ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Συγκριτική αξιολόγηση μεθόδων επίλυσης προβλημάτων χρονοπρογραμματισμού εργασιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΦΕΡΜΕΛΗΣ Δ. ΝΙΚΟΛΑΟΣ

Επιβλέπων : Δημήτριος Ασκούνης

Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2012

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ

Συγκριτική αξιολόγηση μεθόδων επίλυσης προβλημάτων χρονοπρογραμματισμού εργασιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΦΕΡΜΕΛΗΣ Δ. ΝΙΚΟΛΑΟΣ

Επιβλέπων : Δημήτριος Ασκούνης

Αναπληρωτής Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ……………...

....

Αθήνα, Νοέμβριος 2012

....

ΦΕΡΜΕΛΗΣ Δ. ΝΙΚΟΛΑΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Το πρόβλημα χρονοπρογραμματισμού παραγωγής είναι ένα από τα πιο πολύπλοκα συνδυαστικά προβλήματα που έχουν ποτέ διατυπωθεί. Αποτελεί ένα από τα σημαντικότερα θέματα που αφορούν ένα σύστημα παραγωγής και καθορίζει τη λειτουργικότητα και την απόδοσή του. Ο χρονικός προγραμματισμός παραγωγής θέτει το πρόβλημα του προγραμματισμού σε βραχυπρόθεσμη βάση των διατιθέμενων πόρων (ανθρώπινο δυναμικό, μηχανολογικός εξοπλισμός, οικονομικοί πόροι).

Πολλοί διαφορετικοί αλγόριθμοι έχουν αναπτυχθεί στην προσπάθεια επίλυσης του προβλήματος χρονοπρογραμματισμού παραγωγής για όλους τους τύπους συστημάτων παραγωγής (συστήματα συνεχούς ροής, παραγωγής κατά παραγγελία, κατασκευής έργων). Στη συγκεκριμένη εργασία περιορίζουμε την ανάλυσή μας στο σύστημα παραγωγής κατά παραγγελία, περιγράφοντας τις διάφορες προσεγγίσεις επίλυσής του και δίνοντας έμφαση στην τεχνική των χρονισμένων αυτομάτων.

Τελικός σκοπός της εργασίας αποτελεί η αξιολόγηση μίας τεχνικής βελτιστοποίησης χρονοπρογραμματισμού παραγωγής σε σύστημα job-shop με τη χρήση χρονισμένων αυτομάτων. Στο πλαίσιο της εργασίας αξιοποιείται το χρήσιμο εργαλείο UPPAAL ώστε να μοντελοποιηθούν κάποια παραδείγματα χρονοπρογραμματισμού παραγωγής με τη βοήθεια των χρονισμένων αυτομάτων και να υπολογιστούν οι λύσεις τους αλλά και η σύγκριση αυτών με τις λύσεις των ίδιων προβλημάτων όπως προκύπτουν μέσα από την αξιοποίηση μιας ευφυούς εργαλειοθήκης αλγορίθμων επίλυσης προβλημάτων χρονοπρογραμματισμού (LISA).

Λέξεις Κλειδιά: Χρονοπρογραμματισμός, χρονισμένα αυτόματα, αλγόριθμοι προγραμματισμού εργασιών

Abstract

The problem of production scheduling is one of the most complex combinatorial problems that have ever been. It is one of the most important issues concerning a production system and defines the functionality and performance. The timing of production raises the problem of planning in the short term the allocated resources (manpower, machinery, financial resources).

Many different algorithms have been developed in an attempt to solve the scheduling problem of production for all types of production systems (continuous flow systems, bespoke manufacturing, construction projects). In this paper we restrict our analysis to the production system in order, describing various approaches to solve it and emphasizing the technique of timed automata.

The ultimate aim of this work is the evaluation of a technique to optimize production scheduling system for job-shop using timed automata. In the context of labor exploited useful tool UPPAAL that modeled some examples scheduling production with the help of timed automata and calculate their solutions and compare them with the solutions of the same problems as they arise through the use of an intelligent algorithms toolbox solving scheduling problems (LISA).

Keywords: Scheduling**,** timed automata, task-scheduling algorithms

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στον τομέα Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΕΜΠ, στο πλαίσιο των δραστηριοτήτων του Εργαστηρίου Συστημάτων Αποφάσεων και Διοίκησης.

Το αντικείμενο της διπλωματικής εργασίας αφορά στις τεχνικές βελτιστοποίησης προγραμματισμού παραγωγής με έμφαση στη χρήση αυτομάτων και στην αξιοποίηση ευρετικών μεθόδων και εναλλακτικών αλγορίθμων προσέγγισης.

Οφείλω ιδιαίτερες ευχαριστίες στον Αναπληρωτή Καθηγητή ΕΜΠ κ. Δημήτρη Ασκούνη για την καθοδήγησή του και για το ότι ήταν πάντα πρόθυμος να βοηθήσει στην επίλυση οποιουδήποτε προβλήματος ανέκυπτε καθ’ όλη τη διάρκεια της συγγραφής της διπλωματικής εργασίας.

Ακόμα, θα ήθελα να ευχαριστήσω θερμά τον υποψήφιο Διδάκτορα και αγαπητό φίλο Δημήτρη Πανόπουλο για τη βοήθεια και καθοδήγησή του κατά τη διάρκεια της εργασίας όπως επίσης και για τις γνώσεις που μου μετέδωσε στο πλαίσιο της συνεργασίας μας.

ΦΕΡΜΕΛΗΣ Δ. ΝΙΚΟΛΑΟΣ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1. ΕΙΣΑΓΩΓΗ

1.1 ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ

Ο σκοπός της παρούσας διπλωματικής εργασίας είναι η αξιολόγηση μιας νέας μεθόδου βελτιστοποίησης χρονοπρογραμματισμού σε σύστημα παραγωγής κατά παραγγελία (job-shop) με τη βοήθεια των αυτομάτων πεπερασμένων καταστάσεων (finite state automata). Για την εφαρμογή της μεθόδου το σύστημα μοντελοποιείται σαν ένα χρονισμένο αυτόματο (timed automaton) όπου οι διάφοροι προγραμματισμοί της παραγωγής αντιστοιχούν στα μονοπάτια (paths) του αυτομάτου ενώ οι βέλτιστοι προγραμματισμοί αντιστοιχούν στα συντομότερα μονοπάτια (shortest paths). Στην συνέχεια τα ίδια προβλήματα χρονοπρογραμματισμού επιλύονται και με την βοήθεια μιας δοκιμασμένης βιβλιοθήκης αλγορίθμων επίλυσης προβλημάτων (LISA) και γίνεται σύγκριση των αποτελεσμάτων.

Οι μέθοδοι που προτείνονται στα κεφάλαια που ακολουθούν, προκειμένου να βρεθούν αυτά τα σύντομα μονοπάτια, στηρίζονται στην εφαρμογή των αλγορίθμων επαλήθευσης (verification algorithms), οι οποίοι έχουν σκοπό να εξετάσουν αν ισχύουν κάποιες ιδιότητες όσον αφορά όλες τις εκτελέσεις (runs) του αυτομάτου. Αυτοί οι αλγόριθμοι βασίζονται κατά κάποιο τρόπο στους αλγορίθμους γράφων (graph algorithms) που εξερευνούν μονοπάτια στο γράφημα μεταβάσεων (transition graph). Αυτή η μεθοδολογία εφαρμόστηκε και στην πράξη, χρησιμοποιώντας το εργαλείο UPPAAL σε κάποια παραδείγματα χρονοπρογραμματισμού παραγωγής συστημάτων job-shop. Έτσι, καταφέραμε να βρούμε κάθε φορά το βέλτιστο (ή σχεδόν το βέλτιστο) ολικό χρόνο διεκπεραίωσης ορισμένων εργασιών σε πολύ καλό χρόνο υπολογισμού. Πριν όμως καταλήξουμε στη συγκεκριμένη εφαρμογή, αναλύονται διεξοδικά τόσο η περιοχή του προγραμματισμού παραγωγής όσο και των αλγορίθμων που χρησιμοποιούνται στην εργασία.

1.2 ΟΡΓΑΝΩΣΗ ΤΟΥ ΤΟΜΟΥ

Η δομή που ακολουθήθηκε στην συγγραφή αυτής της διπλωματικής είναι η κάτωθι. Αρχικά, εξηγείται εκτενέστερα στο δεύτερο κεφάλαιο η έννοια του προγραμματισμού παραγωγής, καθώς γίνεται αναφορά στους τύπους συστημάτων παραγωγής και στα διάφορα θέματα (παραμέτρους) που έχουν να κάνουν με τη λειτουργία και το σχεδιασμό τους. Στη συνέχεια, περιοριζόμαστε στο χρονοπρογραμματισμό ενός συγκεκριμένου συστήματος παραγωγής (job-shop) αναλύοντας τα χαρακτηριστικά του.

Στο τρίτο κεφάλαιο, περιγράφονται οι διάφοροι αλγόριθμοι που χρησιμοποιούνται για να λύσουν το πρόβλημα του προγραμματισμού παραγωγής. Έτσι, γνωστοποιούνται οι μαθηματικές τεχνικές, ο γραφικός προγραμματισμός, οι ευρετικοί αλγόριθμοι, η τεχνητή νοημοσύνη, τα νευρωνικά δίκτυα, ο αλγόριθμος κατωφλίου, οι γενετικοί αλγόριθμοι, η τεχνική της έρευνας ταμπού καθώς και η edge-finder. Συγκεκριμένα, παρουσιάζονται ιστορικά στοιχεία για όλες τις τεχνικές, δίνονται παραδείγματα, αναλύεται ο τρόπος λειτουργίας τους και συνάγονται κάποια γρήγορα συμπεράσματα για την αποτελεσματικότητα τους στην διαδικασία της βελτιστοποίησης.

Στο τέταρτο κεφάλαιο, εξηγείται αναλυτικά πως μπορεί μια εργασία, ενός προβλήματος προγραμματισμού παραγωγής συστήματος job-shop, να μοντελοποιηθεί με ένα αυτόματο, του οποίου οι καταστάσεις θα παριστάνουν την πρόοδο των εργασιών κατά τη διάρκεια των βημάτων της, και οι μεταβάσεις θα αντιστοιχούν στην έναρξη και λήξη των βημάτων. Για το σκοπό αυτό, γίνεται μια εκτεταμένη παρουσίαση των αυτομάτων, και ιδιαίτερα των χρονισμένων αυτομάτων, ενώ ταυτόχρονα περιγράφονται κάποιοι βασικοί αλγόριθμοι που εξερευνούν τους γράφους μεταβάσεων ενός αυτόματου καθώς και αλγόριθμοι που συντελούν στη λύση προβλημάτων προσεγγισιμότητας (reachability) για χρονισμένα αυτόματα. Στο τέλος αυτού του κεφαλαίου, μοντελοποιούμε το πρόβλημα του χρονοπρογραμματισμού παραγωγής συστήματος job-shop, χρησιμοποιώντας ένα ιδιαίτερο είδος ακυκλικών χρονισμένων αυτομάτων. Έτσι, η εύρεση του βέλτιστου προγραμματισμού παραγωγής αντιστοιχεί στην εύρεση, με τη βοήθεια των αλγορίθμων προσεγγισιμότητας, του συντομότερου (υπό την έννοια του χρόνου που πέρασε) μονοπατιού στο αυτόματο.

Στο πέμπτο κεφάλαιο γίνεται αναφορά στη βιβλιοθήκη αλγορίθμων και το λογισμικό πακέτο LiSA το οποίο και χρησιμοποιήσαμε για την εναλλακτική επίλυση των προβλημάτων χρονοπρογραμματισμού. Αναλύουμε τον τρόπο που εργαζόμαστε με αυτό το εργαλείο καθώς και τα είδη των αλγορίθμων που χρησιμοποιεί.

Στο έκτο κεφάλαιο πλέον κάνουμε αναφορά στο είδος των προβλημάτων που επιλύσαμε με τα δύο διαφορετικά μοντέλα, στις μεθόδους – αλγορίθμους – προτεραιότητες που εφαρμόσαμε ενώ παρουσιάζουμε και τα αριθμητικά δεδομένα των προβλημάτων σε μορφή πινάκων δίνοντας στην συνέχεια παραδείγματα της επίλυσης αυτών βήμα προς βήμα και με τα δύο εργαλεία.

Στο έβδομο και τελευταίο κεφάλαιο παρουσιάζουμε τα χρονικά αποτελέσματα της μελέτης και των πειραμάτων μας σε πίνακες,διαχωρισμένα ανά μοντέλο , ανά προτεραιότητα – αλγόριθμο και ανά ομάδα προβλημάτων. Στην συνέχεια έχουμε συλλέξει σε έναν συγκεντρωτικό πίνακα τα αποτελέσματα αυτά ώστε υπολογίζοντας τους χρονικούς μέσους όρους να εξάγουμε όσο το δυνατόν πιο ασφαλή συμπεράσματα .Η διπλωματική κλείνει με τα συμπεράσματα αυτής της προσπάθειας και με τις προοπτικές για καλύτερη εκμετάλλευση των δυνατοτήτων του UPPAAL σε συνδυασμό με τη διαμόρφωση του μοντέλου των αυτομάτων και των αλγορίθμων που αξιοποιεί η βιβλιοθήκη LiSA.

ΚΕΦΑΛΑΙΟ 2

Εισαγωγή στον Χρονοπρογραμματισμό Παραγωγής

2. ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΠΑΡΑΓΩΓΗΣ

Ο χρονοπρογραμματισμός της παραγωγής απαντάται στα πλαίσια μια ευρείας γκάμας οικονομικών δραστηριοτήτων. Περιλαμβάνει την εκπλήρωση παραγγελιών οι οποίες δεσμεύουν ορισμένους πόρους για ορισμένες χρονικές περιόδους. Αυτοί οι πόροι δεν είναι απεριόριστοι. Οι παραγγελίες που πρέπει να πραγματοποιηθούν αποτελούνται από στοιχειώδη τμήματα που ονομάζονται *εργασίες*. Κάθε εργασία απαιτεί ορισμένα ποσά συγκεκριμένων πόρων για ένα συγκεκριμένο χρόνο που ονομάζεται χρόνος επεξεργασίας. Μεταξύ των πόρων αυτών συγκαταλέγονται οι μηχανές, το ανθρώπινο δυναμικό, οι πρώτες ύλες, κλπ. Επίσης, τα λειτουργικά κόστη του παραγωγικού συστήματος πρέπει να κυμαίνονται σε λογικά πλαίσια.

Οι τυπικές λειτουργίες του προγραμματισμού εργασιών περιλαμβάνουν την ανάθεση πόρων σε εργασίες, τον καθορισμό της σειράς εκτέλεσης των εργασιών (δρομολόγηση), την εκκίνηση εκτέλεσης της σχεδιασθείσης εργασίας και τον έλεγχο της παραγωγικής διαδικασίας (ανάλυση της κατάστασης εκτελούμενων εργασιών, επίσπευση καθυστερημένων και κρίσιμων εργασιών).

2.1 Η ΣΗΜΑΣΙΑ ΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Τα τελευταία χρόνια παρατηρείται στην βιομηχανία μια τάση για αύξηση της ποιότητας των προϊόντων, μείωση των χρόνων επεξεργασίας, μείωση των ενδιάμεσων αποθεμάτων και αύξηση της ευελιξίας των παραγωγικών συστημάτων. Η τάση αυτή έχει επηρεάσει διεθνώς τη σημασία του χρονοπρογραμματισμού. Μέχρι είκοσι ή τριάντα χρόνια παλιότερα, ο αυτόματος χρονοπρογραμματισμός για σύνθετα συστήματα παραγωγής ήταν μια καθαρά ακαδημαϊκή υπόθεση (Παππής Κ. , 2001).

