ένα μέρος της ακρίβειας με σημαντική βελτίωση στον απαιτούμενο χρόνο υπολογισμού. Σκοπεύουμε να αναδείξουμε την πρακτική αξία τριών συγκεκριμένων αλγορίθμων, εξετάζοντας (α) τα θεωρητικά τους όρια, όπως ο παράγοντας προσέγγισης και η πολυπλοκότητά τους, και (β) τη συμπεριφορά τους σε διάφορα μεγέθη συνόλων εργασιών.

Συνοψίζουμε τους κύριους στόχους μας ως εξής:

- **Ανάλυση Θεωρητικών Χαρακτηριστικών**: Εστιάζουμε στον παράγοντα προσέγγισης κάθε αλγορίθμου, καθώς και στον χρόνο εκτέλεσής του, με βάση την πολυπλοκότητα που αναφέρεται στη βιβλιογραφία.
- **Πειραματική Μελέτη**: Τρέχουμε πειράματα για διαφορετικά μεγέθη συνόλων εργασιών, ώστε να εξετάσουμε πώς κλιμακώνεται ο χρόνος εκτέλεσης και ποια είναι η ποιότητα των λύσεων στην πράξη.
- **Οπτικοποίηση**: Παρουσιάζουμε μερικές γραφικές αναπαραστάσεις που διευκολύνουν την κατανόηση της λειτουργίας των αλγορίθμων και των αποτελεσμάτων.

Μέσα από αυτήν τη μελέτη, επιδιώκουμε να κατανοήσουμε σε ποιο βαθμό επιβεβαιώνονται στην πράξη οι θεωρητικές εγγυήσεις των αλγορίθμων και ποιες είναι οι συνθήκες στις οποίες ανταποκρίνονται αποτελεσματικότερα.

1.2 Στόχος και Αντικείμενο της Εργασίας

Στη συγκεκριμένη ενότητα αναλύουμε τους στόχους και το αντικείμενο της εργασίας, εστιάζοντας στη μελέτη τριών διακριτών προβλημάτων χρονοπρογραμματισμού και των αντίστοιχων αλγορίθμων που υλοποιούμε. Συγκεκριμένα, εξετάζουμε έναν προσεγγιστικό αλγόριθμο για τον χρονοπρογραμματισμό σε τρεις αφιερωμένες μηχανές. Κάθε εργασία απαιτεί υποσύνολο από μία έως τρεις μηχανές για να εκτελεστεί ταυτόχρονα, με τελικό στόχο την ελαχιστοποίηση του makespan (τον συνολικό χρόνο ολοκλήρωσης όλων των εργασιών). Ο αλγόριθμός μας τρέχει σε γραμμικό χρόνο ως προς το πλήθος των εργασιών και επιτυγχάνει λόγο προσέγγισης 7/6 συγκριτικά με τη βέλτιστη λύση. Ακόμη τον αλγόριθμο Longest First (LF), ο οποίος εφαρμόζεται σε ένα πιο γενικό μοντέλο διαγνωστικού χρονοπρογραμματισμού, όπου οι εργασίες εκτελούνται σε μεγαλύτερο πλήθος μηχανών. Ο LF επιλέγει κάθε φορά την εργασία με τη μεγαλύτερη διάρκεια και τη δρομολογεί στην πρώτη διαθέσιμη θέση. Δίνει σταθερό λόγο προσέγγισης (έως και τέσσερις φορές το βέλτιστο στη γενική περίπτωση, με βελτιωμένα όρια για συγκεκριμένες κατηγορίες γραφημάτων) και εκτελείται σε πολυωνυμικό χρόνο, καθιστώντας το κατάλληλο για προβλήματα μεγάλης κλίμακας. Επίσης τον αλγόριθμο Preemptive Lazy Binning (PLB), ο οποίος απευθύνεται σε ένα διαφορετικό πλαίσιο, όπου επιδιώκουμε να ελαχιστοποιήσουμε τον αριθμό των βαθμονομήσεων (calibrations) σε μία μηχανή. Ο PLB επιτρέπει την προ-διακοπή εργασιών, εκτελείται σε πολυωνυμικό χρόνο ($O(n^2)$) και παρέχει ακριβή λύση, σε αντίθεση με τους δύο πρώτους που είναι προσεγγιστικοί. Αφού παρουσιάσουμε τις βασικές έννοιες του χρονοπρογραμματισμού και το θεωρητικό υπόβαθρο στις προσεγγιστικές τεχνικές, αναλύουμε διεξοδικά τη λειτουργία των τριών αυτών αλγορίθμων και τα ιδιαίτερα χαρακτηριστικά τους. Στη συνέχεια, υλοποιούμε και δοκιμάζουμε πειραματικά καθέναν από αυτούς, αξιολογώντας την απόδοσή τους σε διαφορετικά μεγέθη δεδομένων. Ολοκληρώνουμε με μια συνολική αποτίμηση των ευρημάτων μας

Κεφάλαιο 2. Βασικές Έννοιες

2.1 Βασικές Έννοιες στο Scheduling

Στην εργασία μας, εστιάζουμε στον χρονοπρογραμματισμό, που αποτελεί τον πυρήνα της διαχείρισης πόρων σε πολλά υπολογιστικά συστήματα. Σε αυτήν την ενότητα, εξηγούμε τις βασικές έννοιες που χρησιμοποιούμε και θέτουμε τα θεμέλια για την περαιτέρω ανάλυση των προσεγγιστικών αλγορίθμων που υλοποιούμε.

Ορισμοί και Χαρακτηριστικά

- **Εργασία (Job):**
  Θεωρούμε ότι μια εργασία αποτελεί μια αυτόνομη ενότητα που πρέπει να ολοκληρωθεί μέσα σε έναν καθορισμένο χρόνο. Κάθε εργασία χαρακτηρίζεται από τον χρόνο επεξεργασίας που απαιτείται για την ολοκλήρωσή της και, σε ορισμένα μοντέλα, από τις απαιτήσεις σε συγκεκριμένους πόρους (π.χ. συγκεκριμένες μηχανές).
- **Μηχανή (Machine):**
  Ορίζουμε μια μηχανή ως τον διαθέσιμο πόρο όπου εκτελούνται οι εργασίες. Εμείς θεωρούμε ότι κάθε μηχανή μπορεί να επεξεργαστεί μόνο μία εργασία τη φορά, γεγονός που θέτει τον βασικό περιορισμό στη διαχείριση της σειράς εκτέλεσης.
- **Makespan:**
  Ο συνολικός χρόνος που απαιτείται για την ολοκλήρωση όλων των εργασιών ονομάζεται makespan. Στόχος μας είναι να ελαχιστοποιήσουμε αυτόν τον χρόνο, επιτυγχάνοντας την καλύτερη δυνατή κατανομή των εργασιών στους διαθέσιμους πόρους.

Μοντέλα και Περιορισμοί

Καθώς προχωράμε στην ανάλυση, διακρίνουμε διάφορα μοντέλα βασιζόμενοι στους περιορισμούς που επιβάλλουμε:

- **Μη Διακοπτόμενες Εργασίες:**
  Σε ορισμένα προβλήματα, οι εργασίες δεν επιτρέπεται να διακοπούν κατά τη διάρκεια της εκτέλεσής τους. Μόλις ξεκινήσει μια εργασία, η εκτέλεσή της συνεχίζεται αδιάκοπα μέχρι την ολοκλήρωσή της, χωρίς παρεμβολές.

- **Επιτρεπόμενη Προ-διακοπή (Preemption):**
  Σε ορισμένα μοντέλα, όπως όταν εφαρμόζουμε τον αλγόριθμο Preemptive Lazy Binning, επιτρέπουμε την προ-διακοπή. Δηλαδή, μπορούμε να διακόψουμε μια εργασία και να την συνεχίσουμε αργότερα, κάτι που μας δίνει μεγαλύτερη ευελιξία αλλά επίσης αυξάνει την πολυπλοκότητα της λύσης.
- **Απαιτήσεις Πόρων:**
  Σε κάποια προβλήματα, οι εργασίες απαιτούν συγκεκριμένους πόρους ή την ταυτόχρονη επεξεργασία από πολλαπλές μηχανές. Εμείς ορίζουμε τους αντίστοιχους τύπους εργασίας ώστε να καθορίσουμε σαφώς σε ποιες μηχανές μπορεί να εκτελεστεί κάθε εργασία.

Αναπαράσταση με DAG

Για να κατανοήσουμε τις εξαρτήσεις μεταξύ των εργασιών, χρησιμοποιούμε την αναπαράσταση του προγράμματος ως Directed Acyclic Graph (DAG). Σε αυτή τη δομή:

- Κάθε κορυφή αντιπροσωπεύει μια εργασία (ή ένα σύνολο εργασιών του ίδιου τύπου).
- Κάθε ακμή καθορίζει τη σειρά εκτέλεσης, εξασφαλίζοντας ότι δεν ξεκινά μια εργασία πριν ολοκληρωθεί εκείνη που προηγείται.

Πολυπλοκότητα και NP-Πληρότητα

Εμείς κατανοούμε ότι πολλά προβλήματα χρονοπρογραμματισμού ανήκουν στην κατηγορία των NP-πληρών προβλημάτων. Αν και μπορούμε να επαληθεύσουμε μια δεδομένη λύση σε πολυωνυμικό χρόνο, η εύρεση της βέλτιστης λύσης απαιτεί χρόνο που αυξάνεται εκθετικά με το μέγεθος της εισόδου. Για το λόγο αυτό, εμείς επιλέγουμε προσεγγιστικές μεθόδους που εγγυώνται λύσεις εντός ενός προκαθορισμένου ορίου από το βέλτιστο αποτέλεσμα, επιτυγχάνοντας έτσι ένα αποδεκτό συμβιβασμό μεταξύ απόδοσης και υπολογιστικού κόστους.

2.2 Προσεγγιστικοί Αλγόριθμοι

Σε αυτή την ενότητα, επεκτείνουμε την ανάλυσή μας εστιάζοντας στον ρόλο και τη σημασία τους στα προβλήματα χρονοπρογραμματισμού που αντιμετωπίζουμε.

Εμείς επιλέγουμε προσεγγιστικούς αλγόριθμους για δύο κύριους λόγους:

- **Ποιότητα Λύσης:**

Μέσω του παράγοντα προσέγγισης, μπορούμε να εγγυηθούμε ότι η λύση που παράγει κάθε αλγόριθμος απέχει το πολύ έναν συγκεκριμένο πολλαπλασιαστή (π.χ. 7/6) από τη βέλτιστη λύση. Αυτή η εγγύηση μας επιτρέπει να αξιολογούμε με ακρίβεια την απόδοση των μεθόδων μας, διασφαλίζοντας ότι οι λύσεις είναι αρκετά «καλές» για πρακτική χρήση.

- **Υπολογιστική Απόδοση:**

  Σε αντίθεση με τους ακριβείς αλγόριθμους, οι προσεγγιστικοί αλγόριθμοι μας επιτρέπουν να επιλύουμε σύνθετα και μεγάλα προβλήματα σε πολυωνυμικό χρόνο, καθιστώντας τους πρακτικούς για εφαρμογές όπου ο χρόνος εκτέλεσης είναι κρίσιμος.

Εμείς, λοιπόν, αξιοποιούμε αυτές τις μεθοδολογίες ώστε να αντιμετωπίσουμε τα NP-πλήρη προβλήματα του χρονοπρογραμματισμού, συνδέοντας τη θεωρητική μας ανάλυση με τις εφαρμογές των αλγορίθμων που υλοποιούμε (π.χ. scheduling σε τρεις αφιερωμένες μηχανές, Longest First, Preemptive Lazy Binning). Αυτή η προσέγγιση μας επιτρέπει να εξασφαλίσουμε έναν αποδεκτό συμβιβασμό μεταξύ ποιότητας λύσης και υπολογιστικού κόστους, κάτι που θα επανεμφανίζεται και στα επόμενα κεφάλαια της εργασίας.

2.3 NP-Προβλήματα Βελτιστοποίησης και NP-Πληρότητα

Σε αυτήν την ενότητα, εμβαθύνουμε στη φύση των προβλημάτων βελτιστοποίησης στον τομέα του χρονοπρογραμματισμού και εξετάζουμε την έννοια της NP-πληρότητας, στοιχεία που μας υποκινούν να επιλέξουμε προσεγγιστικές μεθόδους.

Εμείς γνωρίζουμε ότι:

- **Τα Προβλήματα Βελτιστοποίησης:**

  Στοχεύουν στην εύρεση της «καλύτερης» λύσης από ένα πεπερασμένο σύνολο εφικτών λύσεων, βάσει μιας αντικειμενικής συνάρτησης (π.χ. ελαχιστοποίηση του makespan). Αν και σε ορισμένα προβλήματα βελτιστοποίησης μπορούμε να επαληθεύσουμε μια λύση σε πολυωνυμικό χρόνο, η ανεύρεση της βέλτιστης λύσης μπορεί να απαιτεί χρόνο που αυξάνεται εκθετικά με το μέγεθος της εισόδου.

- **Μετατροπή σε Πρόβλημα Απόφασης:**

  Προκειμένου να αξιολογήσουμε τη δυσκολία ενός προβλήματος βελτιστοποίησης, συχνά το μετατρέπουμε στην αντίστοιχη μορφή προβλήματος απόφασης. Σε αυτή την περίπτωση, θέτουμε ένα όριο (π.χ. το makespan να μην υπερβαίνει μία συγκεκριμένη τιμή) και εξετάζουμε αν υπάρχει λύση που πληροί τον σχετικό περιορισμό. Μέσω αυτής της μετατροπής, μπορούμε να χρησιμοποιήσουμε γνωστές τεχνικές ανάλυσης της πολυπλοκότητας των προβλημάτων απόφασης και να αιτιολογήσουμε την ανάγκη για προσεγγιστικές ή εναλλακτικές μεθόδους, όταν το πρόβλημα απόφασης αποδεικνύεται υπολογιστικά δύσκολο.

- **NP-Πληρότητα**

  Ένα πρόβλημα απόφασης ονομάζεται NP-πλήρες όταν:

  1. Ανήκει στην κλάση NP, δηλαδή μπορούμε να επαληθεύσουμε σε πολυωνυμικό χρόνο αν μια υποψήφια λύση είναι έγκυρη.
  2. Οποιοδήποτε άλλο πρόβλημα της κλάσης NP μπορεί να μετασχηματιστεί πολυωνυμικά σε αυτό.

  Ο παραπάνω ορισμός υποδηλώνει ότι δεν υπάρχει γνωστός αλγόριθμος πολυωνυμικού χρόνου που να επιλύει οποιοδήποτε NP-πλήρες πρόβλημα σε όλη τη γενικότητά του. Πολυάριθμα προβλήματα χρονοπρογραμματισμού κατατάσσονται σε αυτή την κατηγορία, γεγονός που καθιστά την αναζήτηση ακριβών λύσεων μη πρακτική για μεγάλα μεγέθη εισόδου.

Η γνώση της NP-πληρότητας μας οδηγεί στην επιλογή των προσεγγιστικών αλγορίθμων, καθώς κατανοούμε ότι η εύρεση της ακριβούς λύσης σε NP-πλήρη προβλήματα είναι πρακτικά αδύνατη για περιπτώσεις μεγάλης κλίμακας. Εμείς, συνεπώς, χρησιμοποιούμε προσεγγιστικές μεθόδους που εγγυώνται ότι η λύση θα βρίσκεται εντός ενός προκαθορισμένου ορίου από τη βέλτιστη, συνδυάζοντας έτσι τη θεωρητική μας γνώση με πρακτικές εφαρμογές στα προβλήματα χρονοπρογραμματισμού που θα αναλύσουμε στα επόμενα κεφάλαια.

Κεφάλαιο 3.

Αναλυτική Παρουσίαση των Αλγορίθμων

3.1 Αλγόριθμος για Scheduling σε Τρεις Αφιερωμένες Μηχανές

Σε αυτήν την ενότητα, παρουσιάζουμε τον προσεγγιστικό αλγόριθμο που υλοποιούμε για το πρόβλημα του scheduling σε τρεις αφιερωμένες μηχανές. Στόχος μας είναι να ελαχιστοποιήσουμε το makespan (τον συνολικό χρόνο ολοκλήρωσης όλων των εργασιών), με την προϋπόθεση ότι κάθε μηχανή μπορεί να εκτελεί μόνο μία εργασία τη φορά και ότι τοποθετούμε τις εργασίες με τρόπο που αποτρέπει την άσκοπη αναμονή.

3.1.1 Περιγραφή του Προβλήματος

- **Τρεις Μηχανές**

  Διαθέτουμε τρεις μηχανές {m1,m2,m3}. Κάθε μηχανή μπορεί να επεξεργαστεί μία μόνο εργασία κάθε φορά, χωρίς επικαλύψεις ή διακοπή.

- **Σύνολο Εργασιών**

  Έχουμε n εργασίες $\{J_1,\ldots,J_n\}$. Κάθε εργασία J_i έχει:

  1. Χρόνο επεξεργασίας p_i.
  2. Τύπο εργασίας S_i, δηλαδή το υποσύνολο των μηχανών {m1,m2,m3} στις οποίες πρέπει να εκτελεστεί ταυτόχρονα για p_i μονάδες χρόνου.

- **Περιορισμοί και Στόχος**
  - Δεν υπάρχουν χρόνοι εκκίνησης , προθεσμίες ή προτεραιότητες.
  - Επιδιώκουμε να ελαχιστοποιήσουμε το makespan: τον χρόνο μέχρι να ολοκληρωθούν όλες οι εργασίες.

Δεδομένου ότι το συγκεκριμένο πρόβλημα είναι ισχυρά NP-πλήρες, επιλέγουμε έναν προσεγγιστικό αλγόριθμο που εγγυάται ότι η λύση που βρίσκουμε δεν θα ξεπερνά τη βέλτιστη λύση κατά περισσότερο από 7/6.

3.1.2 Κατασκευή του DAG για Διαφορετικά Schedules

Για να περιγράψουμε ένα πρόγραμμα (schedule), χρησιμοποιούμε ένα Directed Acyclic Graph (DAG):

- **Κόμβοι:**

  Κάθε κόμβος αντιστοιχεί σε ένα «σύνολο εργασιών» που απαιτούν το ίδιο ακριβώς υποσύνολο μηχανών (π.χ. {m1}, {m2}, {m1,m2}, κ.λπ.).

- **Ακμές:**

  Εισάγουμε ακμές μεταξύ κόμβων που μοιράζονται τουλάχιστον μία μηχανή, ορίζοντας ποιος κόμβος προηγείται. Με αυτόν τον τρόπο, διασφαλίζουμε ότι δεν γίνεται κοινή χρήση της ίδιας μηχανής ταυτόχρονα.

Ο αλγόριθμός μας παράγει 18 διαφορετικά schedules και επιλέγει τελικά εκείνο με τον μικρότερο makespan. Η βασική φιλοσοφία είναι ότι οι εργασίες με τον ίδιο τύπο (δηλαδή το ίδιο υποσύνολο μηχανών) εκτελούνται διαδοχικά, έτσι ώστε να μην παρεμβάλλονται περιττά κενά και να ελαχιστοποιείται ο χρόνος ολοκλήρωσης. Σε ορισμένες περιπτώσεις (βλ. C και D), διαχωρίζουμε τις εργασίες μίας μόνο μηχανής σε δύο ομάδες (Long – L, Short – S), ώστε να περιορίσουμε περαιτέρω το makespan.

Διακρίνουμε δύο κύριες κατηγορίες προγραμμάτων:

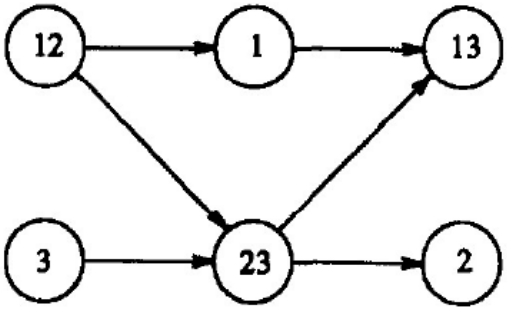
1. **Κανονικά Προγράμματα (A, B):**

   Στην κατηγορία A (A_1, A_2, A_3) και την κατηγορία B (B_1, B_2, B_3), οι εργασίες κάθε τύπου τοποθετούνται η μία πίσω από την άλλη (δηλαδή χωρίς παρεμβολή εργασιών διαφορετικού τύπου). Η εναλλαγή των μηχανών m_1, m_2, m_3 (π.χ. ανταλλαγή ρόλων μεταξύ m_1 και m_2) παράγει παραλλαγές των ίδιων βασικών προγραμμάτων. Συγκεκριμένα, οι A_2 και A_3 προκύπτουν από την A_1 με την αντικατάσταση (permutation) των μηχανών, ενώ αντίστοιχα οι B_2 και B_3 προκύπτουν από τη B_1.

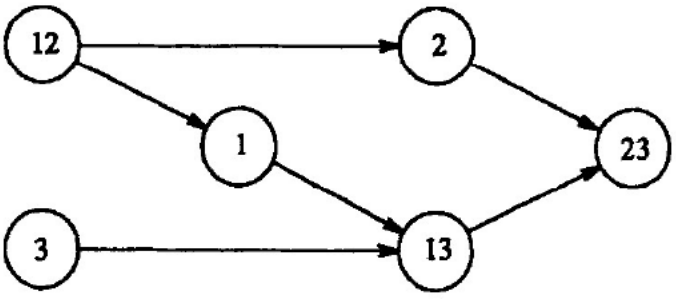
2. **Επέκταση Πέραν των Κανονικών (C, D):**

   Περιλαμβάνουν 12 στρατηγικές (C_{ij},D_{ij}), όπου {i,j} είναι υποσύνολο του {1,2,3}. Σε αυτές τις στρατηγικές, όταν έχουμε μονομηχανιακές εργασίες (π.χ. τύπος {m_1}), τις διαχωρίζουμε σε δύο υποσύνολα: L (Long) και S (Short), ανάλογα με τη διάρκειά τους. Με αυτόν τον τρόπο, μπορούμε να προγραμματίσουμε τις «μεγάλες» εργασίες ξεχωριστά από τις «μικρές» και, συχνά, να επιτύχουμε μικρότερο makespan σε σχέση με τα απλά κανονικά προγράμματα.

Εισαγωγή Εικόνων Schedules

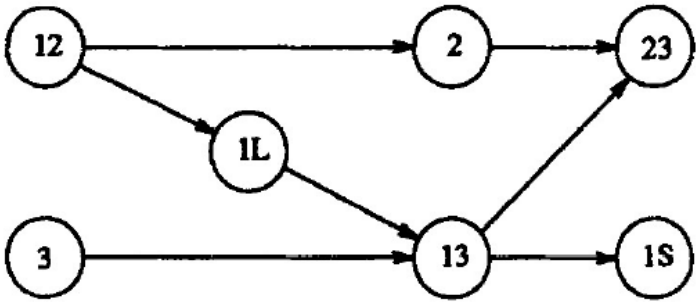


Εικόνα 1: Schedule A_1. Στην εικονογραφημένη αναπαράσταση, κάθε κόμβος αντιστοιχεί σε ένα σύνολο εργασιών που απαιτεί συγκεκριμένο υποσύνολο μηχανών, ενώ τα βέλη δείχνουν τις αλληλεξαρτήσεις λόγω κοινής χρήσης κάποιας μηχανής. Οι εργασίες του κόμβου «12» (μηχανές m_1 και m_2) εκτελούνται παράλληλα με τις εργασίες του κόμβου «3» (μόνο m_3), καθώς δεν υπάρχει επικαλυπτόμενη μηχανή. Μετά την ολοκλήρωση του «12», ξεκινούν οι εργασίες «1» (μόνο m_1), ενώ ταυτόχρονα, μόλις ολοκληρωθούν οι εργασίες «3», μπορούν να αρχίσουν εκείνες του κόμβου «13» (m_1 και m_3). Στη συνέχεια, όταν έχουν τελειώσει τόσο οι εργασίες «12» όσο και «3», απελευθερώνεται η μηχανή m_2 και ξεκινούν οι εργασίες «23» (m_2 και m_3). Τέλος, αφού ολοκληρωθούν οι κόμβοι «1» και «13» (που δεσμεύουν τη m_1), αλλά και ο κόμβος «23» (που δεσμεύει τη m_2), εκτελούνται οι εργασίες «2» (μόνο m_2). Με αυτόν τον τρόπο, το γράφημα απεικονίζει τη σειρά με την οποία προγραμματίζονται οι διάφορες ομάδες εργασιών, αποτρέποντας την ταυτόχρονη χρήση της ίδιας μηχανής και επιτρέποντας την παράλληλη εκτέλεση όπου δεν υπάρχει σύγκρουση.

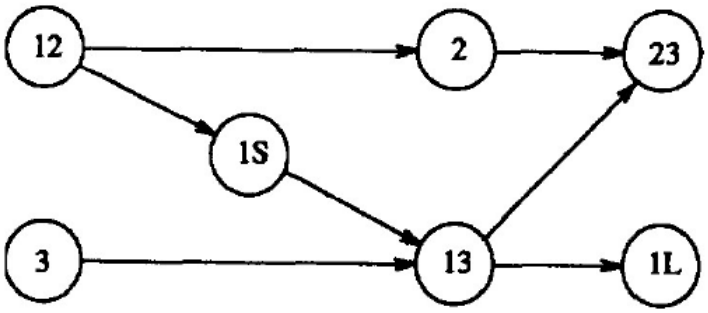


Εικόνα 2: Schedule B_1. Στην παραπάνω αναπαράσταση, κάθε κόμβος δηλώνει ένα σύνολο εργασιών που χρησιμοποιεί συγκεκριμένες μηχανές, ενώ τα βέλη καθορίζουν ποιες ομάδες εργασιών πρέπει να ολοκληρωθούν πριν ξεκινήσει κάποια άλλη, λόγω κοινής χρήσης μηχανών. Συγκεκριμένα, ο κόμβος «12» περιλαμβάνει εργασίες που απαιτούν ταυτόχρονα τις μηχανές m_1 και m_2, οπότε όταν αυτές ολοκληρωθούν, η μηχανή m_2 απελευθερώνεται και μπορούμε να προγραμματίσουμε τις εργασίες του

κόμβου «2», οι οποίες χρειάζονται αποκλειστικά τη m_2. Μόλις ολοκληρωθεί ο κόμβος «2», ξεκινούν οι εργασίες «23», καθώς μοιράζονται τη μηχανή m_2. Παράλληλα, οι εργασίες «1» (μηχανή m_1) και «3» (μηχανή m_3) εκτελούνται ανεξάρτητα, αλλά η ομάδα «13» (μηχανές m_1 και m_3) μπορεί να προγραμματιστεί μόνο αφού ολοκληρωθούν τόσο οι εργασίες «1» όσο και οι εργασίες «3». Τέλος, ο κόμβος «23» απαιτεί τη μηχανή m_3, οπότε δεν μπορεί να ξεκινήσει αν δεν τελειώσει ο κόμβος «13», και επίσης μοιράζεται τη m_2 με τον κόμβο «2». Με αυτόν τον τρόπο, το γράφημα απεικονίζει τη διαδοχική αλληλεξάρτηση των ομάδων εργασιών, αποτρέποντας την ταυτόχρονη δέσμευση της ίδιας μηχανής και αναδεικνύοντας τη δυνατότητα εκτέλεσης σε παράλληλα «μονοπάτια» όπου δεν υπάρχει σύγκρουση.



Εικόνα 3: Schedule C_{12}. Στο συγκεκριμένο γράφημα, κάθε κόμβος αντιστοιχεί σε ένα σύνολο εργασιών που δεσμεύει ορισμένες μηχανές, ενώ τα βέλη υποδεικνύουν τη σειρά εκτέλεσης με βάση τη μηχανή που μοιράζονται. Για παράδειγμα, ο κόμβος «12» (μηχανές m_1 και m_2) πρέπει να ολοκληρωθεί προτού ξεκινήσει ο κόμβος «2» (μόνο m_2), επειδή και οι δύο δεσμεύουν τη m_2. Στη συνέχεια, από τον κόμβο «2» περνάμε στον κόμβο «23» (μηχανές m_2 και m_3), καθώς το σύνολο εργασιών που χρησιμοποιεί τη m_2 δεν μπορεί να εκτελείται παράλληλα με άλλο που επίσης χρειάζεται τη m_2. Παράλληλα, ο κόμβος «1L» (εργασίες μεγάλης διάρκειας που απαιτούν μόνο τη m_1) και ο κόμβος «3» (εργασίες που απαιτούν μόνο τη m_3) εκτελούνται ανεξάρτητα, εφόσον δεν μοιράζονται μηχανές. Ωστόσο, ο κόμβος «13» (m_1 και m_3) μπορεί να ξεκινήσει μόνο όταν ολοκληρωθούν τόσο ο «1L» όσο και ο «3», αφού χρειάζεται και τις δύο μηχανές m_1 και m_3. Τέλος, ο κόμβος «1S» (εργασίες μικρής διάρκειας για τη m_1) εκτελείται αφού ολοκληρωθεί ο κόμβος «13», δεδομένου ότι μοιράζονται τη m_1. Η βασική διαφορά από τα απλά «κανονικά» προγράμματα είναι ότι οι εργασίες που απαιτούν μία μόνο μηχανή (εδώ m_1) έχουν διαχωριστεί σε «1L» και «1S», δηλαδή σε μεγάλες και μικρές, με στόχο τη βέλτιστη αξιοποίηση των μηχανών και τη μείωση του συνολικού χρόνου (makespan).



Εικόνα 4: Schedule D_{12}. Στην παραπάνω αναπαράσταση, κάθε κόμβος δηλώνει ένα σύνολο εργασιών που δεσμεύει συγκεκριμένες μηχανές, ενώ τα βέλη ορίζουν τη διαδοχή με βάση τις κοινές μηχανές. Ο κόμβος «12» (μηχανές m_1 και m_2) ολοκληρώνεται πριν τον κόμβο «2» (μόνο m_2), και στη συνέχεια ο «2» προηγείται του «23» (m_2 και m_3), καθώς όλα αυτά τα σύνολα εργασιών μοιράζονται τη μηχανή m_2. Παράλληλα, ο κόμβος «3» (μόνο m_3) μπορεί να εκτελεστεί ανεξάρτητα, αλλά το σύνολο «13» (m_1 και m_3) ξεκινά μόνο όταν τελειώσουν τόσο οι εργασίες «3» όσο και εκείνες που δεσμεύουν τη m_1. Εδώ, οι μονομηχανιακές εργασίες για τη m_1 χωρίζονται σε «1S» (σύντομης διάρκειας) και «1L» (μεγαλύτερης διάρκειας). Οι «1S» προηγούνται του «13», ώστε να εκμεταλλευτούμε τα κενά διαστήματα της m_1 πριν δεσμευτεί παράλληλα με τη m_3, ενώ οι «1L» εκτελούνται μετά το «13», όταν η m_1 έχει απελευθερωθεί. Με αυτόν τον τρόπο, επιτυγχάνουμε αποτελεσματικότερη κατανομή του φόρτου στη m_1, αποτρέποντας την ταυτόχρονη δέσμευση μηχανών και αξιοποιώντας την παράλληλη εκτέλεση όπου δεν υπάρχει σύγκρουση.

3.1.3 Παράδειγμα

Παραθέτουμε ένα απλό παράδειγμα εργασιών για να δείξουμε πώς υπολογίζουμε το makespan και επιλέγουμε το καλύτερο schedule.

Είσοδος:

M1 : 26

M2 : 25

M3 : 26

M1 : 27

M2 : 26

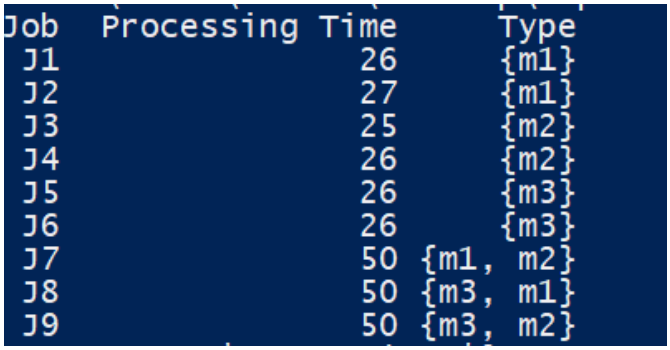
M3 : 26

M1 M2 : 50

M1 M3 : 50

M2 M3 : 50

- Τύπος {m1}: συνολικός χρόνος = 53
- Τύπος {m2}: συνολικός χρόνος = 51
- Τύπος {m3}: συνολικός χρόνος = 52
- Τύπος {m1,m2}: συνολικός χρόνος = 50
- Τύπος {m1,m3}: συνολικός χρόνος = 50
- Τύπος {m2,m3}: συνολικός χρόνος = 50

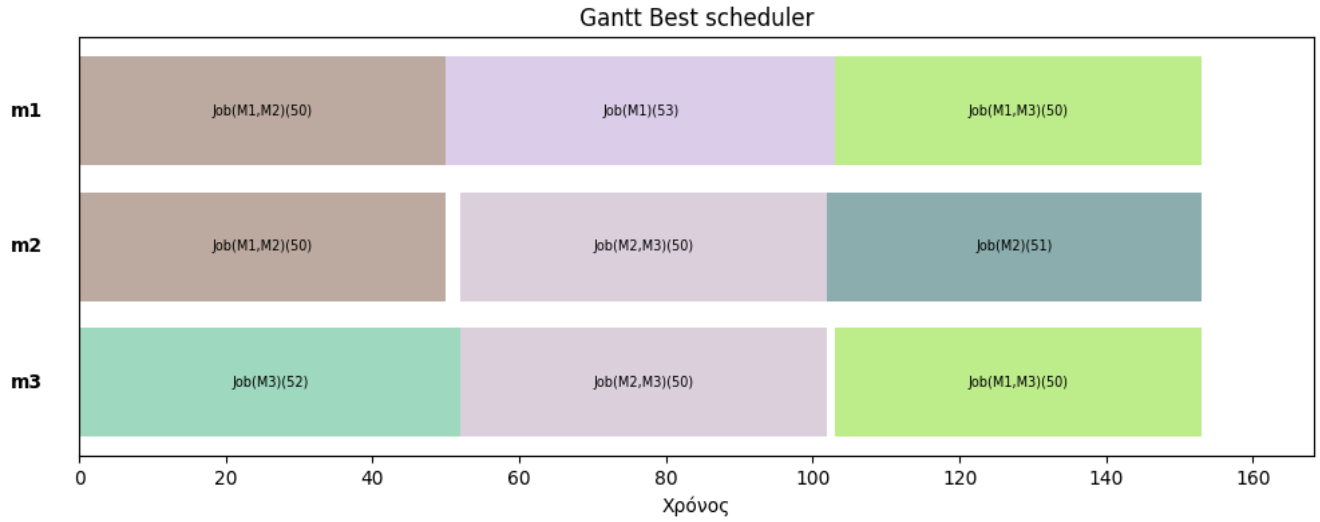


Job	Processing Time	Type
J1	26	{m1}
J2	27	{m1}
J3	25	{m2}
J4	26	{m2}
J5	26	{m3}
J6	26	{m3}
J7	50	{m1, m2}
J8	50	{m3, m1}
J9	50	{m3, m2}

Εικόνα 6: Πίνακας εργασιών για κάθε μηχανή

Εφαρμόζουμε τον αλγόριθμό μας, κατασκευάζουμε το DAG για καθεμία από τις 18 στρατηγικές και υπολογίζουμε το makespan. Οι τιμές διαφέρουν στο makespan για κάθε στρατηγική οπότε επιλέγουμε τελικά τη στρατηγική που δίνει το ελάχιστο makespan. Στο συγκεκριμένο παράδειγμα η καλύτερη στρατηγική είναι η A_1 με makespan =153.

Παράδειγμα εκτέλεσης των εργασιών για το schedule A_1



Εικόνα 5: Scheduler A_1

Το βασικό πλεονέκτημα του αλγορίθμου έγκειται στο ότι λαμβάνουμε υπόψη όλες τις πιθανές «κρίσιμες» διατάξεις εργασιών (A, B, C, D), συμπεριλαμβανομένου του διαχωρισμού των μονομηχανιακών εργασιών σε L και S. Με αυτόν τον τρόπο ελαχιστοποιούμε τις «κακές» περιπτώσεις χρονισμού που θα μπορούσαν να επιδεινώσουν το makespan και διασφαλίζουμε ότι τουλάχιστον μία από τις 18 στρατηγικές θα πετυχαίνει makespan το πολύ 7/6 φορές μεγαλύτερο από το ιδανικό (βέλτιστο).