Παλαιότερα, θεωρείτο λογικό να υπάρχουν αποθέματα ενδιάμεσων και τελικών προϊόντων για να αποσβένονται τα οποιοδήποτε λάθη στον χρονοπρογραμματισμό και για να αποζευγνύονται τα πολύπλοκα συστήματα. Με αυτό τον τρόπο ένας ειδικός - με βάση την πρακτική του εμπειρία – μπορούσε να διαχειρισθεί ικανοποιητικά τα προκύπτοντα απλουστευμένα μοντέλα. Σήμερα, είναι ευρέως αποδεκτό ότι αυτά τα ενδιάμεσα προϊόντα πρέπει προοδευτικά να περιορισθούν για μια σειρά από λόγους:

- Αυξανόμενη πολυπλοκότητα και γρήγορη παλαίωση των προϊόντων
- Περιορισμός των διάφορων ελαττωμάτων που παρουσιάζονται στα προϊόντα κατά την παραγωγική διαδικασία
- Οι πελάτες επιθυμούν συντομότερους χρόνους επεξεργασίας, με δυνατότητα αναθεώρησης της παραγγελίας
- Πιο ευέλικτη και άμεση αντίδραση στις επιθυμητές αλλαγές της παραγωγής καθώς και σε επείγοντα προβλήματα
- Πιο άμεση γνώση της παρέκκλισης της ποιότητας ενός προϊόντος από την επιθυμητή και παράλληλη γνώση της αιτίας αυτής της παρέκκλισης

Χωρίς την ύπαρξη αναλυτικής κατάστασης σχετικά με τις εργασίες που βρίσκονται σε εξέλιξη, ώστε το σύστημα να απλοποιείται, ο άνθρωπος πρέπει να προσπαθήσει ιδιαίτερα ώστε να προσαρμοσθεί άμεσα στο πολυπλοκότερο σύστημα. Στην καλύτερη περίπτωση αυτό οδηγεί σε μειωμένη ικανότητα διαχείρισης του συστήματος με μεγάλη λεπτομέρεια. Στην χειρότερη, η ευρεία γνώση του προβλήματος μπορεί να χαθεί τελείως. Η κατάσταση επιδεινώνεται από το γεγονός ότι το σύστημα από μόνο του αλλάζει με γρήγορους ρυθμούς και γίνεται πιο πολύπλοκο.

Καθίσταται επομένως κατανοητό πως στα σύγχρονα παραγωγικά συστήματα ο χρονοπρογραμματισμός της παραγωγής έχει τεράστια σημασία για την επιβίωση της επιχείρησης στις «άγριες» συνθήκες της αγοράς.

2.2 ΠΑΡΑΓΟΝΤΕΣ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΠΑΡΑΓΩΓΗΣ

Τα προβλήματα χρονοπρογραμματισμού συχνά περιπλέκονται ιδιαίτερα, λόγω μεγάλου αριθμού περιορισμών που συσχετίζουν τις διάφορες εργασίες μεταξύ τους ή τους πόρους με τις εργασίες ή τους πόρους και τις εργασίες με εξωγενείς παράγοντες. Για παράδειγμα, μπορεί να υπάρχουν *περιορισμοί προτεραιότητας (precedence constraints)* που συνδέουν τις εργασίες και καθορίζουν ποιες από αυτές πρέπει να προηγηθούν και για ποιο χρονικό διάστημα. Επίσης, δυο συγκεκριμένες εργασίες είναι δυνατό να συσχετίζονται και να είναι αδύνατο να χρησιμοποιούν δυο πόρους ταυτόχρονα. Ακόμα, ένας συγκεκριμένος πόρος μπορεί να μην είναι διαθέσιμος σε συγκεκριμένα χρονικά διαστήματα εξαιτίας προγραμματισμένης συντήρησης (Παππής Κ. , 2001).

Εφόσον τέτοιου είδους περιορισμοί δύναται να καθιστούν ιδιαίτερα δύσκολη την εύρεση λύσεων με ικανοποιητική ακρίβεια, είναι αναμενόμενο να οδηγείται η έρευνα στην κατεύθυνση επιλύσεως απλουστευμένων εκδόσεων των παραπάνω προβλημάτων, ώστε να προκύπτει μια αρχική εκτίμηση του συνολικού προβλήματος. Τότε, μπορεί κάνεις να δοκιμάσει πόσο ευαίσθητο είναι το σύστημα σε αυτή την πολυπλοκότητα και να προσπαθήσει να βρει προσεγγιστικές λύσεις σε προβλήματα που η πολυπλοκότητα αποδεικνύεται ιδιαίτερα σημαντική.

Το να βρεθεί μια καλή συνάρτηση ("στόχος") που θα πρέπει να μεγιστοποιηθεί ή να ελαχιστοποιηθεί μπορεί να είναι δύσκολο για ένα πρόβλημα χρονοπρογραμματισμού. Και τούτο διότι τέτοιοι σημαντικοί "στόχοι" - όπως π.χ. η ικανοποίηση του πελάτη - είναι δύσκολο να ποσοτικοποιηθούν και δεν εμφανίζονται στους λογιστικούς λογαριασμούς. Επιπροσθέτως, ένα σύστημα παραγωγής σχετίζεται συνήθως με τρεις τύπους στόχων οι οποίοι είναι αρκετά διαφορετικοί:

- Μεγιστοποίηση της παραγωγής σε κάποια συγκεκριμένη χρονική περίοδο
- Ικανοποίηση των επιθυμιών του πελάτη για ποιότητα και ανταπόκριση στις απαιτήσεις του
- Ελαχιστοποίηση στα κόστη απωλειών

Μεταξύ των πιθανών προσεγγίσεων συγκαταλέγονται:

- Επίλυση προβλημάτων με ένα συγκεκριμένο στόχο κάθε φορά
- Επίλυση συνδυασμών των διαφόρων στόχων μέσω καμπυλών
- Συνδυασμός στόχων αποδίδοντας κόστη στις επιθυμίες του πελάτη και στη μη χρήση πόρων

Σε ένα σύστημα με πολλούς πόρους, μια εργασία ή μια ομάδα εργασιών μπορούν να έχουν πολλές επιλογές σχετικά με το πού θα επεξεργαστούν. Οι διαφορετικές επιλογές για μια εργασία ονομάζονται δρομολόγια. Το να βρεθεί το καλύτερο δρομολόγιο ονομάζεται «*πρόβλημα δρομολόγησης*».

Στην πράξη υπάρχει και ένας σημαντικός αριθμός αποφάσεων που συνδέονται με το κλασικό πρόβλημα. Μια τέτοια απόφαση είναι η αλλαγή της ποσότητας ή της σύνθεσης των πόρων, ενώ το πρόβλημα του χρονοπρογραμματισμού βρίσκεται εν εξελίξει, με σκοπό να αντισταθμιστεί το μεταβλητό φορτίο στο σύστημα.

Για παράδειγμα, μια μηχανή μπορεί να είναι ικανή να επιταχύνει κατά τη διάρκεια μιας δύσκολης (από πλευράς αναγκών) περιόδου για να ανταποκριθεί στο αυξημένο φορτίο, με κόστος την αυξημένη φθορά της. Μια άλλη μηχανή είναι δυνατό να επανασυνδεθεί έτσι ώστε να αυξήσει την παραγωγική ικανότητα σε ένα άλλο τμήμα του συστήματος. Είναι πιθανό ακόμα να προστεθεί επιπλέον προσωπικό σε διάφορα τμήματα του συστήματος ή να εκμισθωθούν επιπλέον πόροι ή να μπει προσωρινά στην παραγωγή μια πεπαλαιωμένη γραμμή. Τέλος υπάρχει περίπτωση κάποιος πόρος να είναι διαμοιρασμένος μεταξύ δυο συστημάτων και να υπάρχει κάποιος τρόπος ώστε να αποφασίζεται σε ποιο σύστημα και για πόσο χρόνο θα διατίθεται ο πόρος αυτός. Συνοψίζουμε αυτούς τους τύπους των αποφάσεων με την έκφραση «*διαμόρφωση πόρων*».

Με ένα παρόμοιο τρόπο, μπορούν να διαμορφωθούν και οι διάφορες εργασίες. Για παράδειγμα, είναι δυνατό να υπάρχουν αρκετές επιλογές, από τεχνολογικής απόψεως, για να εκτελεστεί μια εργασία, ακόμα και σε βραχυπρόθεσμο επίπεδο, απαιτώντας διαφορετικά ποσά πόρων και συνεπώς περισσότερο ή λιγότερο χρόνο επεξεργασίας. Αυτή είναι μια *εσωτερική απόφαση* για τη διαμόρφωση του συστήματος. Μια *εξωτερική απόφαση* θα μπορούσε να διαπραγματεύεται με τους πελάτες τους χρόνους προθεσμίας και τις ποινές για τυχόν καθυστερήσεις.

2.3 ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΚΑΙ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Έως τώρα έγινε αναφορά στις σημαντικές αποφάσεις που αφορούν το κλασικό πρόβλημα του χρονοπρογραμματισμού, οι οποίες είναι τριών τύπων: *σειρά επεξεργασίας, χρονισμός* και *δρομολόγηση*. Αναφέρθηκαν επίσης και κάποιες αποφάσεις που σχετίζονται σε ποιο έμμεσο βαθμό, όπως η διαμόρφωση των πόρων ή των εργασιών. Πέρα όμως από αυτές τις αποφάσεις υπάρχει ένας μεγάλος αριθμός αποφάσεων σχετιζόμενος με τα πληροφοριακά συστήματα διοίκησης οι οποίες είναι αναγκαίο να παρθούν ώστε να υποστηρίξουν τις σημαντικές αποφάσεις. Αυτές περιλαμβάνουν έννοιες όπως *πρόβλεψη, κατηγοριοποίηση, ομαδοποίηση, συσσώρευση* και *διαχωρισμό* (Εργαζάκης Κ.Π. 2001).

Υπάρχουν τρεις τύποι *πρόβλεψης* για το πρόβλημα του χρονοπρογραμματισμού: *εσωτερικός, εξωτερικός και διανεμημένος*. Ένα τυπικό παράδειγμα *εξωτερικής* πρόβλεψης είναι η προσπάθεια εκτίμησης των απαιτήσεων των πελατών για τρεις μήνες. Μια τέτοια πρόβλεψη είναι ένα μίγμα των ήδη γνωστών παραγγελιών, των παραγγελιών σε διαπραγμάτευση και της στατιστικής εκτίμησης των παραγγελιών που είναι άγνωστες.

Οι *εσωτερικές* προβλέψεις αναπτύσσονται κατά τη διαδικασία εύρεσης λύσης για το πρόβλημα του χρονοπρογραμματισμού. Για παράδειγμα, το πρόγραμμα μπορεί να κάνει μια χονδρική εκτίμηση του χρόνου ολοκλήρωσης για διάφορες διεργασίες. Οι γνωστοί χρόνοι παράδοσης που έχουν συμφωνηθεί κατά την παραγγελία επιτρέπουν στο πρόγραμμα να εκτιμήσει ποιες διεργασίες είναι εκτός των χρόνων προθεσμίας και έτσι να δώσει μεγαλύτερη προτεραιότητα στη διεργασία με τον μικρότερο επιτρεπόμενο χρόνο ολοκλήρωσης. Πολλά είδη προγραμμάτων που προσπαθούν να επιλύσουν το πρόβλημα του χρονοπρογραμματισμού κάνουν εκτεταμένη χρήση τέτοιων προβλέψεων, λαμβάνοντας υπόψη τρέχοντες χρόνους προπορείας, τρέχοντα κόστη καθυστέρησης, τρέχουσα κρισιμότητα των πόρων, και μελλοντικούς χρόνους άφιξης.

Τα μεγάλα συστήματα χρονοπρογραμματισμού μπορεί να είναι *διανεμημένα*. Για παράδειγμα, μπορεί να υπάρχει ένα υποσύστημα για κάθε τμήμα στο σταθμό παραγωγής με ένα σύστημα σχεδιασμού να τα ελέγχει. Ένα άλλο υποσύστημα που βρίσκεται σε αντίθετη κατεύθυνση με τη ροή παραγωγής, μπορεί να παρέχει τους χρόνους άφιξης στο υπάρχον σύστημα, ενώ ένα άλλο υποσύστημα που βρίσκεται στην ίδια κατεύθυνση με τη ροή της παραγωγής, μπορεί να παρέχει τους αναγκαίους χρόνους προθεσμίας. Με όμοιο τρόπο, το σύστημα σχεδιασμού μπορεί να στέλνει μια συνολική εικόνα τις κρισιμότητας των πόρων, ενώ το υποσύστημα μπορεί να ανταπαντά με διορθώσεις, βασιζόμενο στην λεπτομερή γνώση της περιοχής του.

Το να τηρούνται στοιχεία σχετικά με το που βρίσκεται η παραγγελία ενός πελάτη στο σταθμό παραγωγής είναι επίσης αρκετά σημαντικό. Εντούτοις, αν ένα συγκεκριμένο έργο που βρίσκεται σε εξέλιξη έχει πολλαπλές τελικές χρήσεις τότε ανακύπτει το θέμα της δυναμικής *κατηγοριοποίησης*. Σε αυτή τη περίπτωση η κατηγοριοποίηση των διεργασιών που βρίσκονται εν εξελίξει δεν παραμένει αμετάβλητη, αλλά αλλάζει σύμφωνα με διάφορους παράγοντες όπως για παράδειγμα η προτεραιότητα κάθε μιας.

Αν το προς εξέταση μοντέλο υπαγορεύει πως μια ομάδα εργασιών πρέπει να επεξεργαστεί μαζί, αυτό είναι ένα λεπτομερειακό χρονικό πρόβλημα. Η *ομαδοποίηση* που χρησιμοποιείται σε αυτήν την περίπτωση είναι ένα είδος συσσώρευσης που χρησιμοποιείται ώστε για να διευκολυνθεί το πρόγραμμα που επιλύει το πρόβλημα του χρονοπρογραμματισμού από πλευράς υπολογισμών (παρόλο που έτσι προκύπτει λύση μειωμένης ακριβείας). Για παράδειγμα, αν κριθεί ότι μια ομάδα από εργασίες έχουν παρόμοια χαρακτηριστικά και συγκεντρωθούν μαζί σε όλη τη παραγωγική διαδικασία με σταθερή προτεραιότητα, τότε στη πραγματικότητα υπάρχει *συσσώρευση* αυτών των εργασιών σε μια ομάδα.

2.4 ΤΟ ΓΕΝΙΚΕΥΜΕΝΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Τα τελευταία περίπου 50 χρόνια έχει γίνει σημαντική θεωρητική έρευνα για την επίλυση του προβλήματος του χρονοπρογραμματισμού. Ο αριθμός και η ποικιλία των διαφορετικών μοντέλων που έχουν αναπτυχθεί είναι πράγματι μεγάλος. Είναι επόμενο να έχει δημιουργηθεί ένας συγκεκριμένος συμβολισμός που να αποδίδει με περιεκτικό τρόπο τη δομή των μοντέλων που έχουν αναπτυχθεί στη βιβλιογραφία. (Παππής Κ. , 2001)

Ο συμβολισμός αυτός έχει τρία πεδία και συμβολίζεται με α | β | γ. Το πεδίο α περιγράφει το περιβάλλον των μηχανών και έχει μια απλή είσοδο. Το πεδίο β παρέχει πληροφορίες σχετικά με τα χαρακτηριστικά της επεξεργασίας και τους περιορισμούς και μπορεί να μην έχει καθόλου εισόδους, να έχει μια είσοδο ή να έχει πολλαπλές εισόδους. Το πεδίο γ περιέχει την αντικειμενική συνάρτηση που πρέπει να ελαχιστοποιηθεί και συνήθως έχει μια είσοδο. Ωστόσο, πρέπει να σημειωθεί πως:

- αριθμός των εργασιών που πρόκειται να επεξεργαστούν καθώς και ο αριθμός των μηχανών θεωρείται σταθερός. Ο αριθμός των εργασιών συμβολίζεται με n, ενώ ο αριθμός των μηχανών συμβολίζεται με m. Γίνεται αναφορά σε μια συγκεκριμένη εργασία με τον δείκτη j ή i.
- Στην βιβλιογραφία συχνά απαντώνται διαφορετικοί συμβολισμοί που αναφέρονται στις ίδιες εισόδους.