3.2 Longest First (LF) Algorithm

3.2.1 Περιγραφή

Στην εργασία μας αντιμετωπίζουμε το πρόβλημα του Diagnostic Test Scheduling, το οποίο αφορά τον προγραμματισμό διαγνωστικών δοκιμών σε πολυϋπολογιστικά συστήματα, με σκοπό να ελαχιστοποιήσουμε το συνολικό χρόνο ολοκλήρωσης (makespan). Οι διαγνωστικές δοκιμές χρησιμεύουν για την ανίχνευση σφαλμάτων μεταξύ των υπολογιστικών μονάδων και μοντελοποιούνται ως γράφημα G = (V, E), όπου οι κόμβοι αντιπροσωπεύουν τις μονάδες και οι ακμές τα tests, με κάθε ακμή να συνοδεύεται από έναν θετικό χρόνο εκτέλεσης l(e). Επιπρόσθετα, εφαρμόζουμε περιορισμούς όπως το ότι κάθε μονάδα μπορεί να εκτελεί μία μόνο δοκιμή τη φορά και κάθε δοκιμή πρέπει να ολοκληρώνεται χωρίς διακοπή, ώστε να διασφαλίζουμε ότι το

πρόγραμμα εκτέλεσης ανταποκρίνεται στις απαιτήσεις του προβλήματος που είναι NP-πλήρες.

Εμείς υλοποιούμε τον αλγόριθμο Longest First (LF) χρησιμοποιώντας μια στρατηγική άπληστης επιλογής, όπου αρχικά ταξινομούμε τις ακμές του γραφήματος κατά φθίνουσα σειρά βάσει του χρόνου εκτέλεσης με τη χρήση του heap sort, που έχει πολυπλοκότητα O(mlog m). Στη συνέχεια, για κάθε δοκιμή, εξετάζουμε μέχρι d γειτονικές δοκιμές – όπου d είναι ο μέγιστος βαθμός του γραφήματος – για να εντοπίσουμε το πρώτο διαθέσιμο χρονικό διάστημα, πράγμα που προσδίδει πολυπλοκότητα O(dm). Συνολικά, ο αλγόριθμος μας εκτελείται με πολυπλοκότητα O(m log m + dm), η οποία μπορεί να εκφραστεί ως O(dm log n) και, δεδομένου ότι $m \leq n^2$, καταλήγουμε σε χειρότερη πολυπλοκότητα της τάξης $O(n^2 \log n)$. Με αυτό τον τρόπο, διασφαλίζουμε μια αποδοτική και τεκμηριωμένη προσέγγιση στην επίλυση του προβλήματος, συνδυάζοντας θεωρητικές εγγυήσεις με πρακτική εφαρμογή.

3.2.2 Κατασκευή του Αλγορίθμου Longest First (LF)

Ο αλγόριθμος LF είναι ένας greedy (άπληστος) προσεγγιστικός αλγόριθμος, ο οποίος προγραμματίζει πρώτα τις μεγαλύτερες σε διάρκεια δοκιμές. Η βασική ιδέα είναι ότι αν προγραμματίσουμε τις μεγαλύτερες δοκιμές πρώτες, θα καταφέρουμε να μειώσουμε τα "κενά" που δημιουργούνται στον προγραμματισμό.

Ο αλγόριθμος εκτελείται σε δύο φάσεις:

Φάση 1: Ταξινόμηση των Ακμών

- Αρχικά, ταξινομούμε τις δοκιμές με βάση τον χρόνο εκτέλεσής τους l(e)l σε φθίνουσα σειρά.
- Δηλαδή, οι δοκιμές με μεγαλύτερο χρόνο εκτέλεσης προγραμματίζονται πρώτες.

Φάση 2: Προγραμματισμός των Δοκιμών

- Ξεκινώντας από τη δοκιμή με τη μεγαλύτερη διάρκεια, βρίσκουμε το πρώτο διαθέσιμο χρονικό διάστημα για εκτέλεση.
- Λαμβάνοντας υπόψη τις γειτονικές δοκιμές, αποφεύγουμε επικαλύψεις στις μονάδες που χρησιμοποιούνται.
- Συνεχίζουμε αυτή τη διαδικασία μέχρι να προγραμματιστούν όλες οι δοκιμές.

Αυτή η μέθοδος αποφεύγει μεγάλο αριθμό καθυστερήσεων και επιτρέπει την κατανομή των δοκιμών σε παράλληλες μονάδες, ελαχιστοποιώντας έτσι το makespan.

3.2.3 Παράδειγμα

Ας εξετάσουμε ένα παράδειγμα όπου έχουμε ένα diagnostic graph με 5 κόμβους και 9 διαγνωστικές δοκιμές. Η είσοδος περιλαμβάνει τις δοκιμές και τον χρόνο εκτέλεσης τους:

Είσοδος Δεδομένων:

0 1 10

0 1 10

1 2 10

2 3 10

3 4 10

4 5 10

5 0 10

2 0 10

4 1 10

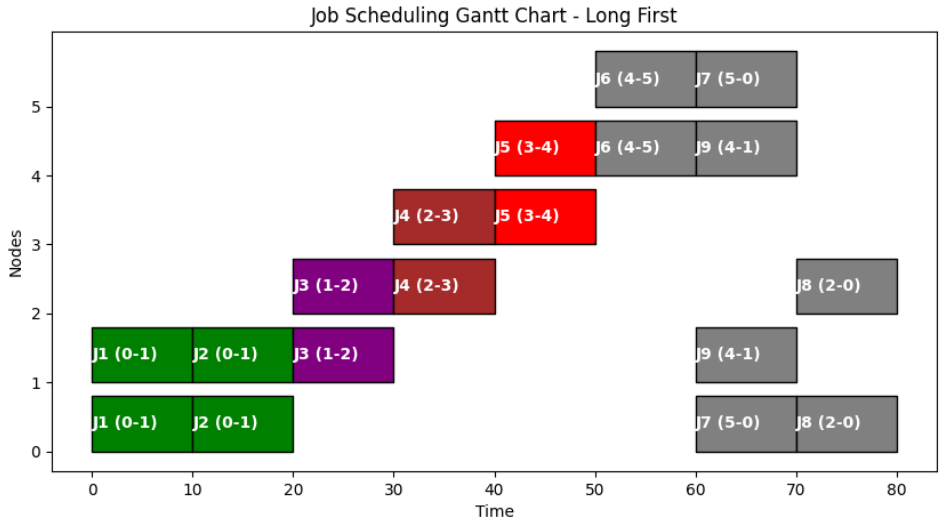
Οι αριθμοί αναπαριστούν: (node1,node2,weight)

όπου weight = χρόνος εκτέλεσης της διαγνωστικής δοκιμής.

Βήματα του Αλγορίθμου

1. **Ταξινόμηση ακμών κατά διάρκεια (Longest First Order):**

   (0, 1, 10), (0, 1, 10), (1, 2, 10), (2, 3, 10), (3, 4, 10), (4, 5, 10), (5, 0, 10), (2, 0, 10), (4, 1, 10)



Εικόνα 6: Πίνακας ταξινόμησης εργασιών

2. **Προγραμματισμός των δοκιμών:**

- Οι πρώτες δοκιμές εκτελούνται στις **μονάδες που δεν είναι απασχολημένες**.
- Όταν μία μονάδα είναι απασχολημένη, η δοκιμή προγραμματίζεται στο πρώτο διαθέσιμο χρονικό διάστημα.

3.3 Preemptive Lazy Binning (PLB) Algorithm

3.3.1 Περιγραφή

Σε αυτήν την ενότητα εξετάζουμε ένα διαφορετικό πρόβλημα χρονοπρογραμματισμού, το οποίο αφορά τη διαχείριση βαθμονομήσεων (calibrations) σε ένα περιβάλλον μονομηχανής. Θεωρούμε ότι κάθε βαθμονόμηση είναι στιγμιαία, αλλά διατηρεί τη μηχανή «έγκυρη» για ένα συγκεκριμένο διάστημα T μονάδων χρόνου. Μόλις παρέλθει αυτό το διάστημα, απαιτείται νέα βαθμονόμηση προκειμένου να εκτελεστούν περαιτέρω εργασίες. Μας δίνεται ένα σύνολο από n εργασίες (jobs), καθεμία με χρόνο έναρξης (release date) r_i, προθεσμία (deadline) d_i και χρόνο εκτέλεσης p_i. Μια εργασία μπορεί να εκτελείται μόνο όταν η μηχανή βρίσκεται σε κατάσταση βαθμονόμησης, ενώ επιτρέπεται η προ-διακοπή (preemption), δηλαδή η διακοπή και η επανεκκίνηση μιας εργασίας σε μεταγενέστερο διάστημα.

Ο στόχος μας είναι να ολοκληρώσουμε όλες τις εργασίες εντός των προθεσμιών τους, χρησιμοποιώντας τον ελάχιστο δυνατό αριθμό βαθμονομήσεων. Η αντικειμενική συνάρτηση, λοιπόν, δεν είναι η ελαχιστοποίηση του makespan ή κάποιας άλλης

κλασικής ποσότητας, αλλά η ελαχιστοποίηση των calibrations που απαιτούνται συνολικά για την εκτέλεση όλων των εργασιών.

Για την αντιμετώπιση αυτού του προβλήματος, υλοποιούμε τον αλγόριθμο Preemptive Lazy Binning (PLB). Η κεντρική ιδέα του PLB είναι να διαχωρίζει τον διαθέσιμο χρόνο σε διαστήματα εύρους T και να προγραμματίζει μέσα σε αυτά όσες εργασίες είναι εφικτό, καθορίζοντας νέο διάστημα βαθμονόμησης όταν δεν μπορεί πλέον να προγραμματιστεί καμία επιπλέον εργασία. Συγκεκριμένα, σε κάθε βήμα:

1. Υπολογίζουμε τον «μέγιστο χρόνο έναρξης» της επόμενης βαθμονόμησης, λαμβάνοντας υπόψη τους υπολειπόμενους χρόνους εκτέλεσης και τα deadlines των εργασιών.
2. Χρησιμοποιούμε τη στρατηγική Earliest Deadline First (EDF) για να προγραμματίζουμε τις εργασίες μέσα στο διάστημα [t, t+T).
3. Όταν δεν είναι εφικτό να χωρέσουν άλλες εργασίες σε αυτό το διάστημα χωρίς να παραβιαστούν οι προθεσμίες, ορίζουμε νέα βαθμονόμηση και επαναλαμβάνουμε τη διαδικασία.

Ο PLB έχει αποδειχθεί ότι παράγει βέλτιστη λύση σε πολυωνυμικό χρόνο, αποφεύγοντας την ψευδοπολυωνυμική πολυπλοκότητα που θα προέκυπτε αν διασπάζαμε τις εργασίες σε μονάδες χρόνου (unit-time). Η πολυπλοκότητά του οφείλεται κυρίως στην ταξινόμηση των εργασιών ($O(n \log n)$) και στη διαδοχική διερεύνηση των χρονικών διαστημάτων (έως $O(n)$ βήματα, καθένα από τα οποία υλοποιείται σε $O(n)$ ή $O(n \log n)$), με αποτέλεσμα συνολική τάξη μεγέθους $O(n^2 \log n)$ στη χειρότερη περίπτωση. Με αυτόν τον τρόπο, επιτυγχάνουμε έναν ακριβή και αποτελεσματικό αλγόριθμο για την ελαχιστοποίηση των βαθμονομήσεων σε περιβάλλον που επιτρέπει προ-διακοπή εργασιών.

3.3.2 Κατασκευή αλγορίθμου

Ταξινόμηση Εργασιών κατά Προθεσμία:

Ξεκινάμε με την ταξινόμηση των εργασιών J σε μη-φθίνουσα σειρά των deadlines (d_j). Έτσι, μπορούμε εύκολα να ελέγχουμε ποιες εργασίες είναι «επείγουσες».

1. **Υπολογισμός του «Τρέχοντος Μέγιστου Χρόνου Έναρξης» Βαθμονόμησης:**

   Σε κάθε βήμα:

- Για κάθε deadline d_k από το σύνολο των εναπομενουσών εργασιών, μετράμε το άθροισμα των χρόνων επεξεργασίας (p_j) των εργασιών που έχουν προθεσμία ≤ d_k.
- Το άθροισμα αυτό το αφαιρούμε από d_k. Παίρνουμε έτσι μια εκτίμηση για το πόσο «αργά» μπορούμε να ξεκινήσουμε τη βαθμονόμηση, ώστε οι εργασίες να μην παραβιάσουν το d_k.
- Επιλέγουμε τη μικρότερη από αυτές τις τιμές ως τον «τρέχοντα μέγιστο χρόνο έναρξης» της νέας βαθμονόμησης, που θα ορίσουμε στο t.

2. **Προγραμματισμός των Εργασιών (EDF):**
   - Ορίζουμε βαθμονόμηση στο χρόνο t. Στη συνέχεια, ξεκινώντας ακριβώς μετά το t, προγραμματίζουμε τις εναπομείνασες εργασίες σε Earliest Deadline First σειρά, έως ότου φτάσουμε σε κάποιο κρίσιμο deadline d_k ή ολοκληρωθεί το διάστημα [t, t+T).
   - Επιτρέπουμε την προ-διακοπή, άρα αν δεν προλαβαίνουμε να τελειώσουμε μια εργασία εντός [t, t+T), μπορούμε να τη μεταφέρουμε στην επόμενη βαθμονόμηση.
3. **Ενημέρωση Υπολειπόμενων Χρόνων:**
   - Μειώνουμε το p_j για όσες εργασίες εκτελέστηκαν (μερικώς ή ολικώς).
   - Αφαιρούμε πλήρως από το σύνολο όσες εργασίες ολοκληρώθηκαν.
4. **Επανάληψη:**
   - Εφόσον παραμένουν εργασίες μη ολοκληρωμένες (pj>0),

     επαναλαμβάνουμε τη διαδικασία: εντοπίζουμε νέο «μέγιστο χρόνο έναρξης» βαθμονόμησης και προγραμματίζουμε.

Ο αλγόριθμος τερματίζει όταν όλες οι εργασίες έχουν εκτελεστεί. Στο τέλος, ο αριθμός των βαθμονομήσεων που ορίστηκαν είναι ελαχιστοποιημένος**.**

3.3.4 Παράδειγμα

Σε αυτήν την ενότητα, παρουσιάζουμε ένα ενδεικτικό παράδειγμα υλοποίησης του αλγορίθμου Preemptive Lazy Binning (PLB).

Είσοδος

5 <-- T (μήκος βαθμονόμησης)

10 <-- N (αριθμός εργασιών)

1 0 54 6 <-- job_id=1, r=0, d=54, p=6

2 0 13 4

3 0 54 8

4 0 23 2

5 0 31 4

6 0 23 5

7 0 57 8

8 0 59 8

9 0 34 2

10 0 40 9

- Έχουμε, T=5 και έχουμε 10 εργασίες, καθεμιά με (r_j,d_j,p_j)
- Παρατηρούμε ότι οι προθεσμίες (d_j) και οι χρόνοι (p_j) είναι αυθαίρετοι, όχι μονάδες χρόνου.

Παρατηρούμε ότι οι προθεσμίες και οι χρόνοι εκτέλεσης είναι αυθαίρετοι, όχι απλά μονάδες χρόνου.

Κατά την εκτέλεση, διαβάζουμε τις εργασίες και τις ταξινομούμε με βάση τα deadlines. Στη συνέχεια, υπολογίζουμε τον μέγιστο δυνατό χρόνο έναρξης βαθμονόμησης t, λαμβάνοντας υπόψη το άθροισμα των υπολειπόμενων χρόνων των εργασιών που έχουν deadline μικρότερο ή ίσο με κάθε συγκεκριμένο d. Αμέσως μετά, εφαρμόζουμε την πολιτική Earliest Deadline First (EDF) στο διάστημα [t, t+T), προγραμματίζοντας τις εργασίες με σειρά προθεσμιών, και επιτρέποντας την προ-διακοπή όταν κάποια εργασία δεν ολοκληρώνεται εντός του διαστήματος. Ενημερώνουμε τα υπόλοιπα p κάθε εργασίας ανάλογα με το χρόνο που εκτελέστηκε και επαναλαμβάνουμε τη διαδικασία μέχρι να ολοκληρωθούν όλες οι εργασίες.

Το τελικό αποτέλεσμα που προκύπτει είναι ότι ο αλγόριθμος εμφανίζει, ως έξοδο, τα χρονικά σημεία έναρξης των βαθμονομήσεων καθώς και τα χρονικά διαστήματα εκτέλεσης κάθε εργασίας. Στο συγκεκριμένο παράδειγμα, παρατηρούμε ότι όλες οι εργασίες ολοκληρώθηκαν επιτυχώς εντός των deadlines, με τον αλγόριθμο να απαιτεί μόνο μία βαθμονόμηση (π.χ. με ένα calibration που ξεκίνησε στο χρόνο 59). Με τη χρήση των διαστημάτων βαθμονόμησης και της στρατηγικής EDF, καταφέρνουμε να εξασφαλίζουμε έναν ολοκληρωμένο και έγκαιρο προγραμματισμό των εργασιών, αποδεικνύοντας την αποδοτικότητα του PLB σύμφωνα με τις θεωρητικές εγγυήσεις.

Κεφάλαιο 4. Πειραματική Μελέτη

Σε αυτό το κεφάλαιο παρουσιάζουμε την πειραματική μελέτη που πραγματοποιήσαμε, προκειμένου να αξιολογήσουμε τους τρεις αλγορίθμους χρονοπρογραμματισμού που υλοποιήσαμε. Στόχος μας είναι:

1. Να **μετρήσουμε την απόδοση** κάθε αλγορίθμου με βάση τον παράγοντα προσέγγισης, ελέγχοντας σε ποιο βαθμό προσεγγίζει τη βέλτιστη λύση.
2. Να **εκτιμήσουμε τον χρόνο εκτέλεσης** για διαφορετικά μεγέθη εισόδου (π.χ. αριθμός εργασιών, πλήθος μηχανών), συγκρίνοντας τα αποτελέσματα με τη θεωρητική πολυπλοκότητα του κάθε αλγορίθμου.
3. Να αναλύσουμε πώς επηρεάζεται η απόδοση από βασικές παραμέτρους (π.χ. N,R,T,M), ώστε να εντοπίσουμε τυχόν ακραίες περιπτώσεις ή μοτίβα που επιβαρύνουν τη λύση.
4. Να ελέγξουμε το σφάλμα των αλγορίθμων σε σχέση με το βέλτιστο φορτίο, εξετάζοντας αν οι θεωρητικές εγγυήσεις επαληθεύονται στην πράξη.

Μελετάμε κάθε αλγόριθμο ώστε να διαπιστώσουμε εάν καθένας από αυτούς ανταποκρίνεται στις προσδοκίες που θέτει η βιβλιογραφία (π.χ. παράγοντας προσέγγισης ή βέλτιστη λύση σε περιβάλλον προ-διακοπής).

Για να είμαστε δίκαιοι και αντικειμενικοί στην αξιολόγησή μας:

- **Δημιουργούμε πολλαπλά σενάρια δοκιμών**, όπου μεταβάλλουμε συστηματικά τις παραμέτρους ανάλογα με τον αλγόριθμο (π.χ. το πλήθος των εργασιών , το πλήθος των μηχανών).
- **Εκτελούμε πολλαπλές φορές** τον αλγόριθμο σε κάθε σύνολο δεδομένων και **υπολογίζουμε τον μέσο όρο** (και, όπου είναι χρήσιμο, την τυπική απόκλιση), ώστε να μειώνουμε τον θόρυβο από τυχαιοποιημένες επιλογές.
- **Παρουσιάζουμε τα αποτελέσματα** σε πίνακες και γραφήματα, σχολιάζοντας τη συμπεριφορά του αλγορίθμου και τη σχέση της με τη θεωρητική πολυπλοκότητα ή τον αντίστοιχο παράγοντα προσέγγισης.
- **Συνοψίζουμε** τυχόν ακραίες περιπτώσεις ή μοτίβα που παρατηρήσαμε, καθώς και ενδεχόμενες αποκλίσεις από τα αναμενόμενα θεωρητικά όρια.

Στις επόμενες ενότητες, εξετάζουμε διαδοχικά κάθε αλγόριθμο. Αρχικά, υπενθυμίζουμε εν συντομία τη λογική και τις θεωρητικές εγγυήσεις του, και στη συνέχεια παρουσιάζουμε αναλυτικά τους πειραματικούς δείκτες (χρόνος εκτέλεσης, απόκλιση

από το βέλτιστο, κ.ο.κ.). Τέλος, προχωράμε σε μια συνολική σύνοψη, όπου συζητάμε τα σημαντικότερα ευρήματα και πιθανές βελτιώσεις.

4.1 Αλγόριθμος για Scheduling σε Τρεις Αφιερωμένες Μηχανές

Σε αυτή την ενότητα παρουσιάζουμε την πειραματική αξιολόγηση του πρώτου αλγορίθμου μας, ο οποίος επιλύει το πρόβλημα του scheduling σε τρεις αφιερωμένες μηχανές. Στόχος μας είναι να ελαχιστοποιήσουμε το makespan, εξασφαλίζοντας ότι καθεμία από τις τρεις μηχανές {m1,m2,m3} εκτελεί κάθε φορά μία μόνο εργασία. Στη βιβλιογραφία, γνωρίζουμε πως ο αλγόριθμος παρέχει λόγο προσέγγισης 7\6 σε σχέση με τη βέλτιστη λύση.

4.1.1 Παράμετροι Πειραματισμού και Διαδικασία Προσομοίωσης

Για τα πειράματα στον αλγόριθμο “Scheduling σε Τρεις Αφιερωμένες Μηχανές”, ορίζουμε και καταγράφουμε τα εξής:

- **N**: Αριθμός εργασιών. Δοκιμάζουμε διάφορες τιμές (π.χ. 100, 1.000, 10.000, …, 900.000), προκειμένου να μελετήσουμε την κλιμάκωσή του σε μικρά και μεγάλα προβλήματα.
- **R** (Ratio): Είναι ο συνολικός χρόνος που απαιτείτε για τις εργασίες που θέλουν δυο μηχανές προς τον βέλτιστο χρόνο που απαιτείτε για όλες τις εργασίες. Αυτό σχετίζεται με το ποσοστό των εργασιών που εκτελούνται σε δύο μηχανές. Εξετάζουμε R από 0.0 έως 1.0 (ανά βήμα 0.1).
- **p** (Processing Time Factor): Ένας σταθερός συντελεστής που καθορίζει τη μέση διάρκεια των εργασιών.
- **OPT**: Ακριβής τιμή του «βέλτιστου» συνολικού φόρτου (makespan) με βάση την κατανομή εργασιών. Χρησιμοποιείται για να υπολογίσουμε πόσο απέχει ο αλγόριθμος από το ιδανικό (σφάλμα).

Καταγραφή Μεγεθών:

1. **Χρόνος Εκτέλεσης** του αλγορίθμου (Time).
2. **Τελικό Makespan** που προκύπτει από το παραγόμενο schedule.
3. **Σφάλμα (Error)**: Ορίζεται ως (Makespan \ OPT) - 1, ώστε να βλέπουμε την αναλογία απόκλισης.

4. **OPT**, για κάθε συνδυασμό (N, R), ώστε να ελέγχουμε πόσο κοντά βρίσκεται το makespan του αλγορίθμου στο εκτιμώμενο βέλτιστο.

4.1.2 Αποτελέσματα και Διαγράμματα

Σε αυτή την ενότητα, εστιάζουμε στα αποτελέσματα που προέκυψαν από τα πειράματα για τον αλγόριθμο “Scheduling σε Τρεις Αφιερωμένες Μηχανές” και παρουσιάζουμε τα διαγράμματα που αποτυπώνουν την απόδοση του. Θέλουμε να αναδείξουμε:

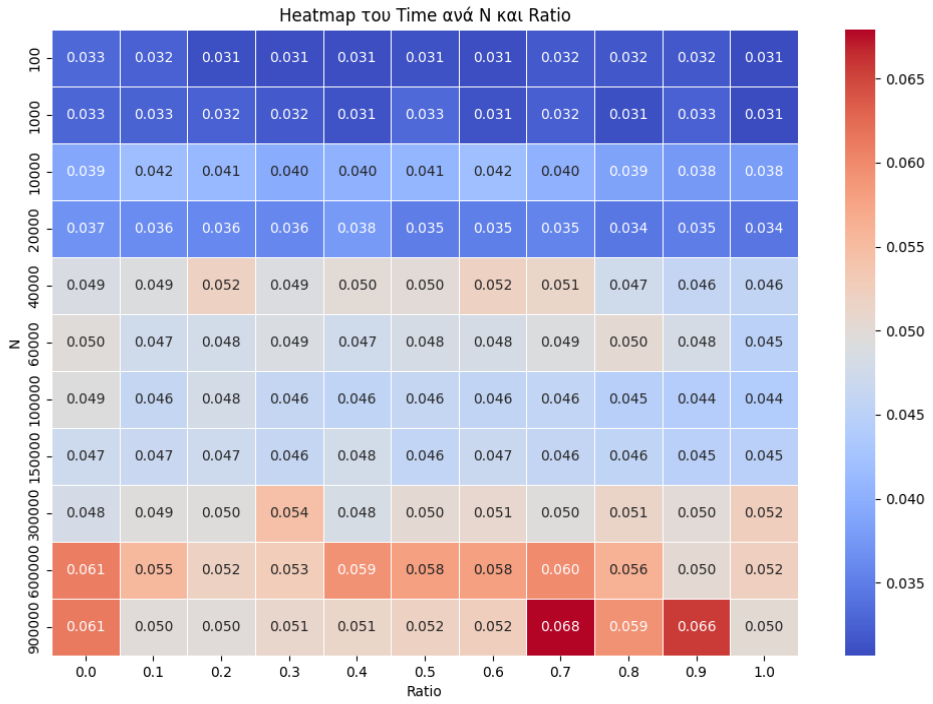
1. **Πώς εξελίσσεται ο χρόνος εκτέλεσης (Time)** του αλγορίθμου, καθώς μεταβάλλονται ο αριθμός των εργασιών (N) και η αναλογία (R) των εργασιών που μοιράζονται σε δύο μηχανές.
2. **Πώς διαμορφώνεται το σφάλμα (Error)**, δηλαδή η απόκλιση του τελικού makespan σε σχέση με το εκτιμώμενο βέλτιστο (OPT).

Για να αποκτήσουμε μια σφαιρική εικόνα, παρουσιάζουμε διαφορετικούς τύπους γραφημάτων:

- **Heatmaps** (Time/Error vs. N και R): Μας επιτρέπουν να δούμε συγκεντρωτικά πώς μεταβάλλεται το Time ή το Error σε όλο το φάσμα των τιμών (N, R).
- **Γραφήματα Time vs. N** (για κάθε R) και αντίστοιχα **Error vs. N** (για κάθε R): Εξετάζουμε αναλυτικότερα τις καμπύλες, ώστε να δούμε αν ο χρόνος κλιμακώνεται γρήγορα ή αργά και πώς το σφάλμα μεγαλώνει ή μειώνεται ανάλογα με το μέγεθος του προβλήματος.
- **Μέσοι Όροι Time ή Error για κάθε N** (ανεξαρτήτως R): Απομονώνουμε την επίδραση του N, ενοποιώντας τις τιμές του R και αναζητούμε γενικές τάσεις κλιμάκωσης ή ακρίβειας.
- **Κατανομές (distributions)** Time και Error: Χρησιμοποιούμε histograms για να δούμε τη στατιστική διασπορά των μετρήσεων, ερμηνεύοντας κατά πόσο ο αλγόριθμος εμφανίζει σταθερή συμπεριφορά ή αν αντιμετωπίζουμε ακραίες τιμές σε ορισμένες περιπτώσεις.

Θα αναλύσουμε βήμα προς βήμα κάθε σύνολο διαγραμμάτων. Ξεκινάμε από την εξέταση του χρόνου εκτέλεσης (Time) και στη συνέχεια προχωρούμε στον παράγοντα σφάλματος (Error). Στο τέλος, σχολιάζουμε τυχόν ακραία μοτίβα, επιβεβαιώνουμε ή διαψεύδουμε τις θεωρητικές προσδοκίες και συνοψίζουμε τα βασικά μας συμπεράσματα.

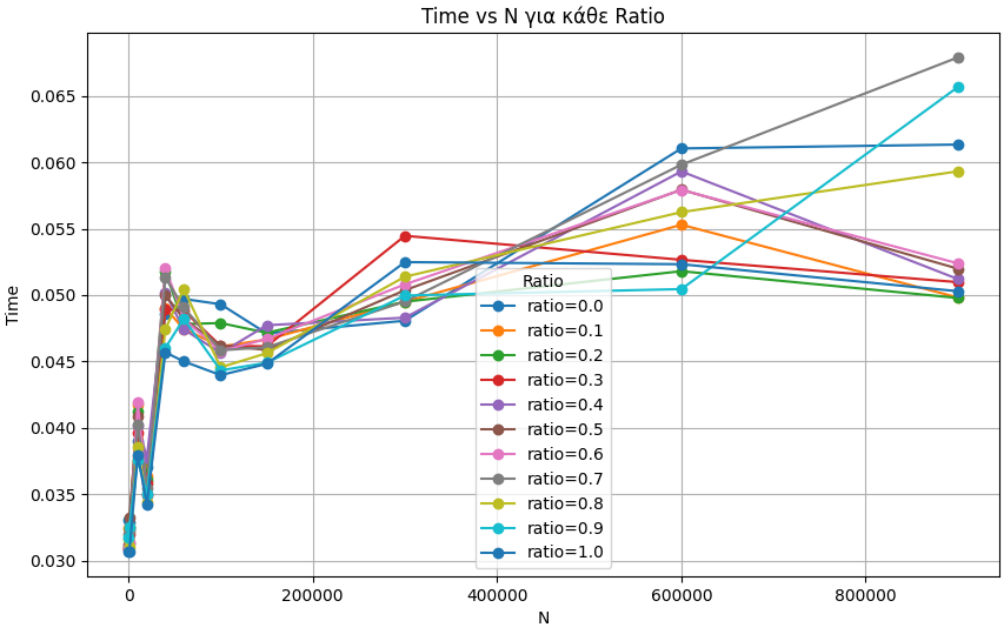
Ανάλυση της Εξέλιξης του Χρόνου Εκτέλεσης του Αλγορίθμου



Εικονα 7 : Heatmap του Χρόνου Εκτέλεσης (Time) συναρτήσει του αριθμού εργασιών N και της αναλογίας εργασιών Rπου εκτελούνται σε δύο μηχανές.

Στα κελιά εμφανίζεται ο μέσος χρόνος εκτέλεσης, με το χρώμα να κυμαίνεται από μπλε (χαμηλότερη τιμή) έως κόκκινο (υψηλότερη τιμή).

Παρατηρούμε ότι, για μικρό N, ο χρόνος εκτέλεσης παραμένει χαμηλός ανεξάρτητα από το R. Αντίθετα, σε μεγάλες τιμές N, βλέπουμε υψηλότερες τιμές χρόνου, ιδίως όταν το R βρίσκεται γύρω στο 0.6–0.7. Αυτό υποδηλώνει ότι το «μικτό» σενάριο, όπου αρκετές εργασίες απαιτούν ταυτόχρονη επεξεργασία σε δύο μηχανές, επιβαρύνει περισσότερο τον αλγόριθμο από ό,τι το σενάριο με αμιγώς μονές ή αμιγώς διπλές εργασίες. Ωστόσο, ακόμα και στις πιο κόκκινες περιοχές, ο χρόνος παραμένει σε ανεκτά επίπεδα, επιβεβαιώνοντας την πολυωνυμική συμπεριφορά του αλγορίθμου. Ένα ενδιαφέρον εύρημα είναι ότι στο ακραίο άκρο R=1.0, ο χρόνος εκτέλεσης δεν κορυφώνεται κατ' ανάγκην, πράγμα που μας οδηγεί στο συμπέρασμα ότι μια διπλομηχανιακή κατανομή μπορεί να διαχειρίζεται πιο ομοιόμορφα από την αλγόριθμο, συγκριτικά με την περίπτωση ενδιάμεσων τιμών R.

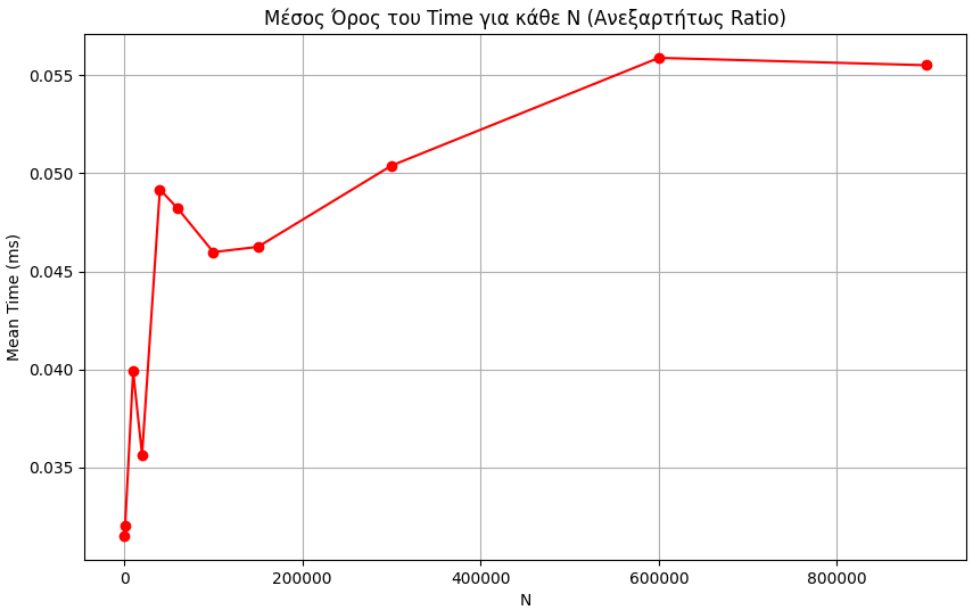


Εικόνα 8: Στο συγκεκριμένο διάγραμμα, απεικονίζουμε τον χρόνο εκτέλεσης του αλγορίθμου συναρτήσει του αριθμού των εργασιών (N) για διαφορετικές τιμές του R.

Παρατηρούμε ότι:

- Ο χρόνος εκτέλεσης αυξάνεται με το N, επιβεβαιώνοντας την πολυπλοκότητα του αλγορίθμου.
- Οι καμπύλες δεν είναι απολύτως μονοτονικές, παρουσιάζοντας μικρές διακυμάνσεις λόγω τυχαίων κατανομών εργασιών.
- Για πολύ μικρές τιμές N, ο χρόνος παραμένει χαμηλός και σχετικά σταθερός.
- Σε μεγάλα N, ορισμένα ratio, ιδιαίτερα όταν R πλησιάζει 1.0, προκαλούν μεγαλύτερη αύξηση του χρόνου εκτέλεσης, γεγονός που υποδηλώνει ότι ο διαμοιρασμός των εργασιών μεταξύ μηχανών επηρεάζει την απόδοση.