Πιο συγκεκριμένα, το *πεδίο α* δύναται να λάβει τις ακόλουθες εισόδους:

Πίνακας 2. 1: Πιθανές είσοδοι του πεδίου α στο γενικό πρόβλημα Χρονοπρογραμματισμού

Πεδίο α	*Κατηγορία Παραγωγικού Συστήματος*
P_m	***Parallel machine shop #1:*** Πρόκειται για m όμοιες μηχανές που λειτουργούν παράλληλα. Μια εργασία j μπορεί να υποστεί επεξεργασία σε οποιαδήποτε μηχανή.
Q_m	***Parallel machine shop #2:*** Πρόκειται για m όμοιες μηχανές που λειτουργούν παράλληλα, αλλά σε αυτήν την περίπτωση έχουν διαφορετικές ταχύτητες.
R_m	***Parallel machine shop #3:*** Πρόκειται για γενίκευση του προηγουμένου με m διαφορετικές μηχανές που λειτουργούν παράλληλα.

F_m	***Flow shop:*** Πρόκειται για m μηχανές εν σειρά. Κάθε εργασία πρέπει να υποστεί επεξεργασία σε κάθε μια από τις m μηχανές. Όλες οι εργασίες έχουν την ίδια δρομολόγηση.
FF_s	***Flexible flow shop:*** Πρόκειται για γενίκευση του προηγουμένου. Αποτελείται από έναν αριθμό βαθμίδων εν σειρά με έναν αριθμό παραλλήλων μηχανών σε κάθε βαθμίδα.
J_m	***Job shop:*** Πρόκειται για m μηχανές. Στην προκειμένη περίπτωση κάθε εργασία ακολουθεί τη δική της ανεξάρτητη διαδρομή.
O_m	***Open shop:*** Κάθε εργασία πρέπει να υποστεί επεξεργασία από κάθε μηχανή m. Ωστόσο, κάποιοι από τους χρόνους επεξεργασίας μπορεί να είναι μηδενικοί. Διαφορετικές εργασίας μπορεί να ακολουθούν διαφορετικές διαδρομές.

Στο *πεδίο β* ορίζονται περιορισμοί που μπορεί να περιλαμβάνουν πολλαπλές εισόδους. Μερικές από αυτές είναι:

- **Release Dates (r_j)**. Ο χρόνος r_j μιας εργασίας j μπορεί να αναφέρεται σαν χρόνος ετοιμότητας. Είναι ο χρόνος κατά τον οποίο μια εργασία φτάνει στο σύστημα και εκφράζει το νωρίτερο δυνατό χρόνο στον οποίο μια εργασία j μπορεί να αρχίσει την επεξεργασία της.
- **Sequence-dependent setup times (s_{jk})**. Οι μηχανές συχνά πρέπει να καθαριστούν ή να προετοιμαστούν εκ νέου μεταξύ των εργασιών. Αυτή η διαδικασία είναι γνωστή ως setup. Αν ο χρόνος αυτής της διαδικασίας εξαρτάται από την εργασία που μόλις ολοκληρώθηκε και από αυτή που πρόκειται να επεξεργαστεί, τότε μιλάμε για χρόνους προετοιμασίας εξαρτώμενους από τη σειρά των εργασιών (sequence-dependent setup times).
- **Permutation (prmu)**. Ένας περιορισμός που μπορεί να υπάρχει σε ένα περιβάλλον flow shop, είναι ότι οι ουρές μπροστά από κάθε μηχανή λειτουργούν σύμφωνα με τη πειθαρχία τύπου FIFO. Αυτό συνεπάγεται ότι η σειρά (ή permutation) με την οποία οι εργασίες εισέρχονται στη πρώτη μηχανή, διατηρείται σε όλο το σύστημα.
- **Preemptions (prmp)**. Οι διακοπές (preemptions) υποδηλώνουν ότι δεν είναι αναγκαίο να διατηρείται μία εργασία σε μια μηχανή μέχρι την ολοκλήρωσή της. Επιτρέπεται να διακοπεί η επεξεργασία μιας εργασίας σε οποιαδήποτε χρονική στιγμή και να ξεκινήσει στη ίδια μηχανή η επεξεργασία μιας άλλης εργασίας. Το ποσό της επεξεργασίας που έχει υποστεί μια εργασία που έχει διακοπεί, δεν χάνεται. Όταν μια εργασία, που έχει διακοπεί, επιστρέφει πίσω

στη μηχανή (ή σε μια άλλη μηχανή, στην περίπτωση μηχανών συνδεδεμένων παράλληλα) χρειάζεται επεξεργασία για χρόνο ίσο με τον εναπομείναντα χρόνο επεξεργασίας της. Όταν επιτρέπονται τέτοιου είδους διακοπές, η είσοδος prmp εμφανίζεται στο πεδίο β. Αν δεν εμφανίζεται αυτή η είσοδος, τότε οι διακοπές δεν επιτρέπονται.

- **Precedence constraints (prec)**. Οι περιορισμοί προτεραιότητας μπορεί να εμφανίζονται σε απλές μηχανές ή σε περιβάλλοντα με μηχανές συνδεδεμένες παράλληλα και απαιτούν να έχουν ολοκληρωθεί μια ή περισσότερες εργασίες πριν επιτραπεί η έναρξη της επεξεργασίας μιας άλλης εργασίας. Υπάρχουν αρκετές μορφές περιορισμών προτεραιότητας. Αν κάθε εργασία έχει το πολύ ένα προκάτοχο και ένα διάδοχο, οι περιορισμοί αναφέρονται σαν αλυσίδες. Οι περιορισμοί μπορεί να δίνονται υπό μορφή δένδρου του τύπου intree ή outree. Στην περίπτωση που έχουμε δένδρο του τύπου intree (outree), όλα τα τόξα του γράφου κατευθύνονται προς (απομακρύνονται από) τη ρίζα του δένδρου. Αν ο όρος prec δεν εμφανίζεται σαν είσοδος στο πεδίο β τότε οι εργασίες δεν υπόκεινται σε περιορισμούς προτεραιότητας.
- **Blocking (block)**. Το μπλοκάρισμα είναι ένα φαινόμενο που μπορεί να εμφανίζεται σε συστήματα παραγωγής διαφόρων ειδών. Ένα σύστημα flow shop, για παράδειγμα, είναι δυνατό να έχει περιορισμένο αποθηκευτικό χώρο (buffer) μεταξύ δυο διαδοχικών μηχανών. Όταν αυτός ο ενδιάμεσος χώρος αποθήκευσης είναι κορεσμένος, η μηχανή που προηγείται αυτού του χώρου δεν μπορεί να προωθήσει μια εργασία που η επεξεργασία της έχει τελειώσει. Αυτό το φαινόμενο είναι γνωστό σαν μπλοκάρισμα. Η ολοκληρωμένη εργασία πρέπει να παραμείνει στην μηχανή, εμποδίζοντάς την να επεξεργαστεί μια άλλη εργασία. Το φαινόμενο εμφανίζεται συχνά στην περίπτωση μηδενικού αποθηκευτικό χώρου μεταξύ δυο διαδοχικών μηχανών.
- **Breakdowns (brkdwn)**. Οι βλάβες των μηχανών έχουν σαν αποτέλεσμα να μην είναι πάντα διαθέσιμες.
- **Recirculation (recrc)**. Η ανακυκλοφορία μπορεί να εμφανίζεται στα συστήματα τύπου job – shop, όπου μια εργασία μπορεί να επισκέπτεται μια μηχανή περισσότερες από μια φορές.
- **No-wait (nwt)**. Η απαίτηση no-wait είναι άλλο ένα φαινόμενο που μπορεί να συμβεί (κυρίως στα συστήματα flow shop). Δεν επιτρέπεται ανάμεσα σε δυο

διαδοχικές μηχανές να υπάρχει καθυστέρηση για τις εργασίες. Αυτό σημαίνει ότι ο χρόνος έναρξης της επεξεργασίας μιας εργασίας στη πρώτη μηχανή πρέπει να καθυστερήσει όσο πρέπει για να διασφαλιστεί ότι η εργασία μπορεί να περάσει διαμέσου του συστήματος χωρίς να χρειαστεί να περιμένει για κάποια μηχανή. Ένα παράδειγμα είναι ένα εργοστάσιο που κατασκευάζει ατσάλινα φύλλα ελάσματος. Σε αυτή την περίπτωση, ένα φύλλο ελάσματος δεν θα μπορούσε να περιμένει, γιατί θα κρύωνε. Είναι φανερό ότι αν ισχύει η απαίτηση no – wait, οι μηχανές λειτουργούν επίσης και κάτω από την πειθαρχία τύπου FIFO.

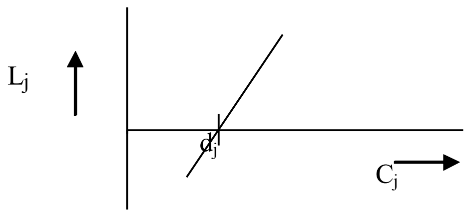
- **Reentrance**. Οι εργασίες επιστρέφουν στο σύστημα παραγωγής αρκετές φορές πριν ολοκληρωθούν πλήρως. Αυτή η πρακτική είναι πολύ κοινή στην βιομηχανία ημιαγωγών.
- **Machine eligibility constraints (M_j)**. Η είσοδος M_j (περιορισμοί καταλληλότητας των μηχανών) μπορεί να εμφανίζεται στο πεδίο β όταν το περιβάλλον που έχουμε είναι m μηχανές συνδεδεμένες παράλληλα (Pm). Όταν ο συμβολισμός M_j είναι παρόν, δεν είναι όλες οι m μηχανές ικανές για να επεξεργαστούν την εργασία j. Το σετ M_j είναι το σετ των μηχανών που μπορούν να επεξεργαστούν την εργασία j. Αν το πεδίο β δεν περιλαμβάνει το συμβολισμό M_j, τότε η εργασία j μπορεί να επεξεργαστεί σε οποιαδήποτε από τις m μηχανές.
- **Tooling constraints**. Οι μηχανές συχνά πρέπει να διαθέτουν ένα ή περισσότερα εργαλεία επεξεργασίας για να επεξεργαστούν τις εργασίες που πρέπει. Αυτά τα εργαλεία είναι διαφόρων τύπων και μερικά μπορεί να είναι σε περιορισμένη διαθεσιμότητα.

Οποιαδήποτε άλλη είσοδος εμφανίζεται στο πεδίο β είναι από μόνη της επεξηγηματική. Για παράδειγμα, η σχέση $p_j = p$ συνεπάγεται ότι όλοι οι χρόνοι επεξεργασίας είναι ίσοι και η σχέση $d_j = d$ σημαίνει ότι όλοι οι χρόνοι προθεσμίας είναι ίσοι. Οι χρόνοι προθεσμίας, σε αντίθεση με τους χρόνους απελευθέρωσης, συνήθως δεν δηλώνονται ρητά σε αυτό το πεδίο. Ο τύπος της αντικειμενικής συνάρτησης δίνει σαφή ένδειξη για το αν οι εργασίες έχουν χρόνο προθεσμίας ή όχι.

Το *πεδίο γ* περιέχει συνήθως μια είσοδο και παρέχει πληροφορίες σχετικά με το στόχο που είναι προς ελαχιστοποίηση. Ο αντικειμενικός στόχος που πρέπει να ελαχιστοποιηθεί είναι πάντα μια συνάρτηση των χρόνων ολοκλήρωσης των εργασιών, ο οποίος, βεβαίως, εξαρτάται από τον χρονοπρογραμματισμό. Ο χρόνος ολοκλήρωσης μια εργασίας j στην μηχανή i συμβολίζεται με C_{ij}. Ο χρόνος στον οποίο μια εργασία j βγαίνει από το σύστημα (δηλ. ο χρόνος ολοκλήρωσής της στην τελευταία μηχανή στην οποία χρειάζεται επεξεργασία) συμβολίζεται με C_j. Ο αντικειμενικός στόχος μπορεί επίσης να είναι συνάρτηση των χρόνων προθεσμίας. Η *καθυστέρηση (lateness)* μιας διεργασίας j καθορίζεται ως:

$$L_j = C_j - d_j$$

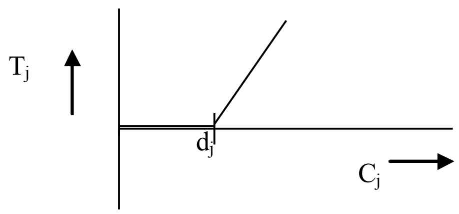
το οποίο είναι θετικό όταν η εργασία j έχει ολοκληρωθεί αργότερα και αρνητικό όταν έχει ολοκληρωθεί νωρίτερα. Η μορφή της συνάρτησης είναι η ακόλουθη:



Η *βραδύτητα (tardiness)* μιας εργασίας j καθορίζεται ως:

$$T_j = \max(C_j - d_j, 0) = \max(L_j, 0)$$

Η μορφή της συνάρτησης είναι η ακόλουθη:



Η διαφορά μεταξύ της βραδύτητας και της καθυστέρησης, έγκειται στο γεγονός ότι η βραδύτητα δεν είναι ποτέ αρνητική.

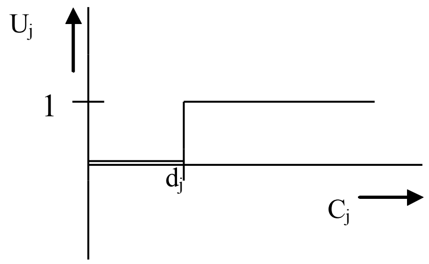
Αν μια εργασία ολοκληρωθεί πριν το χρόνο προθεσμίας της, τότε έχουμε ένα άλλο μέτρο (*earliness*) που δίνεται από τη σχέση:

$$E_j = \max(d_j - C_j, 0)$$

Η μοναδιαία ποινή μιας εργασίας, καθορίζεται ως:

$$U_j = \begin{cases} 1, \text{αν } C_j > d_j \\ 0, \text{διαφορετικά} \end{cases}$$

Η μορφή της συνάρτησης είναι η ακόλουθη:



Οι παραπάνω συναρτήσεις (καθυστέρηση, βραδύτητα και μοναδιαία ποινή) είναι οι συναρτήσεις που σχετίζονται με τους χρόνους προθεσμίας.