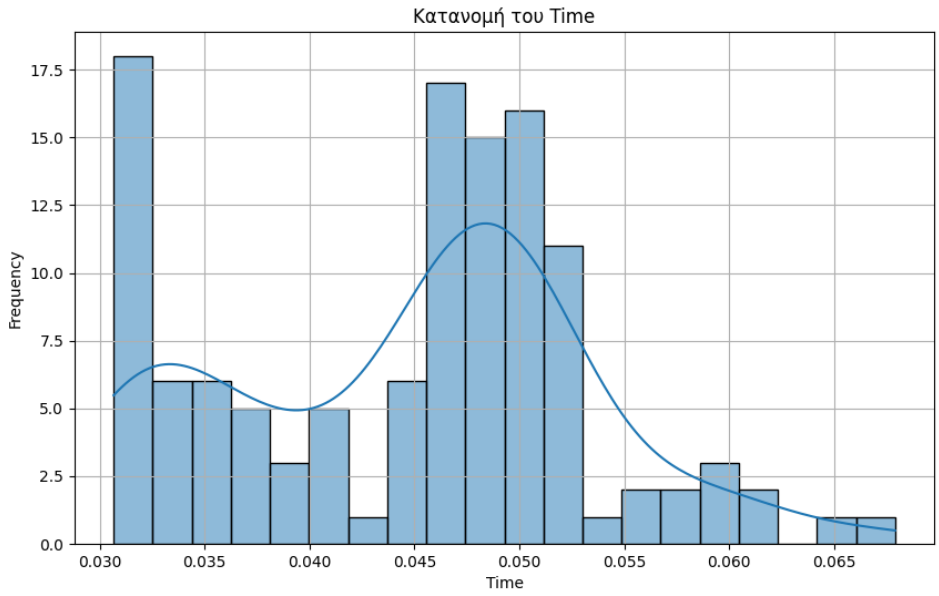
Γενικά, το διάγραμμα επιβεβαιώνει ότι η εκτέλεση του αλγορίθμου παρουσιάζει ήπια κλιμάκωση με το N, ενώ η παράμετρος R παίζει σημαντικό ρόλο στη συνολική απόδοση, ειδικά όταν περισσότερες εργασίες κατανέμονται σε δύο μηχανές.



Εικόνα 9: Το γράφημα παρουσιάζει τον μέσο χρόνο εκτέλεσης του αλγορίθμου σε σχέση με τον αριθμό των εργασιών N, ανεξαρτήτως του λόγου κατανομής R.

Παρατηρούμε ότι, για μικρές τιμές του N, ο χρόνος παρουσιάζει διακυμάνσεις, αλλά καθώς αυξάνεται το πλήθος των εργασιών, η συμπεριφορά γίνεται πιο σταθερή και εμφανίζει μία σχεδόν γραμμική αύξηση.

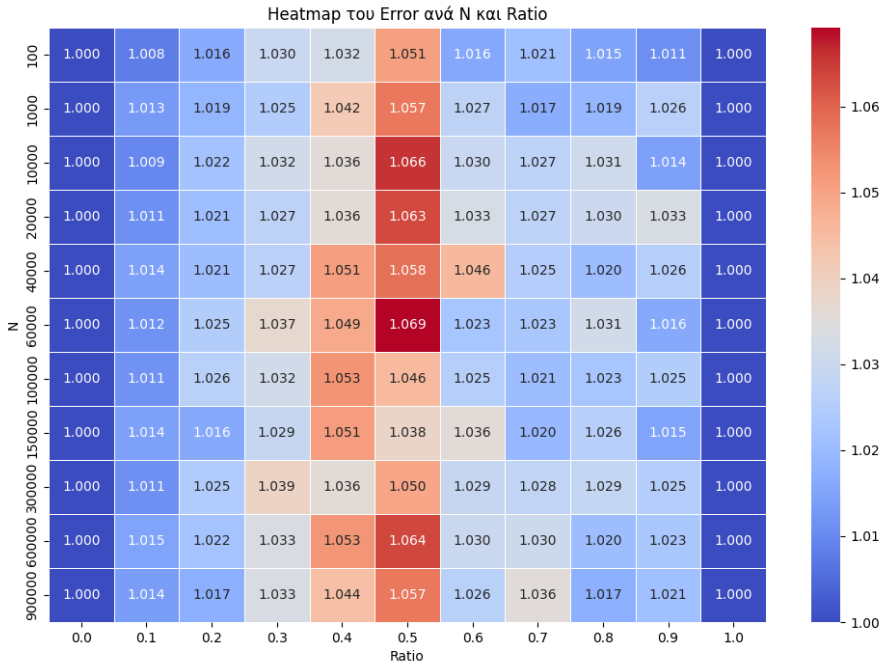
Αυτό επιβεβαιώνει ότι η πολυπλοκότητα του αλγορίθμου είναι κοντά στο O(N), καθώς ο χρόνος εκτέλεσης αυξάνεται αναλογικά με τον αριθμό των εργασιών. Η ήπια αύξηση του χρόνου στις υψηλότερες τιμές του N υποδηλώνει ότι ο αλγόριθμος παραμένει αποδοτικός ακόμα και για μεγάλα δεδομένα, χωρίς να παρουσιάζει εκθετική αύξηση.



Εικόνα 10: Το γράφημα **"Κατανομή του Time"** δείχνει τη συχνότητα εμφάνισης διαφορετικών χρόνων εκτέλεσης του αλγορίθμου.

Παρατηρούμε δύο κύριες συγκεντρώσεις τιμών: μία γύρω από 0.030 - 0.035 και μία δεύτερη μεταξύ 0.045 - 0.055, όπου εντοπίζονται οι περισσότερες εκτελέσεις. Η κατανομή εμφανίζει ασυμμετρία, με ορισμένες υψηλότερες τιμές (άνω των 0.060), γεγονός που υποδηλώνει ότι σε ορισμένες περιπτώσεις ο χρόνος εκτέλεσης αυξάνεται. Αυτό πιθανότατα οφείλεται σε αυξημένο αριθμό εργασιών ή μεγαλύτερο ποσοστό διαμοιραζόμενων εργασιών μεταξύ μηχανών. Συνολικά, ο αλγόριθμος εμφανίζει σταθερότητα στον χρόνο εκτέλεσης, με ορισμένες αποκλίσεις σε πιο απαιτητικές περιπτώσεις.

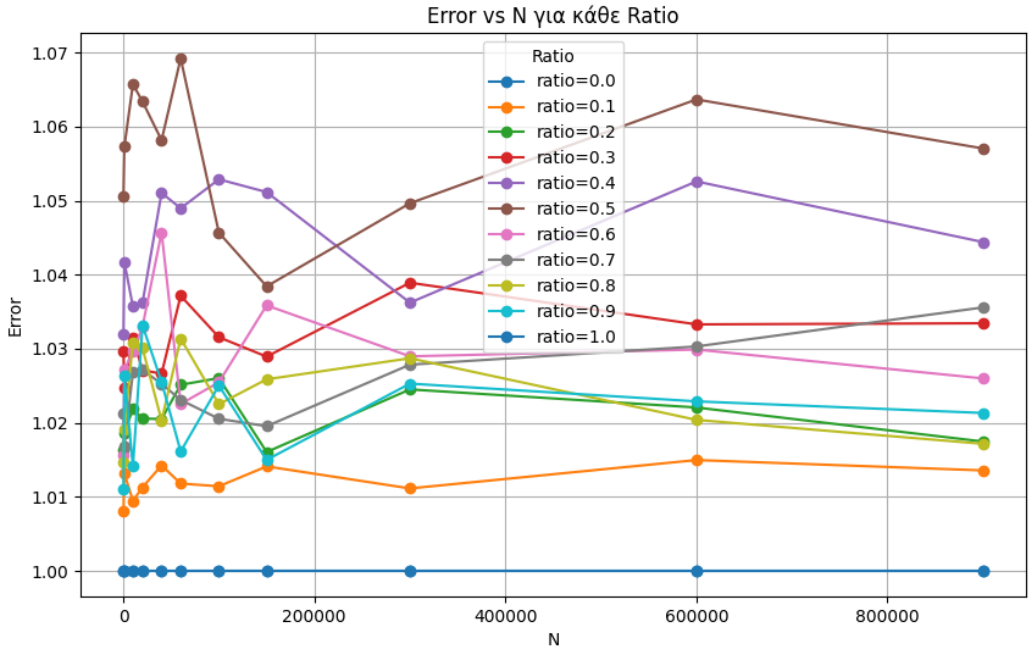
Ανάλυση του Σφάλματος και της Απόκλισης από το Βέλτιστο



Εικόνα 11: Το heatmap του Error ανά N και Ratio αποτυπώνει την απόκλιση του τελικού makespan από το βέλτιστο (OPT) για διαφορετικές τιμές του αριθμού εργασιών (N) και της αναλογίας διαμοιραζόμενων εργασιών (R).

Παρατηρούμε ότι για ακραίες τιμές του R (0.0 ή 1.0), το error είναι 1.0, δηλαδή η λύση που παράγει ο αλγόριθμος είναι βέλτιστη. Οι μεγαλύτερες αποκλίσεις εμφανίζονται για ενδιάμεσες τιμές του R (0.4 - 0.6), όπου το error μπορεί να φτάσει έως 1.069, αλλά παραμένει πάντα κοντά στη θεωρητική εγγύηση του 7/6.

Επιπλέον, η αύξηση του N δεν επιφέρει μεγάλη μεταβολή στο error, γεγονός που υποδηλώνει ότι ο παράγοντας προσέγγισης είναι σταθερός ανεξάρτητα από το μέγεθος του προβλήματος. Αυτό ενισχύει τη θεωρητική ανάλυση του αλγορίθμου, η οποία εγγυάται ότι το makespan δεν ξεπερνά το 7/6 της βέλτιστης λύσης, ακόμα και για μεγάλα δεδομένα.

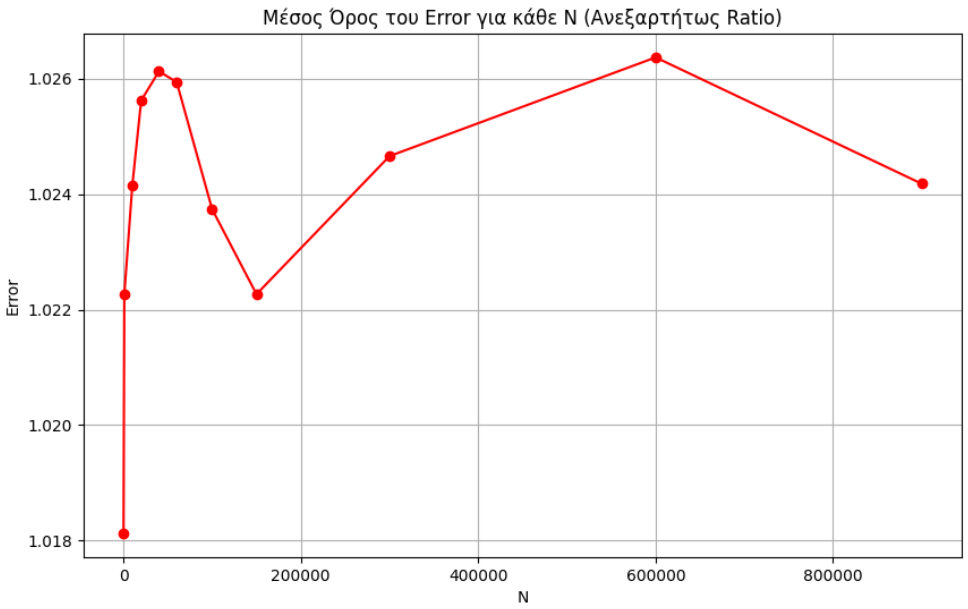


Εικόνα 12: Το Error vs N για κάθε Ratio αποτυπώνει την απόκλιση του makespan από το βέλτιστο καθώς μεταβάλλεται το πλήθος των εργασιών (N) και η αναλογία των εργασιών που εκτελούνται σε δύο μηχανές (R).

Παρατηρούμε ότι για ακραίες τιμές του R (0.0 και 1.0), το error παραμένει σταθερό στο 1.0, επιβεβαιώνοντας ότι ο αλγόριθμος αποδίδει τη βέλτιστη λύση. Για ενδιάμεσες τιμές του R, εμφανίζεται μεγαλύτερη απόκλιση, ιδιαίτερα κοντά στο R = 0.4 - 0.6, όπου το error φτάνει μέγιστες τιμές περίπου 1.07.

Παράλληλα, παρατηρούμε ότι καθώς το N αυξάνεται, η απόκλιση τείνει να σταθεροποιείται γύρω από 1.02-1.03 για τις περισσότερες τιμές του R, δείχνοντας ότι η ακρίβεια του αλγορίθμου βελτιώνεται όσο το πρόβλημα γίνεται μεγαλύτερο. Επιπλέον, η διακύμανση του error μειώνεται, γεγονός που υποδηλώνει ότι η συμπεριφορά του αλγορίθμου γίνεται πιο προβλέψιμη σε μεγάλα δεδομένα.

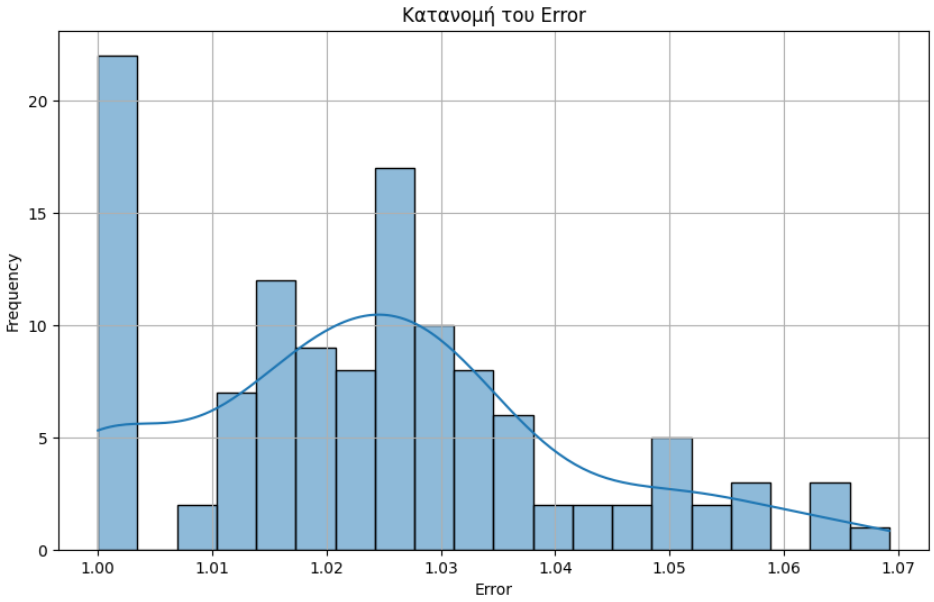
Αυτά τα αποτελέσματα επιβεβαιώνουν τον θεωρητικό παράγοντα προσέγγισης του αλγορίθμου, ο οποίος παραμένει εντός του εύρους που εγγυάται η βιβλιογραφία, αποδεικνύοντας ότι η μέθοδος είναι σταθερή και αποδοτική ακόμα και για μεγάλης κλίμακας προβλήματα.



Εικόνα 13: Μέσος Όρος του Error για κάθε N (Ανεξαρτήτως Ratio) αποτυπώνει τη γενική τάση του σφάλματος καθώς αυξάνεται το πλήθος των εργασιών (N).

Παρατηρούμε ότι το error ξεκινά από μια τιμή κοντά στο 1.018 για πολύ μικρές τιμές του N και αυξάνεται σταδιακά μέχρι περίπου 1.026. Στη συνέχεια, εμφανίζει μια μικρή μείωση πριν σταθεροποιηθεί κοντά στο 1.024 για τις μεγαλύτερες τιμές του N.

Αυτό το μοτίβο υποδηλώνει ότι, αν και το error αυξάνεται ελαφρώς σε μεσαία μεγέθη προβλημάτων, τείνει να σταθεροποιείται καθώς το N γίνεται πολύ μεγάλο, επιβεβαιώνοντας τη σταθερότητα του αλγορίθμου. Η μέση τιμή του error (~1.024-1.026) επιβεβαιώνει επίσης τον θεωρητικό παράγοντα προσέγγισης, που αναμένεται να παραμένει σε χαμηλά επίπεδα. Συνεπώς, ο αλγόριθμος διατηρεί μια αξιόπιστη απόδοση ακόμα και σε μεγάλα μεγέθη δεδομένων.



Εικόνα 14: Η κατανομή του Error δείχνει τη συχνότητα εμφάνισης διαφορετικών τιμών σφάλματος στον αλγόριθμο.

Παρατηρούμε ότι οι περισσότερες τιμές συγκεντρώνονται γύρω στο 1.02 - 1.03, ενώ υπάρχουν και περιπτώσεις όπου το error φτάνει κοντά στο 1.06 - 1.07. Παράλληλα, υπάρχει ένα έντονο peak στο 1.00, που υποδηλώνει ότι σε αρκετές περιπτώσεις ο αλγόριθμος πέτυχε τη βέλτιστη λύση.

Η γενική μορφή της κατανομής δείχνει ότι το error είναι χαμηλό στη συντριπτική πλειονότητα των περιπτώσεων, γεγονός που επιβεβαιώνει τη σταθερότητα του αλγορίθμου και τον παράγοντα προσέγγισης. Οι εξαιρετικά υψηλές τιμές του error εμφανίζονται σπάνια, γεγονός που υποδηλώνει ότι σε συγκεκριμένες περιπτώσεις το πρόγραμμα επηρεάζεται από το μέγεθος ή τη δομή των δεδομένων. Ωστόσο, το error παραμένει χαμηλό και σε σύγκριση με τη θεωρητική προσέγγιση, ο αλγόριθμος διατηρεί μια αξιόπιστη απόδοση.

Συνολικό Συμπέρασμα για τον Αλγόριθμο "Scheduling σε Τρεις Αφιερωμένες Μηχανές"

Ο αλγόριθμος επιδεικνύει σχεδόν γραμμική αύξηση στον χρόνο εκτέλεσης καθώς μεγαλώνει το πλήθος των εργασιών (N), με μικρές αποκλίσεις ανάλογα με το Ratio (R). Το σφάλμα (Error) παραμένει χαμηλό, κυμαινόμενο κυρίως μεταξύ 1.02 και 1.05, επιβεβαιώνοντας ότι η απόδοση παραμένει κοντά στο βέλτιστο σε όλο το εύρος των πειραμάτων.

Η σταθερότητα του αλγορίθμου σε μεγάλες κλίμακες δεδομένων δείχνει ότι είναι αξιόπιστος και αποδοτικός, διατηρώντας την προσέγγισή του εντός του θεωρητικού ορίου 7/6. Τα αποτελέσματα υποδεικνύουν ότι η μέθοδος είναι κατάλληλη για πρακτικές εφαρμογές, προσφέροντας έναν καλό συμβιβασμό μεταξύ υπολογιστικού κόστους και ποιότητας λύσης.

4.2 Αλγόριθμος Longest First (LF)

Σε αυτήν την ενότητα εστιάζουμε στον δεύτερο αλγόριθμο της μελέτης μας, τον **Longest First (LF)**, και παρουσιάζουμε την πειραματική αξιολόγηση που πραγματοποιήσαμε. Θυμίζουμε ότι ο LF στοχεύει να προγραμματίσει πρώτα τις μεγαλύτερες σε διάρκεια διαγνωστικές δοκιμές, ελαχιστοποιώντας έτσι τα «κενά» στον προγραμματισμό. Εδώ εξετάζουμε πώς εξελίσσεται το σφάλμα (Error) και ο χρόνος εκτέλεσης, κυρίως σε σχέση με τον αριθμό των μηχανών (M) και για διαφορετικά μεγέθη εργασιών (N).

4.2.1 Παράμετροι Πειραματισμού και Διαδικασία Προσομοίωσης

Για τα πειράματα του Longest First, ορίσαμε:

- **M** (Number of Machines):
    - Αποτελεί πάντα ζυγό αριθμό, με τιμές [10,20,40,60,80,120,150]
    - Όσο αυξάνεται το M, τόσο πιο πολύπλοκη γίνεται η «γραφική» δομή των δοκιμών (edge coloring).
- **N** (Number of Jobs):
    - Αντιπροσωπεύει το συνολικό πλήθος των διαγνωστικών δοκιμών.
    - Χρησιμοποιούμε τιμές [15.000,30.000,50.000,60.000,80.000,100.000]
    - Αν N είναι πολύ μικρό σε σχέση με το M, η γεννήτρια δεδομένων ενδέχεται να αποτύχει.
- **p** (Processing Time Factor):
    - Ρυθμίζει τη διάρκεια κάθε εργασίας μέσω της συνάρτησης $D=\lfloor (x+3)\backslash 4 \rfloor \times p$, , όπου p=20.
    - Μπορούμε να μεταβάλουμε το p σε ενδεχόμενα σενάρια για να ελέγξουμε την ευαισθησία του αλγορίθμου στον χρόνο εκτέλεσης.

Σε αυτή τη μελέτη, θέλουμε κυρίως να δούμε πώς μεταβάλλεται το σφάλμα (Error) για διάφορες τιμές του (M) και του (N). Επιπλέον, καταγράφουμε τον χρόνο εκτέλεσης του αλγορίθμου, για να διαπιστώσουμε αν η επιπλέον πολυπλοκότητα με πολλά μηχανήματα οδηγεί σε σημαντική καθυστέρηση ή αν ο LF παραμένει σχετικά αποδοτικός.

4.2.2 Αποτελέσματα και Διαγράμματα

Στην ενότητα αυτή, εστιάζουμε στα αποτελέσματα που προέκυψαν από τα πειράματα για τον αλγόριθμο **Longest First (LF)** και παρουσιάζουμε τα διαγράμματα που αποτυπώνουν την απόδοσή του. Στόχος μας είναι να δείξουμε:

1. **Πώς εξελίσσεται ο χρόνος εκτέλεσης (Time)** του αλγορίθμου, καθώς μεταβάλλονται ο αριθμός των μηχανών (M)(και ο αριθμός των εργασιών (N)).
2. **Πώς διαμορφώνεται το σφάλμα (Error)**, δηλαδή η απόκλιση του τελικού makespan σε σχέση με ένα βέλτιστο.

Για να αποκτήσουμε μια σφαιρική εικόνα, οργανώνουμε τις αναπαραστάσεις μας σε δύο κύριες κατηγορίες:

- **Ανάλυση Χρόνου Εκτέλεσης**

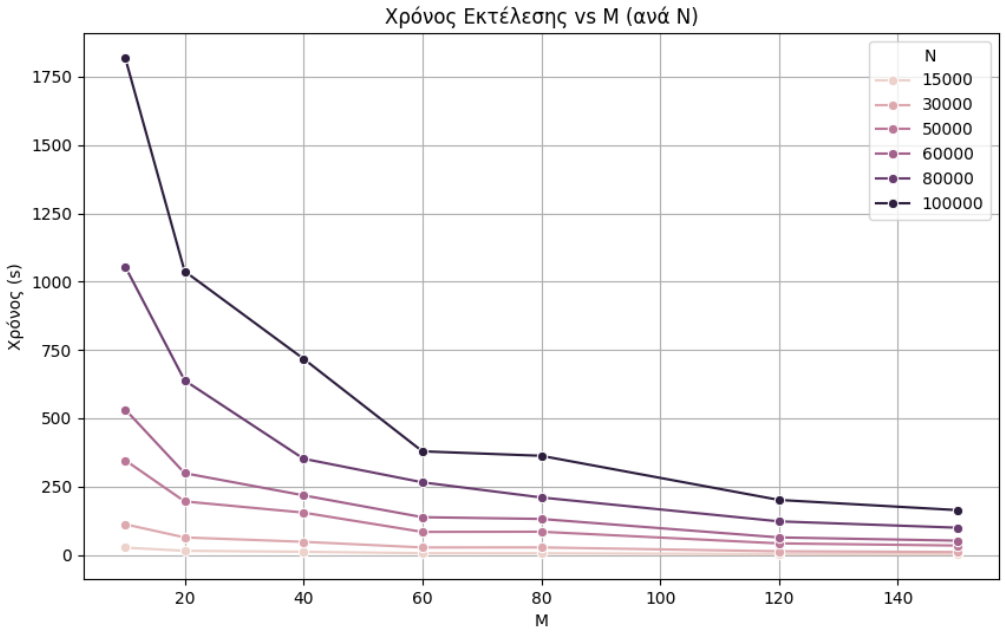
  Εξετάζουμε γραφήματα **Time vs. M, Time vs. N**, καθώς και ένα διάγραμμα με τον **μέσο όρο του χρόνου σε σχέση με το N**. Με αυτόν τον τρόπο, αξιολογούμε πώς κλιμακώνεται ο αλγόριθμος σε διαφορετικά μεγέθη προβλήματος.

- **Ανάλυση Σφάλματος / Απόκλισης**

  Παρουσιάζουμε **Error vs. M** και **Error vs. N** για να δείξουμε σε ποιο βαθμό η λύση του LF απέχει από το βέλτιστο makespan. Επίσης, εξετάζουμε πώς μεταβάλλεται το σφάλμα σε διάφορες συνθήκες φόρτου εργασίας.

Θα αναλύσουμε βήμα προς βήμα κάθε σύνολο γραφημάτων. Ξεκινάμε από την εξέλιξη του χρόνου εκτέλεσης και στη συνέχεια προχωρούμε στην εξέταση του σφάλματος. Στο τέλος, σχολιάζουμε τυχόν ακραίες περιπτώσεις που διαπιστώνουμε και συνοψίζουμε τα κύρια συμπεράσματά μας, ελέγχοντας αν οι πειραματικές παρατηρήσεις συνάδουν με τη θεωρητική ανάλυση του αλγορίθμου.

Ανάλυση Χρόνου Εκτέλεσης



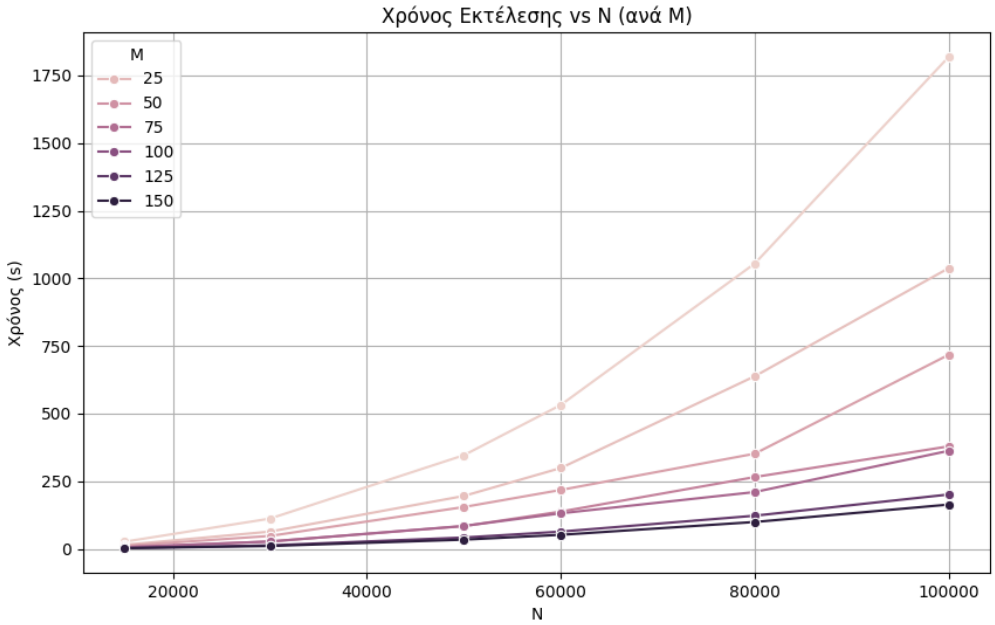
Εικόνα 15: Χρόνος Εκτέλεσης vs. M (ανά N)

Παρατηρούμε ότι**:**

- Ο χρόνος εκτέλεσης μειώνεται καθώς αυξάνεται ο αριθμός των μηχανών, ιδιαίτερα για χαμηλές τιμές M.
- Για μεγάλα M, η μείωση του χρόνου εκτέλεσης επιβραδύνεται, επιβεβαιώνοντας ότι από κάποιο σημείο και μετά η προσθήκη περισσότερων μηχανών δεν επιφέρει σημαντικά κέρδη.
- Όσο περισσότερες είναι οι εργασίες (N), τόσο μεγαλύτερος είναι ο συνολικός χρόνος εκτέλεσης, αλλά η συμπεριφορά της καμπύλης παραμένει ίδια.

Συμπέρασμα:

Η αύξηση των μηχανών βελτιώνει την απόδοση του αλγορίθμου, αλλά το όφελος μειώνεται σταδιακά. Υπάρχει ένα σημείο όπου η προσθήκη επιπλέον μηχανών δεν προσφέρει σημαντική επιτάχυνση, γεγονός που επιβεβαιώνει τη θεωρητική μας ανάλυση για την πολυπλοκότητα του αλγορίθμου.



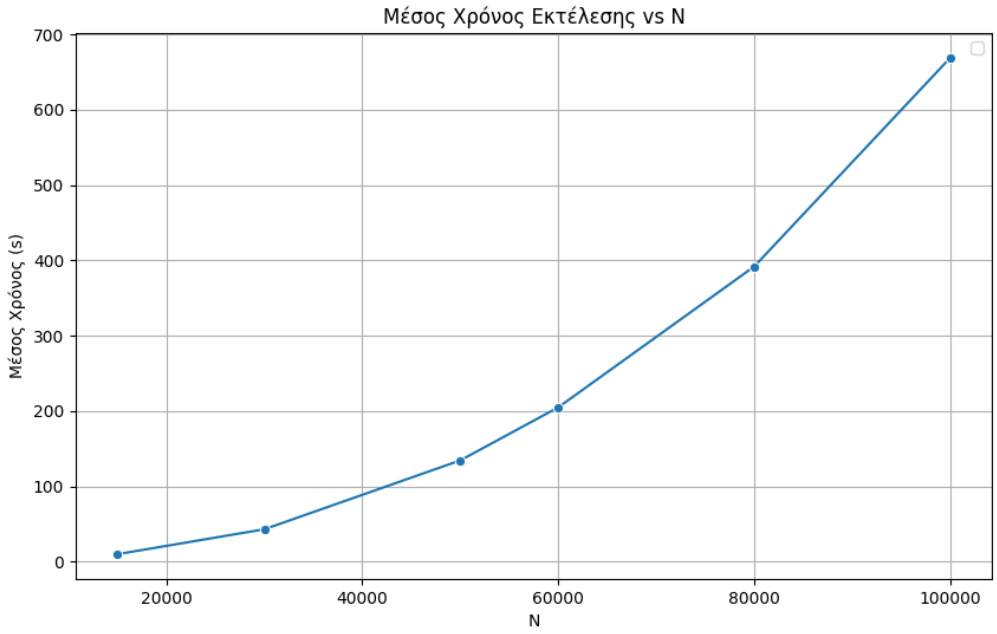
Εικόνα 16: Χρόνος Εκτέλεσης vs. N (ανά M)

Παρατηρούμε ότι**:**

- Ο χρόνος εκτέλεσης αυξάνεται σχεδόν γραμμικά καθώς αυξάνεται το N, επιβεβαιώνοντας την πολυωνυμική πολυπλοκότητα του αλγορίθμου.
- Όσο περισσότερες μηχανές διαθέτουμε (M), τόσο χαμηλότερος είναι ο συνολικός χρόνος εκτέλεσης, καθώς η κατανομή των εργασιών γίνεται πιο αποδοτική.
- Για χαμηλές τιμές M, η αύξηση του N οδηγεί σε μεγαλύτερη επιβάρυνση του χρόνου εκτέλεσης, ενώ για υψηλές τιμές M, η αύξηση του N έχει μικρότερη επίδραση.

Συμπέρασμα:

Ο αλγόριθμος Longest First επηρεάζεται έντονα από τον αριθμό των εργασιών, με τη χρονική του απόδοση να κλιμακώνεται αναμενόμενα. Παράλληλα, η αύξηση των μηχανών μειώνει σημαντικά τον χρόνο εκτέλεσης, ωστόσο η επίδραση αυτή μειώνεται όσο μεγαλώνει ο αριθμός των M, επιβεβαιώνοντας τα θεωρητικά όρια απόδοσης.



Εικόνα 17: Μέσος Χρόνος Εκτέλεσης vs. N

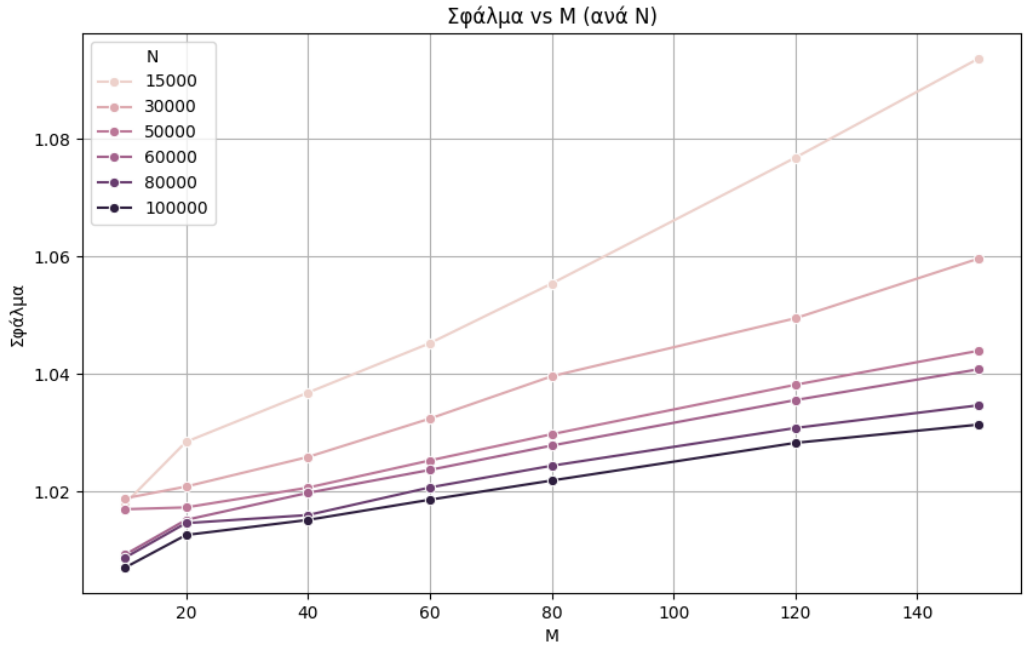
Παρατηρούμε ότι**:**

- Ο μέσος χρόνος εκτέλεσης αυξάνεται καθώς αυξάνεται ο αριθμός των εργασιών.
- Η σχέση μεταξύ N και χρόνου εκτέλεσης φαίνεται να ακολουθεί πολυωνυμική τάση, γεγονός που συμφωνεί με τη θεωρητική πολυπλοκότητα του αλγορίθμου.
- Για μικρές τιμές του N, η αύξηση του χρόνου εκτέλεσης είναι πιο ομαλή, ενώ για μεγάλες τιμές του N, η αύξηση γίνεται απότομη, γεγονός που αντικατοπτρίζει την αύξηση του συνολικού φόρτου εργασίας.

Συμπέρασμα:

Ο αλγόριθμος Longest First παρουσιάζει μία αναμενόμενη αύξηση στον χρόνο εκτέλεσης καθώς αυξάνεται το N, με τη μορφή πολυωνυμικής αύξησης. Αυτό επιβεβαιώνει ότι, αν και ο αλγόριθμος είναι αποδοτικός για μικρότερα σύνολα εργασιών, η απόδοσή του επιβαρύνεται σημαντικά καθώς το μέγεθος των δεδομένων μεγαλώνει.