Τα ακόλουθα παραδείγματα, είναι πιθανές συναρτήσεις που ζητείται να ελαχιστοποιηθούν:

- **Makespan (C_{max}).** Το κριτήριο makespan, που ορίζεται ως max(C_1, . . ., C_n), είναι ισοδύναμο με το χρόνο ολοκλήρωσης της τελευταίας εργασίας που τελειώνει την επεξεργασία της στο σύστημα. Ένα ελάχιστο makespan συνήθως συνεπάγεται υψηλό ποσοστό χρησιμοποίησης των μηχανών.
- **Μέγιστη καθυστέρηση (Maximum lateness).** Η μέγιστη καθυστέρηση, L_{max}, καθορίζεται ως max(L_1, . . ., L_n). Μετράει την χειρότερη παραβίαση των χρόνων προθεσμίας.
- **Συνολικός σταθμισμένος χρόνος ολοκλήρωσης (Total weighted completion time – $\Sigma w_j C_j$).** Το άθροισμα των σταθμισμένων χρόνων ολοκλήρωσης των n εργασιών, δίνει μια ένδειξη του συνολικού κόστους παρακράτησης των μη ολοκληρωμένων εργασιών. Το άθροισμα των χρόνων ολοκλήρωσης, αναφέρεται συνήθως στη βιβλιογραφία σαν χρόνος ροής (flowtime). Ο συνολικός σταθμισμένος χρόνος ολοκλήρωσης αναφέρεται και ως σταθμισμένος χρόνος ροής.
- **Μειωμένος (discounted) συνολικός χρόνος ολοκλήρωσης ($\Sigma w_j(1 - e^{-rC_i})$).** Σε αυτή την περίπτωση έχουμε μια πιο γενική συνάρτηση κόστους από την προηγούμενη, όπου τα κόστη έχουν υποστεί μια μείωση με ρυθμό r, 0 < r < 1, ανά μονάδα χρόνου. Αυτό σημαίνει ότι αν η εργασία j δεν έχει ολοκληρωθεί τον χρόνο t, ένα επιπλέον κόστος $w_j r e^{-rt} dt$ προστίθεται, κατά τη χρονική περίοδο [t, t + dt]. Αν η διεργασία j ολοκληρώνεται τον χρόνο t, το συνολικό κόστος για την περίοδο [0, t] είναι $\Sigma w_j(1 - e^{-rt})$. Η τιμή του r κυμαίνεται συνήθως κοντά στο μηδέν, π.χ. 0,1 η 10%.

- **Συνολική σταθμισμένη βραδύτητα (total weighted tardiness – $\Sigma w_j T_j$).** Αυτή, είναι επίσης μια πιο γενική συνάρτηση κόστους από τον συνολικό σταθμισμένο χρόνο ολοκλήρωσης. Υπάρχει και ένα άλλο κριτήριο ($\Sigma w_j E_j$ – total weighted earliness) που χρησιμοποιείται στην περίπτωση που υπάρχει πρόστιμο για εργασίες που ολοκληρώνονται νωρίτερα από τους προκαθορισμένους χρόνους προθεσμίας.
- **Σταθμισμένος αριθμός των καθυστερημένων διεργασιών (weighted number of tardy jobs – ΣwjUj).** Ο σταθμισμένος αριθμός των καθυστερημένων εργασιών δεν είναι μόνο ένα μέτρο ακαδημαϊκού ενδιαφέροντος, αλλά συχνά αποτελεί αντικειμενικό στόχο στην πράξη, εφόσον είναι ένα μέτρο που μπορεί να καταγραφεί πολύ εύκολα.

Πρέπει να σημειωθεί πως πολλά μοντέλα χρονοπρογραμματισμού των πραγματικών προβλημάτων δεν μπορούν να περιγραφούν επαρκώς με βάση μόνο τα παραπάνω. Για παράδειγμα μπορεί να ορισθεί ένα σύστημα που θα είναι μίξη ενός job shop και ενός open shop. Τα δρομολόγια κάποιων εργασιών θα είναι σταθερά, ενώ τα δρομολόγια κάποιων άλλων εργασιών είναι (μερικώς) ανοικτά. Υπάρχει επίσης και ένα μίγμα παράλληλων μηχανών και συστημάτων flow shop. Αντί για ένα αριθμό μηχανών συνδεδεμένων παράλληλα υπάρχει ένας αριθμός συστημάτων flow shops συνδεδεμένων παράλληλα και μια εργασία πρέπει να περάσει μέσα από οποιοδήποτε από αυτά.

Παλιότερα η έρευνα επικεντρωνόταν περισσότερο σε μοντέλα με ένα απλό στόχο. Τελευταία, οι ερευνητές άρχισαν να μελετούν μοντέλα με πολλαπλούς στόχους (π.χ. με δυο κριτήρια – bicriteria objectives). Αρκετά άλλα χαρακτηριστικά του χρονοπρογραμματισμού έχουν μελετηθεί και αναλυθεί στην βιβλιογραφία. Τέτοια χαρακτηριστικά περιλαμβάνουν περιοδικό ή κυκλικό χρονοπρογραμματισμό, προσωπικό χρονοπρογραμματισμό, χρονοπρογραμματισμό περιορισμένων πόρων και άλλα.

Τέλος, κρίνεται σκόπιμο να αναφερθεί πως έχει γίνει σημαντική έρευνα για την εύρεση αλγορίθμων αλλά και άλλων ευρετικών μεθόδων που να δίνουν έναν βέλτιστο χρονοπρογραμματισμό σε πολυωνυμικό χρόνο. Εντούτοις, πολλά προβλήματα δεν μπορούν να επιλυθούν σε πολυωνυμικό χρόνο. Αυτά τα προβλήματα είναι γνωστά ως *NP – hard* (Παππής Κ. , 2001) (Εργαζάκης Κ.Π. , 2001)

2.5 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ JOB SHOP

Ένα από τα πιο δημοφιλή μοντέλα στην θεωρία του χρονοπρογραμματισμού είναι αυτό του χρονοπρογραμματισμού εργασιών συστημάτων job shop, αφού θεωρείται πως παρέχει μια ικανοποιητική αναπαράσταση του γενικότερου τομέα και έχει καταφέρει να "κερδίσει" μια ισχυρή φήμη όσον αφορά στην δυσκολία του να λυθεί. Είναι πιθανώς, το πιο μελετημένο και καλά θεμελιωμένο μοντέλο στην ντετερμινιστική θεωρία, το οποίο χρησιμεύει ως ένα συγκριτικό μοντέλο για διαφορετικές τεχνικές επίλυσης - καινούριες αλλά και παλιές - και από τη στιγμή που έχει πολλές πρακτικές εφαρμογές πραγματικά αξίζει να κατανοηθεί αλλά και να επιλυθεί (Jain A.S. Meeran , 1998).

Τυπικά το ντετερμινιστικό πρόβλημα του χρονοπρογραμματισμού εργασιών , το οποίο αναφέρεται συνήθως ως Π_j, αποτελείται από ένα πεπερασμένο σετ J, n εργασιών $\{ J_i \}$ οι οποίες πρέπει να υποστούν επεξεργασία σε ένα πεπερασμένο σετ M, m μηχανών $\{ M_k \}$.Κάθε εργασία J_i πρέπει να περάσει από κάθε μηχανή και να αποτελείται από μία αλυσίδα ή ένα συνδυασμό m_i διεργασιών $O_{i1},O_{i2},..,O_{im}$, που πρέπει να προγραμματιστούν να υποστούν επεξεργασία σε μία προκαθορισμένη δεδομένη σειρά, μία απαίτηση η οποία ονομάζεται *περιορισμός διαδοχής*. Υπάρχουν N διεργασίες στο σύνολο, όπου: $N = \sum_{i=1}^{n} m_i$.

Η O_{ik} είναι η διεργασία της εργασίας J_i η οποία πρέπει να επεξεργασθεί από τη μηχανή M_k για μία αδιάκοπη χρονική περίοδο $\tau_{ik.}$ Ακόμη, καμία εργασία δεν μπορεί να διακοπεί κατά την επεξεργασία της, δηλαδή να διακοπεί στη μέση της επεξεργασίας και να ολοκληρωθεί σε κάποια άλλη χρονική στιγμή. Κάθε εργασία έχει τη δική της διαδρομή στις μηχανές η οποία είναι ανεξάρτητη από τις άλλες εργασίες. Επιπλέον, το πρόβλημα πλαισιώνεται από περιορισμούς δυναμικότητας ή περιορισμούς διάζευξης οι οποίοι συνομολογούν πως κάθε μηχανή μπορεί να επεξεργάζεται μόνο μία διεργασία και κάθε διεργασία μπορεί να υπόκειται επεξεργασία μόνο από μία μηχανή κάθε φορά (Jain A.S. Meeran , 1998).

Αν ο χρόνος ολοκλήρωσης της διεργασίας J_i στη μηχανή M_k είναι Ci_k τότε η χρονική διάρκεια, στην οποία όλες οι διεργασίες για όλες τις εργασίες θα έχουν ολοκληρωθεί αναφέρεται ως *"makespan"* C_{max}, δηλαδή συνολικός χρόνος ολοκλήρωσης όλου του πλάνου εργασιών. Στην παραλλαγή βελτιστοποίησης του προβλήματος Π_j, ο σκοπός του προγραμματιστή είναι να καθορίσει τους χρόνους έναρξης επεξεργασίας για όλες τις διεργασίες όλων των εργασιών, $t_{ik} \geq 0$, έτσι ώστε να ελαχιστοποιηθεί ο συνολικός χρόνος μέσα στον οποίο όλο το πλάνο εργασιών θα έχει ολοκληρωθεί, ενώ παράλληλα θα ικανοποιούνται όλοι ο περιορισμοί διάζευξης αλλά και οι περιορισμοί δυναμικότητας.

Αυτός λοιπόν είναι ο σκοπός μας, ο οποίος μπορεί να δοθεί ως $\boldsymbol{C^*_{max}}$ ακολούθως:

$$C^*_{max} = min\ (C_{max}) = min\ (max\ (t_{ik} + \tau_{ik})\ :\ \forall\ Ji \in J, Mk \in M$$

Θα πρέπει επίσης να σημειωθεί πως οι διαστάσεις του προβλήματος Π_j, είναι $n\times m$ και ότι το N είναι $n\times m$ αρκεί το $m_i=m$ για κάθε εργασία $J_i \in J$ και ότι κάθε εργασία θα πρέπει να υπόκειται επεξεργασία ακριβώς μία φορά σε κάθε μηχανή. Σε μια πιο γενική κατάσταση του προβλήματος του χρονοπρογραμματισμού εργασιών οι επαναλήψεις των μηχανών (ή η έλλειψη μηχανών) επιτρέπονται μόνο στη δεδομένη σειρά των εργασιών $J_i \in J$ και για αυτό το mi θα πρέπει να είναι μεγαλύτερο ή μικρότερο αντίστοιχα από το m. Το κύριο ενδιαφέρον αυτής της έρευνας είναι η περίπτωση όπου m_i=m.

Οι μέθοδοι επίλυσης του προβλήματος που ήδη παρουσιάσαμε, δηλαδή του Π_j, μπορεί να είναι είτε μέθοδοι βελτιστοποίησης είτε μέθοδοι προσέγγισης, είτε δηλαδή είναι κατασκευαστικές μέθοδοι όπου «κατασκευάζουν» μια λύση για το πρόβλημα αξιοποιώντας τα δεδομένα του προβλήματος, είτε προσπαθούν να βρουν μία λύση με την τεχνική της διαρκούς αναδιάταξης της διαδοχής των εργασιών, προφανώς μέχρι την εύρεση της κατάλληλης σειράς επεξεργασίας.

Έτσι, υπάρχει πάντα η επιλογή για κάποιον που μελετά και επιθυμεί να επιλύσει το πρόβλημα να διαλέξει να εφαρμόσει μία προσεγγιστική μέθοδο η οποία μπορεί να δώσει μια «ικανοποιητική» λύση σε ένα αποδεκτό χρονικό διάστημα, είτε να προτιμήσει μία διαδικασία βελτιστοποίησης η οποία θα οδηγεί σε μία γενικώς βέλτιστη λύση, αλλά που απαιτεί σαφώς περισσότερο χρόνο υπολογισμού.

Η πλειοψηφία αυτών των μεθόδων, είτε πρόκειται για μεθόδους βελτιστοποίησης είτε για μεθόδους προσέγγισης, παρουσιάζουν το Π_j, χρησιμοποιώντας το διασπαστικό γραφικό μοντέλο (disjunctive graph model) ***G = {N , A , E}*** των Roy και Sussmann (1964) (Roy B.,Sussmann B. 1964). Για αυτό, προτού αναλυθούν οι διάφορες μέθοδοι επίλυσης του προβλήματος (στο επόμενο κεφάλαιο) κρίνεται σκόπιμο να δοθεί η μαθηματική του αναπαράσταση με βάση αυτό το μοντέλο :

Ελαχιστοποίηση του t* με το * $t \in N$ **με τους περιορισμούς** :

$ti - tj \geq \tau j$ **περιορισμός συνδετικός** $\forall \quad i, j \in N, (i, j) \in A$

$ti - tj \geq \tau j \vee tj - ti \geq \tau i$ **διαζευκτικός περιορισμός,** $\forall i, j \in N, i \neq j, (i, j) \in Ek, \forall k \in M$

$tj \geq 0$ **περιορισμός για τον νωρίτερο χρόνο έναρξης** $\forall k \in M$

ΚΕΦΑΛΑΙΟ 3

Μέθοδοι Επίλυσης Προβλημάτων Χρονοπρογραμματισμού

3. ΜΕΘΟΔΟΙ ΕΠΙΛΥΣΗΣ ΠΡΟΒΛΗΜΑΤΩΝ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Μέχρι στιγμής έχουν παρατεθεί γενικά στοιχεία για τα συστήματα παραγωγής καθώς και για το πρόβλημα του χρονοπρογραμματισμού. Ακολούθως εξετάζονται λεπτομερώς οι μαθηματικές προσεγγίσεις που χρησιμοποιούνται για την επίλυση του προβλήματος του χρονοπρογραμματισμού παραγωγής. Ανάλογα με τη φύση του συστήματος παραγωγής (περιβάλλον μηχανών, ιδιαίτερα χαρακτηριστικά, αντικειμενικοί στόχοι χρονοπρογραμματισμού) χρησιμοποιούνται διαφορετικές προσεγγίσεις που ανήκουν στις βασικές κατηγορίες:

- **Μέθοδοι Βελτιστοποίησης (Optimization Methods)**. Κάποια προβλήματα χρονοπρογραμματισμού είναι σχετικά εύκολα (από απόψεως απαιτούμενου υπολογιστικού χρόνου) στο να επιλυθούν μέσω καταλλήλων αλγορίθμων. Αυτοί οι αλγόριθμοι οδηγούν σε μια βέλτιστη λύση και είναι γνωστοί ως μέθοδοι βελτιστοποίησης.
- **Προσεγγιστικές Μέθοδοι (Approximation Methods)**. Πολλά προβλήματα χρονοπρογραμματισμού είναι από τη φύση τους πολύ δύσκολα ως προς την επίλυσή τους και ονομάζονται NP – hard. Δεν μπορούν να διαμορφωθούν σαν προβλήματα γραμμικού προγραμματισμού και επιπλέον δεν υπάρχουν απλοί κανόνες ή αλγόριθμοι που να παρέχουν μια βέλτιστη λύση λογικό υπολογιστικό χρόνο. Σε αυτήν την περίπτωση χρησιμοποιούνται οι προσεγγιστικές μέθοδοι, που εγγυώνται την εύρεση μιας βέλτιστης λύσης, αλλά αντίθετα προσφέρουν εφικτές (ως προς την υλοποίησή τους λύσεις) που δεν απέχουν ιδιαίτερα από τη βέλτιστη, σε σχετικά σύντομο χρόνο.

Στη συνέχεια του παρόντος κεφαλαίου θα αναπτυχθούν οι χρησιμοποιούμενες προσεγγίσεις και η κατηγοριοποίησή τους. (Brucker P , 2001)

3.1 ΜΕΘΟΔΟΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ

3.1.1 Αποδοτικές Μέθοδοι (efficient methods)

Ένας αποδοτικός αλγόριθμος λύνει ένα δεδομένο πρόβλημα βέλτιστα με απαίτηση που αυξάνει πολυωνυμικά με το μέγεθος των δεδομένων εισόδου. Αυτές οι μέθοδοι κατασκευάζουν απλά μία βέλτιστη λύση από τα δεδομένα του προβλήματος ακολουθώντας ένα δεδομένο και απλό σετ κανόνων, οι οποίοι καθορίζουν τη σειρά επεξεργασίας των διεργασιών.