Ανάλυση Σφάλματος / Απόκλισης



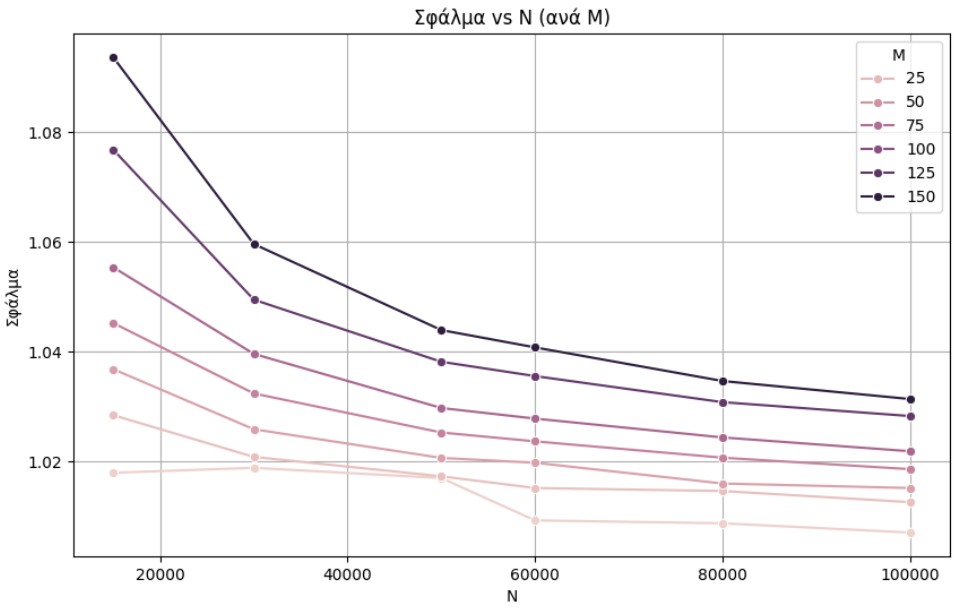
Εικόνα 18 : Σφάλμα vs. M (ανά N)

Παρατηρούμε ότι**:**

- Το σφάλμα αυξάνεται όσο αυξάνεται το M, υποδηλώνοντας ότι η κατανομή των εργασιών μεταξύ περισσότερων μηχανών μπορεί να επηρεάζει την προσέγγιση της βέλτιστης λύσης.
- Για μικρές τιμές του N, το σφάλμα εμφανίζεται υψηλότερο, αλλά καθώς N αυξάνεται, η αύξηση του σφάλματος γίνεται πιο ομαλή.
- Η τάση υποδεικνύει ότι, αν και ο αλγόριθμος προσαρμόζεται καλά για μεγάλες τιμές N, η αύξηση του M μπορεί να οδηγεί σε ελαφρώς μεγαλύτερη απόκλιση από τη βέλτιστη λύση.

Συμπέρασμα:

Η αύξηση του M τείνει να επιδεινώνει το σφάλμα του αλγορίθμου Longest First, ιδιαίτερα για μικρά N. Ωστόσο, για μεγαλύτερες τιμές N, η απόκλιση φαίνεται να σταθεροποιείται, γεγονός που υποδηλώνει ότι η επίδραση του M στην ακρίβεια της λύσης μειώνεται όταν το πρόβλημα έχει επαρκή αριθμό εργασιών.



Εικόνα 19: Σφάλμα vs. N (ανά M)

Παρατηρούμε ότι**:**

- Καθώς αυξάνεται το N, το σφάλμα μειώνεται για όλες τις τιμές του M, γεγονός που δείχνει ότι ο αλγόριθμος Longest First αποδίδει καλύτερα σε μεγάλα σύνολα εργασιών.
- Οι καμπύλες για μεγαλύτερες τιμές του M ξεκινούν από υψηλότερο σφάλμα, αλλά συγκλίνουν προς χαμηλότερες τιμές καθώς αυξάνεται το N.
- Για χαμηλές τιμές N, το σφάλμα είναι υψηλότερο, ειδικά όταν ο αριθμός των μηχανών είναι μεγαλύτερος, κάτι που δείχνει ότι η εκτέλεση του αλγορίθμου επηρεάζεται περισσότερο από τη σχέση N/M όταν τα δεδομένα είναι περιορισμένα.

Συμπέρασμα:

Ο αλγόριθμος Longest First τείνει να βελτιώνει την ακρίβειά του καθώς το N αυξάνεται, κάτι που υποδηλώνει ότι η απόδοσή του σταθεροποιείται για μεγάλα προβλήματα. Παρότι για μικρές τιμές N παρατηρείται αυξημένο σφάλμα, όσο περισσότερες εργασίες εκτελούνται, τόσο πιο κοντά βρίσκεται η λύση στον θεωρητικό βέλτιστο χρόνο.

Συμπεράσματα για τον Αλγόριθμο Longest First (LF)

Ο αλγόριθμος Longest First (LF) έδειξε μια ξεκάθαρη συμπεριφορά όσον αφορά τον χρόνο εκτέλεσης, την πολυπλοκότητα και τον παράγοντα προσέγγισης. Μέσα από τα

πειραματικά αποτελέσματα, επιβεβαιώσαμε ότι ο LF λειτουργεί αποδοτικά σε μεγάλα σύνολα εργασιών και παρουσιάζει σταδιακή βελτίωση στην ακρίβειά του καθώς αυξάνεται το N.

Πολυπλοκότητα και Απόδοση

- Ο χρόνος εκτέλεσης του LF μειώνεται όσο αυξάνεται ο αριθμός των μηχανών M, κάτι που υποδεικνύει ότι ο αλγόριθμος αξιοποιεί αποδοτικά τους διαθέσιμους πόρους.
- Παρατηρήσαμε ότι η αύξηση του N οδηγεί σε σχεδόν γραμμική αύξηση του χρόνου εκτέλεσης, επιβεβαιώνοντας τη θεωρητική πολυπλοκότητα O(dmlog n).
- Όταν ο αριθμός μηχανών είναι μικρός, ο χρόνος εκτέλεσης είναι σημαντικά υψηλότερος, καθώς ο προγραμματισμός των διαγνωστικών δοκιμών γίνεται λιγότερο ευέλικτος.

Παράγοντας Προσέγγισης και Σφάλμα

- Το σφάλμα του αλγορίθμου μειώνεται καθώς το N αυξάνεται, γεγονός που δείχνει ότι η προσέγγισή του γίνεται πιο ακριβής για μεγαλύτερα σύνολα εργασιών.
- Όταν ο αριθμός των μηχανών αυξάνεται, παρατηρούμε μια ελαφρά αύξηση στο σφάλμα, ειδικά για μικρές τιμές του N, γεγονός που πιθανώς σχετίζεται με την ανάγκη πιο ισορροπημένου καταμερισμού των εργασιών.
- Παρά το γεγονός ότι ο LF είναι ένας άπληστος αλγόριθμος (greedy), καταφέρνει να παρέχει ικανοποιητικά αποτελέσματα, με το σφάλμα να συγκλίνει σε σταθερές τιμές κοντά στο 1.02-1.05, επιβεβαιώνοντας την αποτελεσματικότητά του.

4.3 Αλγόριθμος Preemptive Lazy Binning (PLB)

Σε αυτήν την ενότητα εστιάζουμε στον **Preemptive Lazy Binning (PLB)**, έναν αλγόριθμο που επιτρέπει την προ-διακοπή των εργασιών (preemption) και στοχεύει στην ελαχιστοποίηση του αριθμού βαθμονομήσεων σε ένα μόνο μηχάνημα. Ο PLB διατηρεί βέλτιστη λύση ($O(n^2)$ πολυπλοκότητα), αξιοποιώντας την ιδέα του Lazy Binning σε εργασίες με αυθαίρετους χρόνους εκτέλεσης.

4.3.1 Παράμετροι Πειραματισμού και Διαδικασία Προσομοίωσης

Για να μελετήσουμε την απόδοση του PLB, ορίσαμε τις εξής βασικές παραμέτρους:

- **N** (Number of Jobs):
    - Δοκιμάζουμε διάφορα μεγέθη N (π.χ. 1.000, 5.000, 10.000, …), ώστε να δούμε πώς κλιμακώνεται ο χρόνος εκτέλεσης.
- **T** (Time Intervals):
    - Πειραματιζόμαστε τόσο με **σταθερό** T όσο και με **δυναμικό** T που εξαρτάται από το N (T = sqrt{N} και T=log 2{N}).
    - Στόχος είναι να ελέγξουμε αν η διάρκεια του διαστήματος βαθμονόμησης επηρεάζει αισθητά την απόδοση του αλγορίθμου.
- **(pmin,pmax)** (Minimum & Maximum Processing Time):
    - Ρυθμίζουν το εύρος διάρκειας των εργασιών. Χρησιμοποιούμε τυπικά pmin=1 και pmax=100, ώστε να εξασφαλίσουμε ποικιλία στις εκτελεστικές απαιτήσεις των jobs.
- **d_ratio** (Deadline Ratio):
    - Ελέγχουμε πόσο «χαλαρές» ή «αυστηρές» είναι οι προθεσμίες, ορίζοντας d_ratio=1.5. Όσο μεγαλύτερο το d_ratio, τόσο μεγαλύτερο το διάστημα μεταξύ 0 και dmax.

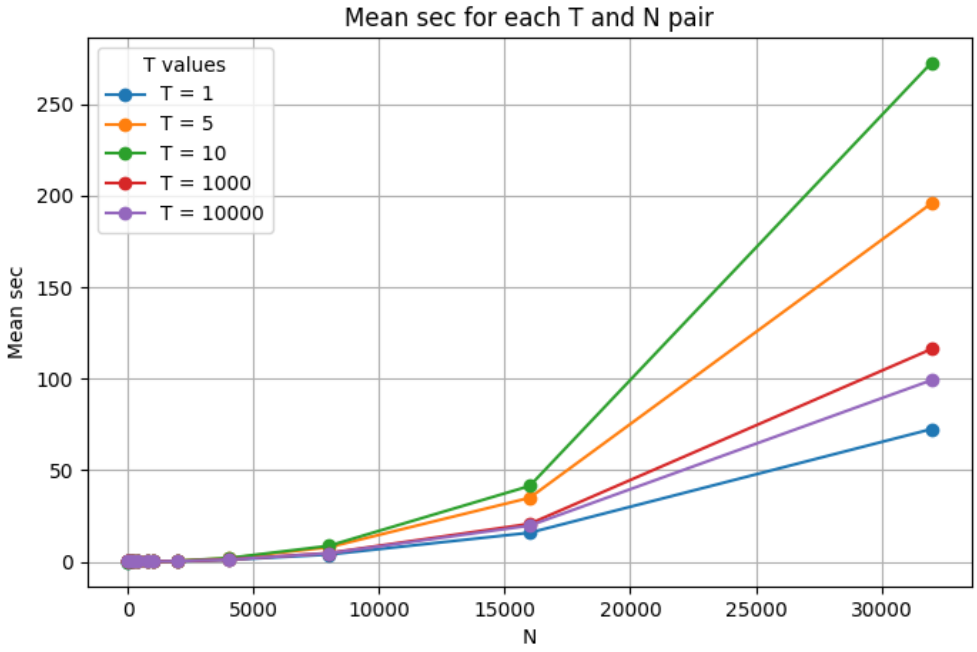
Σε κάθε περίπτωση, **καταγράφουμε**:

1. **Χρόνο Εκτέλεσης** του PLB (Time).
2. **Αριθμό Βαθμονομήσεων** που χρησιμοποιούνται τελικά.
3. **Πλήθος Εργασιών** που εκτελέστηκαν σε κάθε διάστημα βαθμονόμησης.

4.3.2 Αποτελέσματα και Διαγράμματα

Παρουσιάζουμε τα γραφήματα που προέκυψαν από τα πειράματα.

Στόχος μας είναι: Να εξετάσουμε πώς εξελίσσεται ο χρόνος εκτέλεσης, καθώς μεταβάλλεται το N και το T.



Εικόνα 20: Μέσος Χρόνος Εκτέλεσης vs. N (για διαφορετικές σταθερές τιμές T)

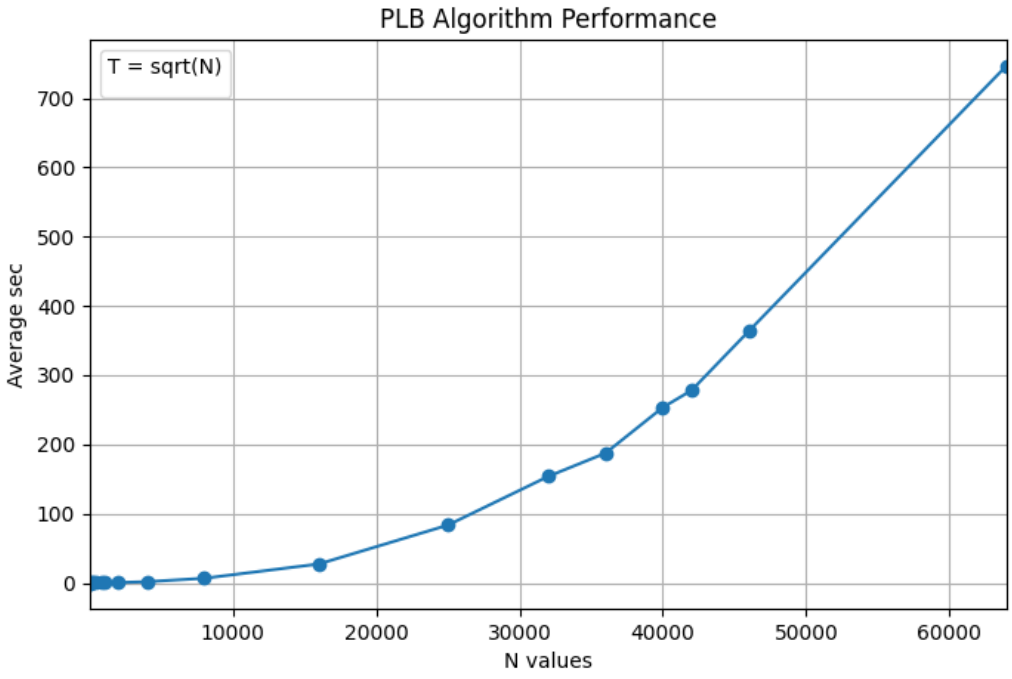
Παρατηρούμε ότι**:**

- **Μικρό N**: Για λίγες εργασίες, ο χρόνος εκτέλεσης παραμένει σχετικά χαμηλός για όλα τα T. Ο αλγόριθμος δεν επιβαρύνεται ιδιαιτέρως, καθώς οι εργασίες ολοκληρώνονται γρήγορα.
- **Μεσαίο-Μεγάλο N**: Καθώς αυξάνεται ο αριθμός των εργασιών, οι διαφορές μεταξύ των τιμών T γίνονται πιο εμφανείς.
    - Ορισμένες τιμές T (π.χ. 10) οδηγούν σε απότομη άνοδο του χρόνου εκτέλεσης, υποδεικνύοντας ότι ο συχνός προγραμματισμός των βαθμονομήσεων αυξάνει το overhead.
    - Αντίθετα, για πολύ μικρό T (π.χ. 1) ή πολύ μεγάλο T (π.χ. 10000), ο χρόνος εκτέλεσης διαμορφώνεται σε σχετικά χαμηλότερα επίπεδα.
- **Διάστημα Βαθμονόμησης T**: Ερμηνεύουμε ότι η επιλογή του T επιδρά σημαντικά στη διαχείριση των εργασιών:
    - Με **πολύ μικρό T**, πραγματοποιούμε πολλές βαθμονομήσεις, αλλά η διαχείριση γίνεται σε μικρά τμήματα, μειώνοντας το συνολικό overhead.
    - Με **πολύ μεγάλο T**, εκτελούμε λιγότερες βαθμονομήσεις, όμως οι εργασίες ίσως χωρούν όλες σε ελάχιστα διαστήματα, διατηρώντας χαμηλό overhead.

- Ενδιάμεσες τιμές T (π.χ. 5, 10) ενδέχεται να προκαλέσουν μεγαλύτερη δυσκολία στον προγραμματισμό, ανεβάζοντας τον χρόνο εκτέλεσης σε μεγάλα N.

Συμπέρασμα:

Η συμπεριφορά του PLB όσον αφορά τον χρόνο εκτέλεσης εξαρτάται έντονα από το πώς ορίζουμε το T. Για πολύ μικρό ή πολύ μεγάλο T, η προ-διακοπή και η διαχείριση των διαστημάτων μπορούν να λειτουργήσουν αποτελεσματικά, ενώ ενδιάμεσες τιμές T μπορεί να δημιουργήσουν μεγαλύτερο overhead όταν ο αριθμός εργασιών N γίνεται μεγάλος.



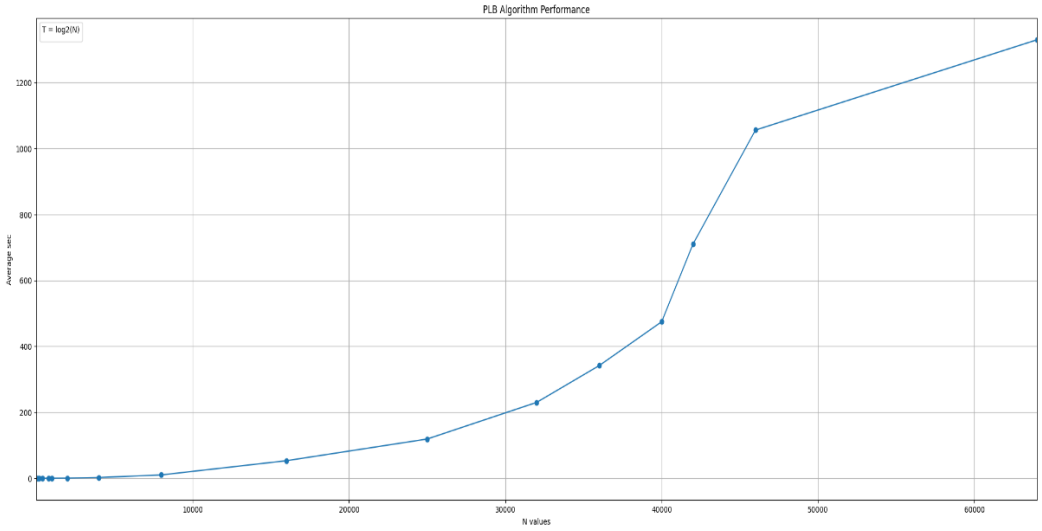
Εικόνα 21: Μέσος Χρόνος Εκτέλεσης vs. N (για τιμές T= sqrt{N})

Παρατηρούμε ότι**:**

- **Αύξηση του T**: Καθώς το N μεγαλώνει, το T αυξάνεται με ρυθμό sqrt{N}. Αυτό σημαίνει ότι τα διαστήματα βαθμονόμησης «επεκτείνονται» σταδιακά, επιτρέποντας περισσότερες εργασίες να ολοκληρωθούν σε ένα «bin».

- **Κλιμάκωση Χρόνου Εκτέλεσης**: Ο χρόνος εκτέλεσης αυξάνεται πιο απότομα από γραμμικά, επιβεβαιώνοντας ότι ο αλγόριθμος αντιμετωπίζει αυξημένη πολυπλοκότητα σε πολύ μεγάλα N.
- **Συμβιβασμός**: Η επιλογή T=sqrt{N} προσφέρει μια ενδιαφέρουσα ισορροπία, με λιγότερες βαθμονομήσεις από ό,τι για σταθερό T πολύ μικρό, αλλά παραμένει ικανή να χειριστεί επαρκώς την κατανομή των εργασιών καθώς το N αυξάνεται.

Συμπέρασμα: Με T=sqrt{N}, ο αλγόριθμος PLB εμφανίζει συνεχή αύξηση του χρόνου εκτέλεσης για μεγάλα N. Παρότι το κόστος μεγαλώνει αρκετά, το προσαρμοζόμενο T αξιοποιεί πιο αποδοτικά τα διαστήματα βαθμονόμησης σε σύγκριση με μια αυθαίρετη σταθερή τιμή, επιβεβαιώνοντας ότι η στρατηγική ορισμού του T μπορεί να βελτιώσει την απόδοση ανάλογα με το μέγεθος του προβλήματος.



Εικόνα 22: Μέσος Χρόνος Εκτέλεσης vs. N (για τιμές T=log 2{N})

Παρατηρούμε ότι**:**

- **Μικρές τιμές N**: Για N μέχρι περίπου 10.000, ο χρόνος εκτέλεσης παραμένει σε χαμηλά επίπεδα, υποδεικνύοντας ότι η επιλογή T=log2(N) είναι αρκετά ευέλικτη όταν το πρόβλημα είναι μικρό ή μεσαίο.
- **Μεσαίες προς μεγάλες τιμές N**: Καθώς N αυξάνεται πάνω από τις 20.000–30.000 εργασίες, ο χρόνος εκτέλεσης ανεβαίνει απότομα. Αυτό συμβαίνει γιατί, παρά το ότι το T αυξάνεται λογαριθμικά, ο συνολικός φόρτος εργασίας ενδέχεται να αυξάνεται ταχύτερα, απαιτώντας περισσότερη επεξεργασία από τον αλγόριθμο.

- **Ρυθμιστικός ρόλος του log2(N)**: Η λογαριθμική αύξηση του T επιτρέπει την αποφυγή υπερβολικά συχνών βαθμονομήσεων, ωστόσο σε πολύ μεγάλα N ο χρόνος εκτέλεσης μπορεί και πάλι να γίνει υψηλός λόγω του γενικότερου κόστους προγραμματισμού.

Συμπέρασμα:

Το σχήμα T=log2(N) προσφέρει μια ενδιαφέρουσα ισορροπία για το PLB σε μεσαία μεγέθη προβλήματος, κρατώντας χαμηλό τον αριθμό βαθμονομήσεων. Ωστόσο, καθώς το N αυξάνεται σημαντικά, η λογαριθμική αύξηση του T δεν επαρκεί για να συγκρατήσει πλήρως τον χρόνο εκτέλεσης, επιβεβαιώνοντας ότι η σχέση N–T είναι καθοριστική για την αποδοτικότητα του αλγορίθμου.

Συμπέρασμα για τον Αλγόριθμο PLB (Preemptive Lazy Binning)

Τα πειράματα έδειξαν ότι ο χρόνος εκτέλεσης του PLB εξαρτάται άμεσα από την επιλογή του παραμέτρου T.

- **Όταν το T είναι σταθερό**, ο χρόνος εκτέλεσης αυξάνεται γρήγορα με το N, καθώς απαιτούνται περισσότερες βαθμονομήσεις.
- **Για T=sqrt{N}**, παρατηρήσαμε μια πιο ισορροπημένη κλιμάκωση, που διατηρεί τον χρόνο εκτέλεσης σε αποδεκτά επίπεδα.
- **Με T=log2(N)**, ο χρόνος αρχικά αυξάνεται ομαλά, αλλά για μεγάλα N η εκθετική τάση γίνεται εμφανής.

Γενικά, η **καλύτερη επιλογή** εξαρτάται από το μέγεθος του προβλήματος:

- Για μικρά και μεσαία N, το T=log2(N) είναι αποδοτικό.
- Για μεγάλα N, το T= sqrt{N} προσφέρει καλύτερη ισορροπία μεταξύ χρόνου εκτέλεσης και ποιότητας προγραμματισμού.

Ο PLB είναι αποδοτικός, αλλά απαιτεί σωστή ρύθμιση του T ώστε να επιτυγχάνει γρήγορη εκτέλεση χωρίς περιττές βαθμονομήσεις.

Κεφάλαιο 5. Εκτέλεση των αλγορίθμων

Οι αλγόριθμοί μας είναι γραμμένοι στη γλώσσα προγραμματισμού **Python**. Τα αρχεία μας είναι τα εξής:

- goesman.py
- LF.py
- PLB.py

5.1 Τρόπος Εκτέλεσης των Αλγορίθμων

Για να εκτελέσουμε τους αλγορίθμους, αρκεί να τρέξουμε τα αντίστοιχα αρχεία τους. Είναι σημαντικό να διασφαλίσουμε ότι κάθε αρχείο βρίσκεται στον ίδιο φάκελο (path) με το αρχείο εισόδου που απαιτείται για την εκτέλεσή του.

Κατά την εκτέλεση, οι αλγόριθμοι θα εκτυπώσουν αυτόματα τα αποτελέσματά τους στην οθόνη μας. Δεν απαιτείται καμία επιπλέον ενέργεια από εμάς, καθώς είναι ρυθμισμένοι να επεξεργάζονται τα δεδομένα του αρχείου εισόδου και να εμφανίζουν τα αποτελέσματα απευθείας στην κονσόλα.

5.2 Μορφή των Δεδομένων για Εισαγωγή

5.2.1 Αρχείο είσοδου για τον αλγόριθμο για Scheduling σε Τρεις Αφιερωμένες Μηχανές

Το αρχείο εισόδου, από το οποίο ο αλγόριθμος για **Scheduling σε Τρεις Αφιερωμένες Μηχανές** θα διαβάσει τα δεδομένα προς επεξεργασία, πρέπει να περιέχει δύο βασικές μεταβλητές. Οι μεταβλητές αυτές είναι οι μηχανές που συμμετέχουν σε κάθε εργασία και ο **χρόνος εκτέλεσης** της εργασίας αυτής.

- **Οι μηχανές (Mx, My)**: Αναφέρονται στις διαθέσιμες μηχανές που μπορούν να εκτελέσουν μια εργασία. Κάθε εργασία μπορεί να εκτελείται είτε από μία είτε από δύο μηχανές.
- **Ο χρόνος εκτέλεσης (T)**: Είναι ο συνολικός χρόνος που απαιτείται για την ολοκλήρωση της εργασίας από τις μηχανές που την εκτελούν.

Το αρχείο αποτελείται από διαδοχικές γραμμές που περιγράφουν τις εργασίες και τον χρόνο εκτέλεσής τους. Κάθε γραμμή έχει τη μορφή:

Mx : T

Mx My : T

όπου:

- Mx, My είναι οι μηχανές που εκτελούν την εργασία.
- T είναι ο αντίστοιχος χρόνος εκτέλεσης.

5.2.2 Αρχείο είσοδου για τον αλγοριθμο Longest First (LF)

Το αρχείο εισόδου, από το οποίο ο αλγόριθμος θα διαβάσει τα δεδομένα προς επεξεργασία, περιέχει πληροφορίες σχετικά με τις εργασίες και τις μηχανές που τις εκτελούν. Κάθε γραμμή περιγράφει μια εργασία, καθορίζοντας τις δύο μηχανές που την εκτελούν καθώς και τον συνολικό χρόνο εκτέλεσής της.

Το αρχείο αποτελείται από διαδοχικές γραμμές της μορφής:

Mx My T

όπου:

- **Mx**: Η πρώτη μηχανή που εκτελεί την εργασία.
- **My**: Η δεύτερη μηχανή που εκτελεί την εργασία.
- **T**: Ο συνολικός χρόνος εκτέλεσης της εργασίας από τις μηχανές Mx και My.

5.2.3 Αρχείο είσοδου για τον αλγοριθμο PLB

Το αρχείο εισόδου, από το οποίο ο αλγόριθμος θα διαβάσει τα δεδομένα προς επεξεργασία, περιέχει πληροφορίες σχετικά με τις εργασίες και τις χρονικές παραμέτρους τους.

Κάθε σύνολο δεδομένων αποτελείται από:

1. **T**: Ο συνολικός διαθέσιμος χρόνος για την εκτέλεση των εργασιών.
2. **N**: Ο αριθμός των εργασιών που περιλαμβάνονται στο πείραμα.
3. **Λίστα εργασιών**: Κάθε εργασία περιγράφεται από τέσσερις τιμές:
    - **job_id**: Ο μοναδικός αριθμός αναγνώρισης της εργασίας.
    - **r**: Η χρονική στιγμή άφιξης της εργασίας (σταθερή τιμή 0).
    - **d**: Η προθεσμία ολοκλήρωσης της εργασίας.
    - **p**: Ο απαιτούμενος χρόνος εκτέλεσης της εργασίας.

Το αρχείο αποτελείται από διαδοχικές γραμμές της μορφής:

T

N

job_id r d p

job_id r d p

όπου:

- **T:** Ορίζει το συνολικό διαθέσιμο χρόνο μέσα στον οποίο πρέπει να εκτελεστούν οι εργασίες.
- **N**: Καθορίζει τον αριθμό των εργασιών που περιλαμβάνονται
- **job_id:** Ένας μοναδικός αριθμός που ταυτοποιεί κάθε εργασία
- **r:** Αντιπροσωπεύει τη χρονική στιγμή κατά την οποία η εργασία γίνεται διαθέσιμη για εκτέλεση. Στο συγκεκριμένο πρόβλημα έχει σταθερή τιμή 0 για όλες τις εργασίες.
- **d:** Καθορίζει τη μέγιστη χρονική στιγμή μέχρι την οποία η εργασία πρέπει να έχει ολοκληρωθεί.
- **p:** Ο συνολικός χρόνος που απαιτείται για την εκτέλεση της εργασίας.

Κεφάλαιο 6. Συμπεράσματα

Σε αυτή τη διπλωματική εργασία, υλοποιήσαμε, οπτικοποιήσαμε και αξιολογήσαμε πειραματικά τρεις διαφορετικούς αλγορίθμους χρονοπρογραμματισμού, με στόχο την ανάλυση της απόδοσής τους σε διαφορετικά σύνολα δεδομένων. Οι αλγόριθμοι αυτοί αφορούν διαφορετικές προσεγγίσεις προγραμματισμού εργασιών και παρέχουν εναλλακτικές στρατηγικές αντιμετώπισης σύνθετων NP-πλήρων προβλημάτων.

1. Συνοπτικά Συμπεράσματα για Κάθε Αλγόριθμο

- **Αλγόριθμος Scheduling σε Τρεις Αφιερωμένες Μηχανές**

  Ο συγκεκριμένος αλγόριθμος εφαρμόζει μια προσεγγιστική στρατηγική με θεωρητικά όρια απόδοσης. Τα αποτελέσματα έδειξαν ότι ο makespan που παράγεται είναι σε συμφωνία με τον θεωρητικό παράγοντα προσέγγισης 7/6. Επιπλέον, παρατηρήσαμε ότι η απόδοσή του εξαρτάται από το πλήθος των εργασιών (N) και την κατανομή τους μεταξύ των μηχανών. Όταν το ποσοστό των εργασιών που απαιτούν περισσότερες από μία μηχανές (R) βρίσκεται σε ενδιάμεσες τιμές, το makespan αυξάνεται ελαφρώς, ενώ για ακραίες τιμές του R, ο αλγόριθμος παρουσιάζει πιο σταθερή απόδοση.

- **Αλγόριθμος Longest First (LF)**

  Ο αλγόριθμος Longest First (LF) ακολουθεί μια greedy στρατηγική, δίνοντας προτεραιότητα στις μεγαλύτερες εργασίες. Από τα πειράματα, προέκυψε ότι η απόδοσή του επηρεάζεται σημαντικά από τον αριθμό των μηχανών (M) και το πλήθος των εργασιών (N). Καθώς αυξάνεται ο αριθμός των μηχανών, ο χρόνος εκτέλεσης μειώνεται σημαντικά, ενώ η ακρίβεια της λύσης βελτιώνεται. Ωστόσο, η αύξηση του πλήθους των εργασιών τείνει να αυξάνει το σφάλμα του αλγορίθμου. Η πολυπλοκότητα του LF είναι αποδεκτή για μεγάλα δεδομένα, καθιστώντας τον κατάλληλο για πρακτικές εφαρμογές.

- **Αλγόριθμος Preemptive Lazy Binning (PLB)**

  Ο PLB διαχειρίζεται εργασίες με δυναμικό χρονοπρογραμματισμό, επιτρέποντας προεκχωρήσεις (preemptions) και περιοδικές βαθμονομήσεις (calibrations). Τα πειράματα έδειξαν ότι η απόδοσή του εξαρτάται από τον παράγοντα T, ο οποίος καθορίζει τον αριθμό των βαθμονομήσεων που επιτρέπονται. Για διαφορετικές τιμές του T (σταθερό, log(N), sqrt(N)), παρατηρήσαμε πως ο χρόνος εκτέλεσης αυξάνεται καθώς μεγαλώνει το N, ενώ το συνολικό κόστος των βαθμονομήσεων παρουσιάζει βέλτιστη συμπεριφορά όταν το T κλιμακώνεται ως log(N). Γενικά,

ο PLB αποδείχθηκε αποτελεσματικός στην επίτευξη καλής κατανομής των εργασιών με ελάχιστες απαιτούμενες βαθμονομήσεις.

2. Συγκριτική Ανάλυση και Τελικές Παρατηρήσεις

- Η **υπολογιστική πολυπλοκότητα** των αλγορίθμων είναι συμβατή με τη θεωρητική ανάλυση. Ο αλγόριθμος Scheduling σε Τρεις Αφιερωμένες Μηχανές παρουσίασε πολυωνυμική συμπεριφορά, ο Longest First ακολούθησε μια O(n log n) στρατηγική, ενώ ο PLB είχε πολυπλοκότητα $O(n^2)$, διατηρώντας ωστόσο αποδοτική συμπεριφορά.
- Ο **παράγοντας προσέγγισης** επαληθεύτηκε στα πειράματα, καθώς οι αλγόριθμοι παρουσίασαν αποκλίσεις από το βέλτιστο εντός των θεωρητικών ορίων.
- Η **κλιμάκωση των αλγορίθμων** έδειξε ότι ενώ ο LF μειώνει τον χρόνο εκτέλεσης καθώς αυξάνονται οι μηχανές, το σφάλμα του επηρεάζεται από το μέγεθος του προβλήματος. Ο PLB προσφέρει μια προσαρμοστική προσέγγιση, διατηρώντας βέλτιστη απόδοση με κατάλληλη ρύθμιση του T.
- Οι **ειδικές περιπτώσεις των αλγορίθμων** ανέδειξαν ότι σε ορισμένες παραμετροποιήσεις, όπως μεγάλες τιμές R ή χαμηλές τιμές T, η απόδοση μπορεί να επιδεινωθεί. Αυτό δείχνει την ανάγκη για σωστή επιλογή των παραμέτρων ανάλογα με την εφαρμογή.

Συμπερασματικά, η εργασία αυτή ανέδειξε τη σημασία των προσεγγιστικών αλγορίθμων στον χρονοπρογραμματισμό, επιβεβαίωσε τα θεωρητικά όρια της απόδοσής τους και άνοιξε νέες κατευθύνσεις για περαιτέρω βελτίωση και εφαρμογή των τεχνικών αυτών.

Βιβλιογραφία

[1] Eric Angel, Evripidis Bampis, Vincent Chau, Vassilis Zissimopoulos. *On the complexity of minimizing the total calibration cost.* 11th International Frontiers of Algorithmics Workshop (FAW 2017), Jun 2017, Chengdu, China. pp. 1–12, 10.1007/978-3-319-59605-1_1. hal-01630651

[2] *An approximation algorithm for scheduling on three dedicated machines* Michel X. Goemans

Department of Mathematics, MIT, Cambridge, MA 02139, USA Received 13 December 1993; revised 30 September 1994

[3] *An Approximation Algorithm for Diagnostic Test Scheduling in Multicomputer Systems*

HENRYK KRAWCZYK AND MAREK KUBALE

[4] L. Bianco, P. Dell'Olmo and M. Speranza, Nonpreemptive scheduling of independent tasks with dedicated resources, Technical Report 320, I.A.S.I.-C.N.R., Roma (1991).

[5] J. Błażewicz, P. Dell'Olmo, M. Drozdowski and M. Speranza, Scheduling multiprocessor tasks on the three dedicated processors, Inform. Process. Lett. 41 (1992) 275–280.

[6] J. Błażewicz, P. Dell'Olmo, M. Drozdowski and M. Speranza, Corrigendum to "Scheduling multiprocessor tasks on three dedicated processors, Inform. Process. Lett. 41 (1992) 275–280", Inform. Process. Lett. 49 (1994) 269–270.

[7] E. Coffman Jr, M. Garey, D. Johnson and A. LaPaugh, Scheduling file transfers, SIAM J. Comput. (1985) 744–780.

[8] G. Bozoki and J. Richard, A branch-and-bound algorithm for the continuous task-process task shop scheduling problem, AIIE Trans. 2 (1970) 246–252.

[9] E. Amman and M. Dal Cin, "Efficient algorithms for comparison-based self-diagnosis," in *Proc. Self-Diagnosis Fault-Tolerance*, Tubingen, West Germany: ATTEMPTO-Verlag, 1981, pp. 1–18.

[10] E. G. Coffman, Jr., M. R. Garey, D. S. Johnson, and A. S. LaPaugh, "Scheduling file transfers in a distributed network," in *Proc. 2nd ACM SIGACT-SIGOPS Symp. Principles Distrib. Comput.*, Montreal, P.Q., Canada, 1983.

[11] W. Dobosiewicz, "Sorting by distributive partitioning," *Inform. Processing Lett.*, vol. 7, pp. 1–6, Jan. 1978.

[12] Bender, M.A., Bunde, D.P., Leung, V.J., McCauley, S., Phillips, C.A.: Efficient scheduling to minimize calibrations. In: 25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2013, pp. 280–287. ACM (2013).