3.1.2 Μαθηματικές Διατυπώσεις (mathematical formulations)

Διάφοροι ερευνητές έχουν συμφωνήσει πως τα προβλήματα χρονοπρογραμματισμού εργασιών μπορούν να λυθούν βέλτιστα χρησιμοποιώντας μεθόδους μαθηματικού προγραμματισμού. Μία από τις πιο γνωστές μεθόδους είναι η μέθοδος μεικτού ακέραιου γραμμικού προγραμματισμού *MIP (Mixed Integer Linear Programming).*

Μία ακόμη μέθοδος που ανήκει στις μεθόδους των μαθηματικών διατυπώσεων είναι η *Langrangian Relaxation (LR)*, η οποία μάλιστα θεωρείται και από τις πιο σημαντικές σε αυτή την κατηγορία μεθόδων επίλυσης. Πιο συγκεκριμένα, πρόκειται για μια μαθηματική προγραμματιστική τεχνική κατάλληλη για βελτιστοποίηση προβλημάτων με περιορισμούς (constrained optimization). Όμοια με τον μηχανισμό των τιμών σε μια αγορά, η μέθοδος αντικαθιστά τους "σκληρούς" περιορισμούς (π.χ. περιορισμούς στην διαθεσιμότητα των μηχανών) με την πληρωμή ορισμένων τιμών (δηλαδή των πολλαπλασιαστών Lagrange) βασιζόμενη στην "ζήτηση" για τη χρήση μιας μηχανής σε κάθε μονάδα χρόνου. Το αρχικό πρόβλημα μπορεί με αυτό τον τρόπο να αναλυθεί σε πολλά μικρότερα και ευκολότερα στην επίλυση υπο-προβλήματα. Τότε χρησιμοποιείται ο κατευθυνόμενος προς τα πίσω δυναμικός προγραμματισμός (backward dynamic programming) για την επίλυση αυτών των υπο-προβλημάτων, όποτε επιβάλλονται άλλοι περιορισμοί. Αφού αυτά τα υπο-προβλήματα επιλυθούν, οι πολλαπλασιαστές προσαρμόζονται βασιζόμενοι στο βαθμό παραβίασης των περιορισμών και ακολουθώντας ξανά τους μηχανισμούς της αγοράς. Τα υπο-προβλήματα τότε επιλύονται ξανά με βάση την καινούρια ομάδα πολλαπλασιαστών και η διαδικασία επαναλαμβάνεται. Με μαθηματικούς όρους, η "διπλή συνάρτηση" μεγιστοποιείται κατά την διαδικασία της αλλαγής της τιμής των πολλαπλασιαστών, όπου οι τιμές της διπλής συνάρτησης είναι τα κατώτατα όρια για το βέλτιστο εφικτό κόστος.

Όταν οι περιορισμοί χαλαρώνουν (relax) λόγω των πολλαπλασιαστών, οι λύσεις των ξεχωριστών υπο-προβλημάτων, όταν τοποθετούνται μαζί, μπορεί να μην συνιστούν έναν εφικτό χρονοπρογραμματισμό. Συνεπώς χρησιμοποιείται μια απλή ευρετική μέθοδος (θα αναλυθούν στη συνέχεια) προς το τέλος της διαδικασίας αλλαγής των πολλαπλασιαστών ώστε να παραχθούν εφικτοί χρονοπρογραμματισμοί που θα ικανοποιούν όλους τους περιορισμούς. Η ποιότητα όλων των εφικτών χρονοπρογραμματισμών μπορεί να εκτιμηθεί ποσοτικά συγκρίνοντας τα κόστη τους με το μεγαλύτερο κάτω όριο που μας δίνει η διπλή συνάρτηση (Brucker P , 2001) (Kaskavelis C., Caramanis M. , 1998) .

3.1.3 Τεχνικές Branch and Bound (Branch and Bound techniques)

Οι αλγόριθμοι *Branch and Bound* χρησιμοποιούν μία δυναμικά κατασκευασμένη δενδρική μορφή ως έναν τρόπο να αναπαραστήσουν τον χώρο λύσεων όλων των δυνατών αλληλουχιών των διεργασιών, όλων των δυνατών δηλαδή, λύσεων του προβλήματος. Η έρευνα ξεκινά από την κορυφή του δένδρου και μια ολοκληρωτική επιλογή έχει γίνει όταν έχουμε πια φτάσει σε ένα από τα χαμηλότερα επίπεδα του δένδρου. Κάθε κόμβος σε ένα επίπεδο *p* του δένδρου, αναπαριστά μια μερική διαδοχή *p* διεργασιών. Η συγκεκριμένη μέθοδος χρησιμοποιείται για την επίλυση συνδυαστικών προβλημάτων βελτιστοποίησης τα οποία είναι και αρκετά χρονοβόρα. Βασίζεται στην ιδέα της έξυπνης απαρίθμησης όλων των εφικτών λύσεων (Brucker P , 2001) .

Για να εξηγήσουμε καλύτερα τις λεπτομέρειες της μεθόδου θεωρούμε το εξής παράδειγμα: έστω ότι το διακριτό πρόβλημα βελτιστοποίησης P το οποίο πρέπει να λυθεί, είναι ένα πρόβλημα ελαχιστοποίησης. Το σύνολο S είναι το σύνολο των εφικτών λύσεων του P. Θεωρούμε επίσης μια υποομάδα προβλημάτων του P τα στοιχεία της οποίας καθορίζονται από την ομάδα S' που είναι ένα υποσύνολο του S. Είναι βολικό να προσδιορίζουμε το P καθώς και κάθε υποομάδα προβλημάτων με τα αντίστοιχα υποσύνολα $S' \subseteq S$. Τα βήματα που είναι αναγκαία για την υλοποίηση ενός αλγορίθμου branch and bound είναι τα ακόλουθα:

- **Branching:** Το S αντικαθίσταται από μικρότερα προβλήματα S_i (i = 1, . . ., r) τέτοια ώστε $\bigcup_{i=1}^{r} S_i = S$. Αυτή η διαδικασία καλείται *branching* και είναι μια διαδικασία περιοδικά επαναλαμβανόμενη. Έτσι, κάθε S_i είναι η βάση για μια επιπλέον διαδικασία branching. Η όλη διαδικασία αναπαρίσταται από ένα *δένδρο branching*. Το S είναι η ρίζα του δέντρου, τα S_i(i = 1, . . ., r) είναι τα παιδιά του S και ούτω καθεξής. Τα διακριτά προβλήματα βελτιστοποίησης που δημιουργούνται από την διαδικασία branching καλούνται *υπο-προβλήματα.*

- **Lower Bounding:** Υπάρχει αλγόριθμος για τον υπολογισμό ενός κάτω ορίου για τις αντικειμενικές τιμές όλων των εφικτών λύσεων ενός υπο-προβλήματος.
- **Upper Bounding:** Υπολογίζεται ένα άνω όριο U της αντικειμενικής τιμής του P. Η αντικειμενική τιμή οποιασδήποτε εφικτής λύσης θα δώσει ένα τέτοιο άνω όριο. Αν το κάτω όριο ενός υπο-προβλήματος είναι μεγαλύτερο ή ίσο με το U τότε αυτό το υπο-πρόβλημα δεν μπορεί να αποφέρει μια καλύτερη λύση για το P. Έτσι, δε χρειάζεται να συνεχισθούν οι διακλαδώσεις από τον αντίστοιχο κόμβο. Για να σταματήσει η διαδικασία branching θα πρέπει σε πολλούς κόμβους του δένδρου το όριο U να είναι κατά το δυνατό μικρότερο. Συνεπώς, στην αρχή του αλγόριθμου branch and bound εφαρμόζεται κάποια ευρετική μέθοδος ώστε να βρεθεί εφικτή λύση με μικρή τιμή U. Αφού εφαρμοσθεί αρκετές φορές την διαδικασία branching, είναι δυνατό να προσεγγισθεί μια κατάσταση στην οποία το υπο-πρόβλημα έχει μόνο μια εφικτή λύση. Τότε, το κάτω όριο LB του υπο-προβλήματος τίθεται ίσο με την αντικειμενική τιμή αυτής της λύσης και αντικαθίσταται το U με LB αν ισχύει LB < U.

Ένα μειονέκτημα της μεθόδου branch and bound είναι ότι μπορεί να είναι εξαιρετικά χρονοβόρα εφόσον ο αριθμός των κόμβων συχνά μπορεί να είναι πολύ μεγάλος (Brucker P , 2001) (Moursli O. , Pochet Y. 2000)

3.2 ΠΡΟΣΕΓΓΙΣΤΙΚΕΣ ΜΕΘΟΔΟΙ

Παρά το γεγονός οι προσεγγιστικές μέθοδοι δεν μπορούν να εξασφαλίσουν την επίτευξη ακριβών λύσεων, μπορούν να προσεγγίσουν ικανοποιητικά λύσεις που είναι πολύ κοντά στις βέλτιστες, με ένα μέτριο χρόνο υπολογισμού και για αυτό είναι πιο κατάλληλες για προβλήματα μεγαλύτερου μεγέθους.

Η σημασία των προσεγγιστικών μεθόδων επίλυσης των προβλημάτων χρονοπρογραμματισμού υποδεικνύεται από τους Glover και Greenberg (1989) που υποστηρίζουν ότι η κατευθυνόμενη αναζήτηση ενός «δένδρου» δεν είναι ουσιαστικά ικανοποιητική για συνδυαστικά και δύσκολα προβλήματα. Ισχυρίζονται, μάλιστα, πως οι ευρετικοί αλγόριθμοι, που είναι εμπνευσμένοι από φυσικά φαινόμενα, καθώς και οι έξυπνοι τρόποι επίλυσης είναι πιο κατάλληλοι για τέτοιου είδους προβλήματα, αφού παρέχουν μια κατάλληλη διμερή σύνδεση μεταξύ της έρευνας των διεργασιών και της τεχνητής νοημοσύνης.

Στη συνέχεια, θα παρουσιασθούν μερικές προσεγγιστικές μέθοδοι επίλυσης κάποιες από τις οποίες εφαρμόζονται στην παρούσα μελέτη.

3.2.1 Μέθοδος Beam Search

Η συγκεκριμένη μέθοδος προέρχεται από τη μέθοδο branch and bound και προσπαθεί να περιορίσει τις διακλαδώσεις με ένα έξυπνο τρόπο ώστε να μην είναι αναγκαίο να εξεταστούν όλα τα κλαδιά του δένδρου. Με αυτό τον τρόπο, απαιτεί λιγότερο υπολογιστικό χρόνο αλλά δεν μπορεί πλέον να εγγυηθεί μια βέλτιστη λύση.

Με τη μέθοδο branch and bound προσπαθούμε να εξαλείψουμε ένα κόμβο καθορίζοντας ένα κατώτατο όριο στις αντικειμενικές τιμές όλων των χρονοπρογραμματισμών που ανήκουν στον απόγονο αυτού του κόμβου. Αν η κατώτατη τιμή είναι μεγαλύτερη από την αντικειμενική τιμή ενός γνωστού χρονοπρογραμματισμού τότε ο κόμβος μπορεί να εξαλειφθεί και ο απόγονος του να παραβλεφθεί. Αν ήταν δυνατό να έχουμε ένα σχετικά καλό χρονοπρογραμματισμό, μέσω κάποιας έξυπνης ευρετικής μεθόδου, πριν αρχίσουμε την διαδικασία branch and bound, θα ήταν δυνατό να εξαλείψουμε πολλούς κόμβους. Εντούτοις, ακόμα και μετά από αυτές τις εξαλείψεις υπάρχουν ακόμα πολλοί κόμβοι που πρέπει να αποτιμηθούν. Το κύριο πλεονέκτημα της μεθόδου branch and bound είναι πως, αφού αποτιμηθούν όλοι οι κόμβοι, είναι σίγουρο με βεβαιότητα πως η τελική λύση θα είναι και η βέλτιστη.

Η μέθοδος beam search είναι μια προσαρμογή της μεθόδου branch and bound, στην οποία δεν αποτιμάται το σύνολο των κόμβων σε οποιοδήποτε δεδομένο επίπεδο. Μόνο οι κόμβοι με ικανοποιητικό εγγενές δυναμικό (promising nodes) στο επίπεδο k επιλέγονται ως κόμβοι από τους οποίους θα ξεκινούν παρακλάδια. Οι υπόλοιποι κόμβοι σε αυτό το επίπεδο απορρίπτονται μόνιμα. Ο αριθμός των κόμβων που διατηρούνται αποτελεί το beam width της αναζήτησης (εύρος της αναζήτησης). Είναι εμφανές πως η διαδικασία που καθορίζει ποιοι κόμβοι έχουν ικανοποιητικό εγγενές δυναμικό είναι μια κρίσιμη συνιστώσα αυτής της μεθόδου. Το να αποτιμήσουμε κάθε κόμβο προσεκτικά ώστε να αποκτήσουμε μια ακριβή εκτίμηση για το δυναμικό του απογόνου του, είναι μια χρονοβόρα διαδικασία.

Σε αυτό το σημείο, υπάρχει μια σχέση ανταλλαγής: μια χονδρική πρόβλεψη είναι γρήγορη αλλά μπορεί να οδηγήσει στην απόρριψη καλών λύσεων ενώ μια πιο λεπτομερής αποτίμηση μπορεί να είναι απαγορευτικά χρονοβόρα. Έτσι, χρησιμοποιείται μια προσέγγιση αποτελούμενη από τα εξής στάδια: γίνεται πρώτα μια χονδρική πρόβλεψη για όλους τους κόμβους που δημιουργούνται στο επίπεδο k. Βασιζόμενοι στα αποτελέσματα αυτών των χοντρικών προβλέψεων, επιλέγουμε ένα αριθμό κόμβων για μια πιο λεπτομερή αποτίμηση ενώ απορρίπτουμε (filtering out) τους υπόλοιπους κόμβους. Ο αριθμός των κόμβων που επιλέγονται για πιο λεπτομερειακή αποτίμηση αποτελεί το filter width (πλάτος φίλτρου). Μετά από εξέταση όλων των κόμβων που περνούν το φίλτρο, επιλέγουμε ένα υποσύνολο αυτών των κόμβων. Ο αριθμός των κόμβων σε αυτό το υποσύνολο είναι ίσος με το beam width, το οποίο εξάλλου πρέπει να είναι μικρότερο από ότι το filter width. Από αυτό το υποσύνολο παράγονται οι διακλαδώσεις προς το επόμενο επίπεδο.

Ένα απλό παράδειγμα μιας χοντρικής πρόβλεψης είναι η εξής: υπολογίζουμε τη συνεισφορά του μερικού χρονοπρογραμματισμού στον αντικειμενικό στόχο και στους χρόνους προθεσμίας των διεργασιών που είναι απρογραμμάτιστες. Με βάση αυτές τις τιμές, οι κόμβοι σε ένα συγκεκριμένο επίπεδο μπορούν να συγκριθούν με αυτούς ενός άλλου επιπέδου και είναι δυνατό να γίνει μια συνολική αξιολόγηση.

Κάθε φορά που ένας κόμβος πρέπει να υποβληθεί σε λεπτομερή αποτίμηση, όλες οι διεργασίες που δεν έχουν ακόμα προγραμματιστεί, προγραμματίζονται σύμφωνα με ένα σύνθετο κανόνα απόδοσης προτεραιότητας (dispatching rule). Ένας τέτοιος χρονοπρογραμματισμός μπορεί να παραχθεί γρήγορα, εφόσον απαιτεί μόνο ταξινόμηση. Το αποτέλεσμα του είναι μια ένδειξη του εγγενούς δυναμικού του κόμβου. Αν ως αποτέλεσμα έχουμε ένα μεγάλο αριθμό διεργασιών, οι κόμβοι μπορεί να φιλτραριστούν εξετάζοντας πρώτα ένα μερικό χρονοπρογραμματισμό ο οποίος λαμβάνεται προγραμματίζοντας μόνο ένα υποσύνολο των διεργασιών που απομένουν με βάση ένα dispatching rule. Αυτός ο διευρυμένος μερικός χρονοπρογραμματισμός μπορεί να αποτιμηθεί και κατόπιν, βασιζόμενοι στην τιμή του, μπορούμε να αποφασίσουμε αν ένας κόμβος θα απορριφθεί ή όχι. Ένας κόμβος που διατηρείται μπορεί να αναλυθεί πιο λεπτομερειακά με το να προγραμματίζουμε όλες τις διεργασίες του που απομένουν, με τον σύνθετο dispatching rule. Η τιμή του αντικειμενικού στόχου αυτού του χρονοπρογραμματισμού αναπαριστά τότε ένα άνω όριο για τον καλύτερο χρονοπρογραμματισμό ανάμεσα στους απόγονους αυτού του κόμβου (Brucker P , 2001) , (Jayamohan M. S. , Rajendran C. 2000) .

3.2.2 Ευρετικές Μέθοδοι (Heuristics)

Στα τέλη της 1980, έγινε ακόμα πιο κατανοητό πως η χρήση των μεθόδων βελτιστοποίησης απαιτούσε την επίλυση προβλημάτων με ιδιαίτερα αυξημένη πολυπλοκότητα και έτσι η έρευνα επικεντρώθηκε ακόμα περισσότερο στην επίλυση των προβλημάτων με τη βοήθεια προσεγγιστικών μεθόδων. Η χρήση των dispatching rules ήταν το πρώτο στάδιο αλλά αργότερα υπήρξε αυξανόμενη ανάγκη για πιο κατάλληλες τεχνικές οι οποίες θα εφάρμοζαν πιο εμπλουτισμένες μεθόδους. Αυτές είναι γνωστές ως *ευρετικές μέθοδοι* (Brucker P , 2001) .

Μια ευρετική διαδικασία αναζήτησης (heuristic search procedure) μπορεί να προκύψει από τον συνδυασμό ευρετικών μεθόδων με αλγόριθμους αναζήτησης. Ο αλγόριθμος αναζήτησης, αναμφίβολα καθορίζει ένα διάστημα μερικών λύσεων του προβλήματος και το εξερευνά ωσότου βρεθεί μια αποδεκτή ή βέλτιστη λύση. Ο μηχανισμός που χρησιμοποιείται για να εκτελεστεί η αναζήτηση είναι επαναληπτικός και αρκετά απλός. Ο αλγόριθμος αρχίζει από μια προφανή ή από μια ήδη γνωστή μερική λύση και καθορίζει τις αλλαγές που μπορούν να γίνουν σε αυτή τη μερική λύση. Η ευρετική γνώση χρησιμοποιείται για να αποτιμηθούν οι πιθανές αλλαγές καθώς και η ποιότητα των μερικών λύσεων που προκύπτουν. Αυτή η γνώση βοηθά κατόπιν στην επιλογή εκ των μερικών λύσεων που προκύπτουν αυτών που αξίζουν να τροποποιηθούν διαδοχικά. Σε κάθε βήμα της διαδικασίας επίλυσης του προβλήματος, ο αλγόριθμος ενημερώνει πολλά μέρη της πληροφορίας στην οποία η ευρετική μέθοδος αναφέρεται και αλλάζει την πορεία της αναζήτησης, αποκλείοντας τις μερικές λύσεις από τις οποίες δεν είναι δυνατό να εξασφαλιστεί ολοκληρωμένη λύση ή μη ολοκληρωμένη λύση καλύτερη από ήδη γνωστές λύσεις (Rajendran C.,Ziegler H. 1999) .

Η χρήση των ευρετικών μεθόδων βοήθησε ιδιαίτερα στην επίλυση προβλημάτων που είχαν απασχολήσει για καιρό τους ερευνητές όπως το πρόβλημα FT 10 (που το έθεσαν οι Fisher & Thompson 1963), ένα πρόβλημα που παρέμενε άλυτο επί 25 έτη.

Μια γνωστή ευρετική μέθοδος (η πρώτη που έδωσε λύση στο πρόβλημα FT 10 και έπαιξε καθοριστικό ρόλο στην ανάπτυξη των προσεγγιστικών μεθόδων) είναι η *shifting bottleneck procedure (SBP)* (Adams, Balas & Zawack, 1988). Ένας από τους λόγους που αυτός ο κατασκευαστικός αλγόριθμος επιτυγχάνει καλά αποτελέσματα είναι οι καλά αναπτυγμένοι αλγόριθμοι και προγραμματιστικές τεχνικές που υπάρχουν για το πρόβλημα του χρονοπρογραμματισμού της μιας μηχανής (one machine scheduling problem). Η μέθοδος SBP χαρακτηρίζεται από τα ακόλουθα βήματα: αναγνώριση υπο-προβλήματος, επιλογή σημείων συμφόρησης (bottleneck selection), επίλυση υπο-προβλήματος και προγραμματισμός επαναβελτιστοποίησης (Demirkol et. Al., 1997). Η σημερινή στρατηγική συνίσταται στην αναγωγή του προβλήματος σε m προβλήματα με μια απλή μηχανή και στην επίλυση κάθε υπο-προβλήματος ένα κάθε φορά. Κάθε λύση που προκύπτει συγκρίνεται με όλες τις άλλες και οι μηχανές κατατάσσονται με βάση τη λύση τους. Η μηχανή που έχει το μεγαλύτερο ελάχιστο όριο, προσδιορίζεται ως η μηχανή συμφόρησης (bottleneck machine). Η μέθοδος SBP διατάσσει πρώτα τις μηχανές συμφόρησης, με τις εναπομείναντες μη διατεταγμένες μηχανές να αγνοούνται και τις μηχανές που ήδη έχουν διαταχθεί να παραμένουν ως έχουν. Κάθε φορά που διατάσσεται μια μηχανή συμφόρησης, κάθε μηχανή που έχει διαταχθεί πιο πριν και που επιδέχεται βελτίωση, βελτιστοποιείται ξανά σε τοπικό επίπεδο επιλύνοντας ξανά το πρόβλημα της μιας μηχανής. Το πρόβλημα της μιας μηχανής λύνεται επαναληπτικά, χρησιμοποιώντας τη προσέγγιση του Carlier (1982) που παρέχει μια ακριβή και γρήγορη λύση.

Ένα ουσιώδες πρόβλημα με τη παραπάνω μέθοδο είναι η δυσκολία στην εκτέλεση της επαναβελτιστοποίησης και το ότι μπορεί να δώσει μη πρακτικές λύσεις. Εντούτοις, η μέθοδος SBP χρησιμοποιήθηκε ιδιαίτερα και μάλιστα υπήρξαν και επεκτάσεις της (Lambrecht & Ivens, 1996) ώστε να αντιμετωπίζει μια ποικιλία παραμέτρων όπως χρόνοι προετοιμασίας των μηχανών και περιβάλλοντα με παράλληλες μηχανές. Η πιο πρόσφατη γενίκευση έχει προέλθει από τον Balas et. al. (1998) που συνδυάζει την ακριβή προσεγγιστική μέθοδο για τη μια μηχανή με την καθοδηγούμενη μέθοδο τοπικής αναζήτησης (local search) και εφαρμόζει αυτό στο πρόβλημα job shop με χρόνους προθεσμίας.

3.2.3 Μέθοδοι Τοπικής Αναζήτησης (Local Search Methods)

Οι ευρετικές μέθοδοι χαρακτηρίζονται ως *κατασκευαστικές*, καθώς ξεκινούν χωρίς κάποιο χρονοπρογραμματισμό και σταδιακά κατασκευάζουν ένα χρονοπρογραμματισμό προσθέτοντας μια διεργασία κάθε φορά.

Σε αυτή την ενότητα περιγράφονται αλγόριθμους που είναι *βελτιωτικοί*. Η ιδέα στην οποία στηρίζονται είναι τελείως διαφορετική από τους κατασκευαστικούς αλγόριθμους. Οι βελτιωτικοί αλγόριθμοι ξεκινούν με ένα ολοκληρωμένο χρονοπρογραμματισμό που μπορεί να επιλεχθεί αυθαίρετα και προσπαθούν να δημιουργήσουν ένα καλύτερο χρονοπρογραμματισμό μετατρέποντας τον ήδη υπάρχοντα. Μια σημαντική κατηγορία των βελτιωτικών αλγόριθμων είναι οι *μέθοδοι τοπικής αναζήτησης*. Μια τέτοια διαδικασία που βασίζεται σε τοπική αναζήτηση, σε αντίθεση με μια γενική διαδικασία αναζήτησης (global search procedure) δεν εγγυάται βέλτιστη λύση. Επιχειρεί να βρει μια καλύτερη λύση από την ήδη υπάρχουσα, μέσα από αναζήτηση στην *γειτονική περιοχή* (*neighborhood*) του υπάρχοντος χρονοπρογραμματισμού. Δυο χρονοπρογραμματισμοί λέμε ότι γειτνιάζουν αν ο ένας μπορεί να ληφθεί διαμέσου καλά ορισμένης τροποποίησης του άλλου (Brucker P , 2001) .

Σε κάθε επανάληψη, μια μέθοδος τοπικής αναζήτησης επιτελεί μια αναζήτηση στη γειτονική περιοχή της λύσης και αξιολογεί τις γειτονικές λύσεις. Η διαδικασία είτε δέχεται είτε απορρίπτει μια υποψήφια λύση σύμφωνα με ένα δεδομένο κριτήριο αποδοχής – απόρριψης. Κάποιος μπορεί να συγκρίνει τις διάφορες μεθόδους τοπικής αναζήτησης σύμφωνα με τα ακόλουθα σχεδιαστικά κριτήρια:

- Την αναπαράσταση του χρονοπρογραμματισμού που χρειάζεται για την διαδικασία
- Τον σχεδιασμό της γειτονικής περιοχής
- Την διαδικασία αναζήτησης στην γειτονική περιοχή
- Το κριτήριο αποδοχής – απόρριψης

Ένας χρονοπρογραμματισμός που αφορά μια απλή μηχανή χωρίς διακοπές (non – preemptive) μπορεί να καθοριστεί από τον απλό συνδυασμό (permutation) n διεργασιών. Ένας χρονοπρογραμματισμός για ένα πρόβλημα του τύπου job shop χωρίς διακοπές, μπορεί να καθοριστεί από m διαδοχικούς αλφαριθμητικούς χαρακτήρες όπου κάθε ένας αναπαριστά ένα συνδυασμό n επιμέρους εργασιών σε μια συγκεκριμένη μηχανή. Με βάση αυτές τις πληροφορίες, είναι δυνατό να υπολογιστούν οι χρόνοι έναρξης και ολοκλήρωσης όλων των επιμέρους εργασιών. Εντούτοις, όταν επιτρέπονται οι διακοπές, η μορφή της αναπαράστασης του χρονοπρογραμματισμού γίνεται σημαντικά πιο περίπλοκη.

Η δομή της γειτονικής περιοχής είναι ένα πολύ σημαντικό μέρος μιας μεθόδου τοπικής αναζήτησης. Για μια απλή μηχανή, η γειτονική περιοχή ενός συγκεκριμένου χρονοπρογραμματισμού μπορεί να καθοριστεί απλά ως το σύνολο των χρονοπρογραμματισμών που μπορούν να ληφθούν αφού πρώτα εφαρμοστεί μια απλή ανταλλαγή μεταξύ γειτονικών ζευγαριών. Ο ορισμός αυτός συνεπάγεται πως υπάρχουν n – 1 χρονοπρογραμματισμοί στην γειτονική περιοχή ενός αρχικού χρονοπρογραμματισμού. Μια μεγαλύτερη γειτονική περιοχή για ένα χρονοπρογραμματισμό μιας απλής μηχανής μπορεί να οριστεί με το να ληφθεί αυθαίρετα μια διεργασία από τον χρονοπρογραμματισμό και να τοποθετηθεί σε άλλη θέση. Εμφανώς, κάθε διεργασία μπορεί να τοποθετηθεί σε n – 1 άλλες θέσεις. Η συνολική γειτονική περιοχή περιέχει λιγότερες από n(n – 1) επιμέρους γειτονικές περιοχές γιατί κάποιες από αυτές είναι όμοιες. Η γειτονική περιοχή ενός χρονοπρογραμματισμού σε ένα πιο πολύπλοκο περιβάλλον μηχανών είναι συνήθως πιο περίπλοκη.

Η διαδικασία αναζήτησης μέσα σε μια γειτονική περιοχή μπορεί να γίνει με διαφορετικούς τρόπους. Ένας απλός τρόπος είναι να επιλέξουμε τυχαία κάποιους χρονοπρογραμματισμούς στη γειτονική περιοχή, να τους αποτιμήσουμε και να αποφασίσουμε ποιον θα αποδεχτούμε. Μπορούμε επίσης να εφαρμόσουμε μια πιο οργανωμένη αναζήτηση και να επιλέξουμε κατόπιν τους χρονοπρογραμματισμούς που εμφανίζονται ελπιδοφόροι. Κάποιος θα μπορούσε να αντιμεταθέσει τις διεργασίες που επηρεάζουν περισσότερο τον αντικειμενικό στόχο. Για παράδειγμα, όταν η συνολική σταθμισμένη βραδύτητα (total weighted tardiness) πρέπει να ελαχιστοποιηθεί, κάποιος θα μπορούσε να μετακινήσει τις διεργασίες που είναι πολύ καθυστερημένες προς την αρχή του χρονοπρογραμματισμού.

Τέλος, το κριτήριο αποδοχής – απόρριψης είναι συνήθως το σχεδιαστικό μέρος που διαφοροποιεί σημαντικά μια μέθοδο τοπικής αναζήτησης. Για παράδειγμα, η διαφορά ανάμεσα σε δυο γνωστές μεθόδους τοπικής αναζήτησης (simulated annealing and tabu – search) έγκειται κυρίως στην διαφοροποίηση των κριτηρίων αυτών. Στην μεν μέθοδο simulated annealing, το κριτήριο αποδοχής – απόρριψης βασίζεται σε στοχαστική διαδικασία, ενώ στη δε μέθοδο tabu – search το κριτήριο βασίζεται σε ντετερμινιστική διαδικασία (Brucker P , 2001) .

- **Simulated Annealing**

Πρόκειται για μια μέθοδο αναζήτησης που έχει την καταγωγή της στα πεδία της φυσικής και της επιστήμης των υλικών. Αναπτύχθηκε αρχικά ως μοντέλο προσομοίωσης για την περιγραφή της σκλήρυνσης συμπυκνωμένου μετάλλου με πυράκτωση.

Η διαδικασία simulated annealing εφαρμόζει μια σειρά επαναλήψεων. Ο αλγόριθμος, στην αναζήτησή του για ένα βέλτιστο χρονοπρογραμματισμό, μετακινείται από τον ένα χρονοπρογραμματισμό στον άλλο. Επιτρέπονται μετακινήσεις προς χειρότερες λύσεις (ανηφορικές – uphill μετακινήσεις), καθώς δίδεται στη διαδικασία η δυνατότητα να μετακινηθεί μακριά από ένα τοπικό ελάχιστο και να βρει μια καλύτερη λύση αργότερα. Στην πράξη, πολλά κριτήρια τερματισμού χρησιμοποιούνται για αυτή τη διαδικασία. Ένας τρόπος είναι να αφήσουμε τη διαδικασία να εκτελεστεί για ένα προκαθορισμένο αριθμό επαναλήψεων. Άλλος τρόπος είναι να αφήσουμε τη διαδικασία να εκτελείται μέχρι ωσότου δεν έχουμε βελτίωση για ένα δεδομένο αριθμό επαναλήψεων.

Η δραστικότητα της μεθόδου εξαρτάται από το σχεδιασμό της γειτονικής περιοχής καθώς και από το πώς η αναζήτηση διεξάγεται μέσα σε αυτή τη περιοχή. Αν η περιοχή είναι σχεδιασμένη με τρόπο που διευκολύνει μετακινήσεις προς καλύτερες λύσεις και μετακινήσεις από τοπικά ελάχιστα, τότε η διαδικασία θα αποδώσει σωστά. Η αναζήτηση σε μια γειτονική περιοχή μπορεί να γίνει τυχαία ή με ένα οργανωμένο τρόπο. Για παράδειγμα, μπορεί να υπολογιστεί η συνεισφορά κάθε διεργασίας στην αντικειμενική συνάρτηση και η διεργασία με το μεγαλύτερο αντίκτυπο στην αντικειμενική συνάρτηση μπορεί να επιλεχθεί σαν υποψήφια για αμοιβαία ανταλλαγή(McMullen P. , Frazier G. 2000) .

- **Tabu Search**

Μοιάζει σε αρκετά σημεία με τη μέθοδο simulated annealing. Μετακινούνται και οι δυο από τον ένα χρονοπρογραμματισμό στον άλλο, με τον επόμενο χρονοπρογραμματισμό να είναι πιθανώς χειρότερος από ότι ο προηγούμενος. Για κάθε αναζήτηση με τη μέθοδο tabu search ορίζεται μια γειτονική περιοχή, όπως και στη μέθοδο simulated annealing. Η αναζήτηση ενός χρονοπρογραμματισμού στην γειτονική περιοχή ως ένα ενδεχόμενο υποψήφιο για μετακίνηση σε αυτόν, είναι ξανά ένα σχεδιαστικό ζήτημα. Όπως και στη μέθοδο simulated annealing, η αναζήτηση μπορεί να γίνει με τυχαίο ή με οργανωμένο τρόπο. Η βασική διαφορά μεταξύ των δυο μεθόδων έγκειται στον μηχανισμό που χρησιμοποιείται για να εγκριθεί ένας υποψήφιος χρονοπρογραμματισμό: Στη μέθοδο tabu search ο μηχανισμός δεν είναι στοχαστικός, αλλά ντετερμινιστικής φύσης.

Σε οποιοδήποτε στάδιο της διαδικασίας κρατείται μια tabu list των μεταβολών (mutations) που η διεργασία δεν επιτρέπεται να εκτελέσει. Οι μεταβολές αυτές μπορεί να είναι για παράδειγμα ζεύγη διεργασιών που δεν μπορούν να υποστούν αμοιβαία ανταλλαγή. Η tabu list έχει ένα σταθερό αριθμό εισόδων (συνήθως μεταξύ πέντε και εννέα) ο οποίος εξαρτάται από την εφαρμογή. Κάθε φορά που συμβαίνει μετακίνηση εξαιτίας μιας μεταβολής στον τρέχοντα χρονοπρογραμματισμό, η ανάποδη (*reverse*) μεταβολή εισάγεται στην αρχή της tabu list. Όλες οι άλλες είσοδοι μετακινούνται προς τα κάτω κατά μια θέση ενώ η τελευταία είσοδος διαγράφεται. Η ανάποδη μεταβολή εισάγεται στην tabu list για να αποφευχθεί η επιστροφή σε ένα τοπικό ελάχιστο που είχε επισκεφτεί πριν.

Στην πράξη, κατά διαστήματα, μια ανάποδη μεταβολή που είναι απαγορευμένη (tabu) θα μπορούσε να είχε οδηγήσει σε ένα νέο χρονοπρογραμματισμό τον οποίο δεν είχαμε επισκεφτεί πριν, με αντικειμενική τιμή μικρότερη από οποιαδήποτε έχει ληφθεί ως εκείνη τη στιγμή. Αυτό μπορεί να συμβαίνει όταν η μεταβολή είναι κοντά στο κάτω άκρο της tabu list και ένας αριθμός μετακινήσεων έχει ήδη γίνει από τότε που η μεταβολή έχει εισαχθεί στη λίστα. Έτσι, αν ο αριθμός των εισόδων στην tabu list είναι πολύ μικρός, μπορεί να έχουμε την εμφάνιση κυκλικής μεταβολής (cycling). Αν ο αριθμός των εισόδων είναι πολύ μεγάλος τότε η αναζήτηση μπορεί να παρεμποδιστεί αδικαιολόγητα.

Οι πληροφορίες που μεταφέρονται κατά τη διάρκεια της εκτέλεσης της μεθόδου tabu search, συνίστανται στη tabu list καθώς και στην βέλτιστη λύση που λαμβάνεται μέσω της διαδικασίας αναζήτησης. Πρόσφατα προτάθηκαν πιο ισχυρές εκδόσεις της μεθόδου tabu search. Αυτές οι εκδόσεις συγκρατούν περισσότερες πληροφορίες. Μια από αυτές χρησιμοποιεί το επονομαζόμενο tabu tree. Σε αυτό το δέντρο, κάθε κόμβος αναπαριστά μια λύση ή ένα χρονοπρογραμματισμό. Καθώς η διαδικασία αναζήτησης πηγαίνει από τη μια λύση στην άλλη (με κάθε λύση να έχει μια tabu list) η διαδικασία δημιουργεί επιπρόσθετους κόμβους. Ορισμένες λύσεις που φαίνονται ότι μπορεί να είναι βέλτιστες (ελπιδοφόρες) μπορεί να μην χρησιμοποιηθούν σαν σημείο εκκίνησης άμεσα, αλλά παρακρατούνται για μελλοντική χρήση. Αν σε ένα συγκεκριμένο σημείο κατά τη διάρκεια της διαδικασίας αναζήτησης, η παρούσα λύση δεν φαίνεται ελπιδοφόρα σαν σημείο εκκίνησης, η διαδικασία αναζήτησης μπορεί να επιστρέψει μέσα στο tabu tree σε άλλο κόμβο (λύση) που είχε παρακρατηθεί πριν και να ξεκινήσει ξανά, αλλά τώρα προς διαφορετική κατεύθυνση (Brucker P , 2001) , (Armentano V. , Schrich C. 2000) .

- **Γενετικοί Αλγόριθμοι (Genetic Algorithms)**

Η μέθοδος που χρησιμοποιεί γενετικούς αλγόριθμους είναι πιο γενική και αφηρημένη από τις μεθόδους simulated annealing και tabu search. Οι γενετικοί αλγόριθμοι, όταν εφαρμόζονται στον χρονοπρογραμματισμό, θεωρούν τις ακολουθίες ή τους διάφορους χρονοπρογραμματισμούς ως *άτομα (individuals)* ή μέλη ενός *πληθυσμού (population)*. Κάθε άτομο χαρακτηρίζεται από την *υγεία του (fitness)*. Η υγεία ενός ατόμου μετράται με την συνδυασμένη τιμή της αντικειμενικής συνάρτησης. Η διαδικασία δουλεύει επαναληπτικά και κάθε επανάληψη είναι μια *γενιά (generation)*. Ο πληθυσμός μιας γενιάς συνίσταται από άτομα που επιζούν από την προηγούμενη γενιά συν τους νέους χρονοπρογραμματισμούς ή τα *παιδιά (children)* από την προηγούμενη γενιά. Το μέγεθος του πληθυσμού συνήθως παραμένει σταθερό από τη μια γενιά στην άλλη. Τα παιδιά γεννώνται μέσω αναπαραγωγής και μεταβολής των ατόμων που ήταν μέρος της προηγούμενης γενιάς (οι *γονείς – parents*). Τα άτομα αναφέρονται ενίοτε και ως “χρωμοσώματα”. Σε ένα περιβάλλον με πολλές μηχανές, ένα χρωμόσωμα μπορεί να συνίσταται από υπό – χρωμοσώματα, κάθε ένα από τα οποία μπορεί να περιέχει την πληροφορία που αφορά την σειρά των διεργασιών σε μια μηχανή. Μια μεταβολή στο χρωμόσωμα ενός γονέα μπορεί να είναι ισοδύναμη με μια αμοιβαία ανταλλαγή γειτονικών ζευγαριών στην αντίστοιχη σειρά. Σε κάθε γενιά, οι περισσότερες διαδικασίες που καθορίζουν την σύνθεση της επόμενης γενιάς μπορεί να είναι σύνθετες και συνήθως να εξαρτώνται από το επίπεδο υγείας των ατόμων της παρούσας γενιάς.

Ένας γενετικός αλγόριθμος, σαν μια διαδικασία αναζήτησης, διαφέρει σε ορισμένα σημεία από τις μεθόδους simulated annealing και tabu search. Σε κάθε επαναληπτικό βήμα ένας αριθμός διαφορετικών χρονοπρογραμματισμών δημιουργούνται και μεταφέρονται στο επόμενο βήμα. Στις άλλες δυο μεθόδους μόνο ένας απλός χρονοπρογραμματισμός μεταφέρεται από την μια επανάληψη στην επόμενη. Έτσι, οι μέθοδοι αυτές μπορούν να θεωρηθούν σαν ειδικές περιπτώσεις των γενετικών αλγορίθμων με μέγεθος πληθυσμού ίσο με 1. Αυτή η διαφοροποίηση είναι ένα σημαντικό χαρακτηριστικό των γενετικών αλγορίθμων. Συνεπώς, στους γενετικούς αλγορίθμους, η σύλληψη της ιδέας της γειτονικής περιοχής δεν βασίζεται σε ένα απλό χρονοπρογραμματισμό αλλά περισσότερο σε μια ομάδα προγραμματισμών. Ο σχεδιασμός της γειτονικής περιοχής του παρόντος πληθυσμού των χρονοπρογραμματισμών βασίζεται σε πιο γενικές τεχνικές από αυτές που χρησιμοποιούνται στις μεθόδους simulated annealing και tabu search. Ένας νέος χρονοπρογραμματισμός μπορεί να παραχθεί συνδυάζοντας μέρη διαφορετικών χρονοπρογραμματισμών μέσα στον πληθυσμό. Για παράδειγμα, στο πρόβλημα χρονοπρογραμματισμού ενός συστήματος job shop, ένας νέος χρονοπρογραμματισμός μπορεί να παραχθεί συνδυάζοντας την ακολουθία των επιμέρους εργασιών σε μια μηχανή, σε ένα παρόντα χρονοπρογραμματισμό, με την ακολουθία διεργασιών σε μια άλλη μηχανή ενός άλλου παρόντος χρονοπρογραμματισμού. Αυτή η διαδικασία αναφέρεται συνήθως ως *crossover effect*.

Η χρήση των μεθόδων simulated annealing, tabu search και των γενετικών αλγόριθμων έχει τα πλεονεκτήματα και τα μειονεκτήματα της. Ένα πλεονέκτημα είναι ότι μπορούν να εφαρμοστούν σε ένα πρόβλημα χωρίς να είμαστε αναγκασμένοι να γνωρίζουμε πολλά για τις δομικές ιδιότητες του προβλήματος. Μπορούν να κωδικοποιηθούν πολύ εύκολα και να δώσουν λύσεις που συνήθως είναι αρκετά καλές. Εντούτοις, το ποσό του υπολογιστικού χρόνου που απαιτείται για να ληφθεί μια τέτοια λύση, τείνει να είναι σχετικά μεγάλο σε σύγκριση με ποιο αυστηρές προσεγγίσεις που λαμβάνουν σε μεγαλύτερο βαθμό τα χαρακτηριστικά του προβλήματος (Brucker P , 2001) .

3.2.4 Κανόνες Απόδοσης Προτεραιοτήτων (Dispatching Rules)

Πρόκειται για κανόνες που δίνουν προτεραιότητα σε εργασίες που αναμένουν για επεξεργασία σε μια μηχανή. Το σχέδιο προτεραιότητας μπορεί να λάβει υπόψη τις ιδιότητες των εργασιών και των μηχανών καθώς και την τρέχουσα χρονική στιγμή. Όταν μια μηχανή καθίσταται διαθέσιμη για να δεχτεί εργασίες, ένα dispatching rule ελέγχει τις εργασίες που αναμένουν και επιλέγει αυτή με τη μεγαλύτερη προτεραιότητα. Στο πεδίο αυτό, υπάρχει ενεργός έρευνα εδώ και αρκετές δεκαετίες και πολλοί κανόνες έχουν αναπτυχθεί και μελετηθεί στη βιβλιογραφία (Brucker P , 2001) .

Τα dispatching rules μπορούν να κατηγοριοποιηθούν με πολλούς τρόπους. Για παράδειγμα, μια διάκριση μπορεί να γίνει μεταξύ των στατικών και δυναμικών κανόνων. Οι *στατικοί κανόνες* δεν εξαρτώνται από τον χρόνο (non time – dependent). Είναι απλά μια συνάρτηση των δεδομένων που αφορούν τις διεργασίες ή τις μηχανές ή και τα δυο μαζί. Οι *δυναμικοί κανόνες* από την άλλη πλευρά, εξαρτώνται από τον χρόνο (time – dependent).

Ένας δεύτερος τρόπος κατηγοριοποίησης των dispatching rules είναι σύμφωνα με την πληροφορία στην οποία βασίζονται. Ένας *τοπικός κανόνας* (*local rule*) χρησιμοποιεί μόνο πληροφορία που αφορά στην ουρά όπου αναμένει η εργασία ή στη μηχανή στην οποία πρόκειται να υποστεί επεξεργασία. Ένας *γενικός κανόνας* (*global rule*) μπορεί να χρησιμοποιήσει πληροφορία που αφορά άλλες μηχανές, όπως ο χρόνος επεξεργασίας της εργασίας στην επόμενη μηχανή του δρομολογίου ή το τρέχον μήκος της ουράς σε αυτή τη μηχανή.

Στη συνέχεια παρατίθενται ορισμένοι από αυτούς τους κανόνες:

- **Service in random order (SIRO) rule.** Σύμφωνα με αυτό τον κανόνα, όποτε μια μηχανή καθίσταται διαθέσιμη, η επόμενη διεργασία επιλέγεται στη τύχη, ανάμεσα από αυτές που αναμένουν για επεξεργασία. Δεν γίνεται προσπάθεια για βελτιστοποίηση κανενός αντικειμενικού στόχου.
- **Earliest release date first (ERD) rule.** Αυτός ο κανόνας είναι κατά κάποιο τρόπο ισοδύναμος με τον κανόνα first-come-first-served. Κατά μια έννοια ελαχιστοποιεί τις αποκλίσεις στους χρόνους αναμονής των διεργασιών σε μια μηχανή.
- **Earliest due date first (EDD) rule.** Όποτε μια μηχανή καθίσταται διαθέσιμη, η διεργασία με τον νωρίτερο χρόνο προθεσμίας επιλέγεται για να επεξεργαστεί. Αυτός ο κανόνας τείνει να ελαχιστοποιεί την μέγιστη καθυστέρηση (maximum lateness) ανάμεσα στις διεργασίες που αναμένουν για επεξεργασία. Στην πράξη, σε ένα περιβάλλον με μια απλή μηχανή, με n διεργασίες διαθέσιμες την χρονική στιγμή 0, ο κανόνας EDD όντως ελαχιστοποιεί την μέγιστη καθυστέρηση.
- **Minimum slack first (MS) rule.** Σύμφωνα με τον κανόνα, οι εργασίες διατάσσονται σύμφωνα με το κριτήριο $\max(d_j - p_j - t, 0)$ όπου d_j είναι ο χρόνος προθεσμίας, p_j ο χρόνος επεξεργασίας και t ο τρέχων χρόνος. Λόγω αυτού του ορισμού συνεπάγεται πως σε κάποια χρονική στιγμή η εργασία j μπορεί να έχει μεγαλύτερη προτεραιότητα από τη εργασία k ενώ σε κάποια επόμενη χρονική στιγμή μπορεί να έχουν την ίδια προτεραιότητα.
- **Weighted shortest processing time first (WSPT) rule.** Όποτε μια μηχανή καθίσταται διαθέσιμη, η διεργασία με τον μεγαλύτερο λόγο βάρους (w_j) προς χρόνο επεξεργασίας (p_j) προγραμματίζεται για τη συνέχεια. Έτσι οι διεργασίες προγραμματίζονται με φθίνουσα σειρά w_j / p_j. Ο κανόνας αυτός τείνει να μειώνει το σταθμισμένο (weighted) άθροισμα των χρόνων ολοκλήρωσης, δηλ. το $\Sigma w_j C_j$. Σε περιβάλλον με μια απλή μηχανή με n διεργασίες διαθέσιμες την χρονική στιγμή 0, αυτό όντως ισχύει. Όταν όλα τα βάρη είναι ίσα, ο κανόνας WSPT ανάγεται στον κανόνα **shortest processing time first (SPT)**.
- **Longest processing time first (LPT) rule.** Αυτός ο κανόνας διατάσσει τις διεργασίες σε φθίνουσα σειρά χρόνων επεξεργασίας. Όταν έχουμε μηχανές σε παράλληλη διάταξη αυτός ο κανόνας τείνει να εξισορροπήσει το φόρτο εργασίας σε κάθε μηχανή. Η λογική που ακολουθεί είναι η εξής: είναι

επωφελές να κρατάμε τις διεργασίες με μικρούς χρόνους επεξεργασίας για αργότερα γιατί αυτές οι διεργασίες είναι χρήσιμες προς το τέλος για την εξισορρόπηση του φόρτου των μηχανών. Αφού καθορίσουμε την ανάθεση των διεργασιών στις μηχανές, οι διεργασίες σε οποιαδήποτε μηχανή μπορούν να αλλάξουν ξανά σειρά χωρίς να επηρεαστεί η ισορροπία του φόρτου των μηχανών.

- **Shortest setup time first (SST) rule.** Όποτε μια μηχανή καθίσταται διαθέσιμη, ο κανόνας αυτός επιλέγει για επεξεργασία την διεργασία με τον συντομότερο χρόνο προετοιμασίας (setup time).
- **Critical path (CP) rule.** Ο κανόνας αυτός χρησιμοποιείται όταν οι διεργασίες υπόκεινται σε περιορισμούς προτεραιότητας (precedence constraints). Από τον αντίστοιχο γράφο που εκφράζει αυτούς τους περιορισμούς, επιλέγεται η διεργασία που είναι στη κεφαλή της μακρύτερης αλληλουχίας των χρόνων επεξεργασίας.
- **Largest number of successors (LNS) rule.** Ο κανόνας αυτός μπορεί να χρησιμοποιηθεί επίσης όταν οι διεργασίες υπόκεινται σε περιορισμούς προτεραιότητας. Επιλέγει την διεργασία που έχει τον μεγαλύτερο αριθμό διεργασιών που να την ακολουθούν.

Τα dispatching rules είναι χρήσιμα για την εύρεση σχετικά ικανοποιητικών λύσεων αν έχουμε σχετικά απλούς αντικειμενικούς στόχους, όπως το κριτήριο makespan, ο συνολικός χρόνος ολοκλήρωσης ή η μέγιστη καθυστέρηση. Εντούτοις, στην πράξη οι αντικειμενικοί στόχοι μπορεί να είναι συνδυασμοί αρκετών βασικών στόχων και επίσης μια συνάρτηση του χρόνου ή της ομάδας των διεργασιών που περιμένουν για επεξεργασία. Έτσι, αν τους χρησιμοποιήσουμε αυτούσιους, είναι δυνατό να οδηγηθούμε σε μη αποδεκτούς χρονοπρογραμματισμούς.

Ένα βασικό μειονέκτημα των dispatching rules, είναι ότι το αν θα δώσουν αποδεκτή ή όχι λύση εξαρτάται σε μεγάλο βαθμό από τη φύση του προβλήματος για το οποίο εφαρμόζονται. Έχουν περιορισμένη ικανότητα σχετικά με τη λήψη αποφάσεων, καθώς λαμβάνουν υπόψη μόνο την τρέχουσα κατάσταση της μηχανής και του περιβάλλοντος που σχετίζεται άμεσα με αυτή. Επιπλέον, η ποιότητα της λύσης υποβαθμίζεται καθώς αυξάνεται η πολυπλοκότητα του προβλήματος (Brucker P , 2001) , (Jayamohan M. S. , Rajendran C. 2000) .

3.2.5 Τεχνητή Νοημοσύνη και Νευρωνικά Δίκτυα

Η τεχνητή νοημοσύνη είναι το πεδίο της επιστήμης των υπολογιστών που αφορούν στον συνδυασμό της υπολογιστικής ευφυΐας και τη βιολογικής νοημοσύνης. Έχει τις ρίζες της στην βιολογική κατανόηση και χρησιμοποιεί «αρχές» από τη φύση προκειμένου να καταφέρει να δώσει λύσεις. Χρησιμοποιώντας, λοιπόν, αυτήν την κατανόηση της φύσης ο βασικός στόχος της τεχνητής νοημοσύνης είναι το να καταφέρουν να κάνουν τους υπολογιστές πιο χρήσιμους στην επίλυση των προβλημάτων. Δύο βασικές μεθοδολογίες της τεχνητής νοημοσύνης μπορούν να βρουν εφαρμογή στην επίλυση του δεδομένου προβλήματος, το οποίο μελετάμε: η προσέγγιση που ικανοποιεί δεδομένους περιορισμούς *(constraint satisfaction – CS)* και η μέθοδος των νευρωνικών δικτύων.

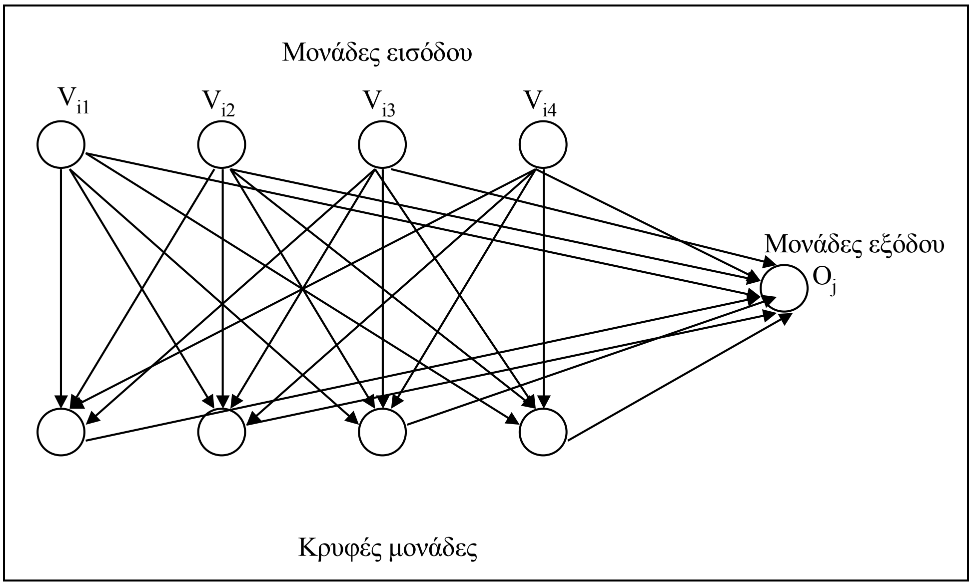
Η πρώτη μέθοδος της προσέγγισης με βάση την ικανοποίηση περιορισμών έχει σκοπό την σημαντική μείωση του χώρου όπου θα διεξαχθεί η έρευνα για την επίλυση του προβλήματος αφού χρησιμοποιεί κανόνες, οι οποίοι περιορίζουν τη σειρά με την οποία επιλέγονται οι διάφορες παράμετροι καθώς και τη διαδοχή με την οποία οι πιθανές τιμές ανατίθενται στις μεταβλητές αυτές. Το πρόβλημα των περιορισμών *(Constrained Satisfaction Problem)* θεωρείται ότι έχει λυθεί όταν έχουμε πια αναθέσει τιμές σε όλες τις μεταβλητές του προβλήματος με τρόπο τέτοιο ώστε να μην παραβιάζονται οι περιορισμοί του προβλήματος (Dornan B. , 1987) .

Παρ' ότι θεωρούνται μέρος της τεχνητής νοημοσύνης οι μέθοδοι αυτές να χρησιμοποιούν μια συστηματική, δενδρικής μορφής, έρευνα, η οποία θεωρείται ότι πλησιάζει κατά πολύ τους αλγορίθμους των τεχνικών branch and bound, που ήδη έχουν αναλυθεί παραπάνω.

Τα νευρωνικά δίκτυα είναι οργανωμένα όπως ακριβώς και ο ανθρώπινος εγκέφαλος. Ένα νευρωνικό δίκτυο συντίθεται από ένα αριθμό διασυνδεδεμένων νευρώνων ή μονάδων. Η πληροφορία υπόκειται επεξεργασία μέσα σε ένα πλήρως διασυνδεδεμένο δίκτυο με μονάδες παράλληλης επεξεργασίας. Οι συνδέσεις μεταξύ των μονάδων έχουν βάρη που αναπαριστούν τις δυνάμεις των συνδέσεων μεταξύ των μονάδων. Ένα πολυστρωματικό, feed – forward δίκτυο αποτελείται από μονάδες εισόδου, κρυφές μονάδες και μονάδες εξόδου.

Η γνώση του δικτύου αποθηκεύεται στα βάρη και υπάρχουν μέθοδοι που προσαρμόζουν τα βάρη στις συνδέσεις για την λήψη κατάλληλων αποκρίσεων. Για κάθε διάνυσμα εισόδου υπάρχει ένα πιο κατάλληλο διάνυσμα εξόδου. Ένας αλγόριθμος εκμάθησης υπολογίζει τη διαφορά μεταξύ του διανύσματος εξόδου του τρέχοντος δικτύου και του πιο κατάλληλου διανύσματος εξόδου και προτείνει αυξητικές ρυθμίσεις στα βάρη. Ένας τέτοιος αλγόριθμος είναι αλγόριθμος ανάστροφης μετάδοσης λάθους (back propagation learning algorithm) (Dornan B. , 1987)

Η απλότητα τους καθώς και η ικανότητα τους να μαθαίνουν και να γενικεύουν έχουν κάνει τα νευρωνικά δίκτυα μια πολύ δημοφιλή μεθοδολογία, που μπορεί να βρει εφαρμογή σε πολλά προβλήματα.



Σχήμα 3. 1: Παράδειγμα Νευρωνικού Δικτύου

ΚΕΦΑΛΑΙΟ 4

Χρονοπρογραμματισμός Job-shop Συστημάτων με Χρονισμένα Αυτόματα

4. ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ JOB-SHOP ΣΥΣΤΗΜΑΤΩΝ ΜΕ ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ

4.1 ΕΙΣΑΓΩΓΙΚΑ ΣΤΟΙΧΕΙΑ ΣΤΑ ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ

Ορισμός 1 (Χρονισμένα αυτόματα): Ένα χρονισμένο αυτόματο είναι μια πλειάδα της μορφής $A=(Q,C,s,f,\Delta)$ όπου το Q είναι ένα πεπερασμένο σύνολο καταστάσεων, το C είναι ένα πεπερασμένο σύνολο χρονιστών (clocks), το Δ είναι μια σχέση μετάβασης που περιλαμβάνει στοιχεία της μορφής (q,ϕ,p,q') όπου q και q' εκφράζουν καταστάσεις, $p \subseteq C$ και τέλος το ϕ είναι ο βοηθός μετάβασης (transition guard) και αποτελεί ένα συνδυασμό της άλγεβρας Boole της μορφής ($c \in I$) για κάποιο χρονιστή c και κάποιο διάστημα εύρους ακεραίων (integer-bounded interval) I. Τα s και f είναι η αρχική και τελική κατάσταση αντίστοιχα.

Ορισμός 2 (Τιμές των χρονιστών): Η τιμή ενός χρονιστή είναι μια συνάρτηση της μορφής $v: C \to \Re_+ \cup \{0\}$, ή ισοδύναμα ένα C-διαστάσεων διάνυσμα στο σύνολο $\Re_+$. Συμβολίζουμε το σύνολο των τιμών των χρονιστών με H. Ένα αυτόματο είναι ένα ζεύγος της μορφής $(q,v) \in Q \times H$ που περιλαμβάνει μια διακριτή κατάσταση (μερικές φορές ονομάζεται και “τοποθεσία”) και μια τιμή ενός χρονιστή. Κάθε υποσύνολο $p \subseteq C$ επάγετε (induces) μια συνάρτηση μηδενισμού (reset function) $\mathrm{Re}set_p : H \to H$ που ορίζεται για κάθε τιμή χρονιστή v και για κάθε μεταβλητή χρονιστή $c \in C$ ως

$$\mathrm{Re}set_p v(c) = \begin{cases} 0 & \alpha\nu \ \ c \in \rho \\ v(c) & \alpha\nu \ \ c \notin \rho \end{cases}$$

Αυτό σημαίνει ότι η συνάρτηση $\mathrm{Re}set_p$ μηδενίζει όλους τους χρονιστές στο ρ ενώ αφήνει όλους τους υπόλοιπους χρονιστές ανεπηρέαστους. Τέλος χρησιμοποιούμε το **1** για να δηλώσουμε το μοναδιαίο διάνυσμα (1,…,1) και το **0** για το μηδενικό διάνυσμα.

Ορισμός 3 (Βήματα και εκτελέσεις): Σε ένα αυτόματο υπάρχουν δυο ειδών βήματα:

i. Διακριτά βήματα (discrete steps): $(q,v) \xrightarrow{0} (q',v')$, όπου υπάρχει μετάβαση $\delta = (q,\phi,\rho,q') \in \Delta$, τέτοια ώστε ο v να ικανοποιεί το ϕ και $v' = \mathrm{Re}set_p(v)$.

ii. Χρονικά βήματα (time steps): $(q,v) \xrightarrow{t} q, v + t \cdot 1), t \in \Re_+$

Ξεκινώντας από ένα ζεύγος της μορφής (q_0, v_0), θεωρούμε ως εκτέλεση ενός αυτομάτου μια πεπερασμένη ακολουθία βημάτων της μορφής:

$$\xi: \quad (q_0, v_0) \xrightarrow{t_1} (q_1, v_1) \xrightarrow{t_2} (q_2, v_2) \xrightarrow{t_3} \ldots \xrightarrow{t_n} (q_n, v_n)$$

Το λογικό μήκος μιας τέτοιας εκτέλεσης είναι n, ενώ το μετρικό της μήκος είναι $t_1 + t_2 + t_3 + \ldots + t_n$.

Πολλές φορές θεωρείται χρήσιμο να προσθέτουμε στο χρονισμένο αυτόματο A έναν επιπλέον χρονιστή t που δείχνει το συνολικό χρόνο και ο οποίος έχει το χαρακτηριστικό ότι είναι σε κάθε κατάσταση ενεργός και δεν μηδενίζεται ποτέ. Έτσι προκύπτει ένα καινούργιο αυτόματο A' που το ονομάζουμε ***διευρυμένο αυτόματο* (*extended automaton*)** του A ενώ τις εκτελέσεις του τις αποκαλούμε ***διευρυμένες εκτελέσεις*** του A. Η κατάσταση (q,v,t) στο A' μπορεί να προσπελαστεί αν και μόνο αν η κατάσταση (q,v) στο A μπορεί να προσπελαστεί με το πέρασμα χρόνου t.

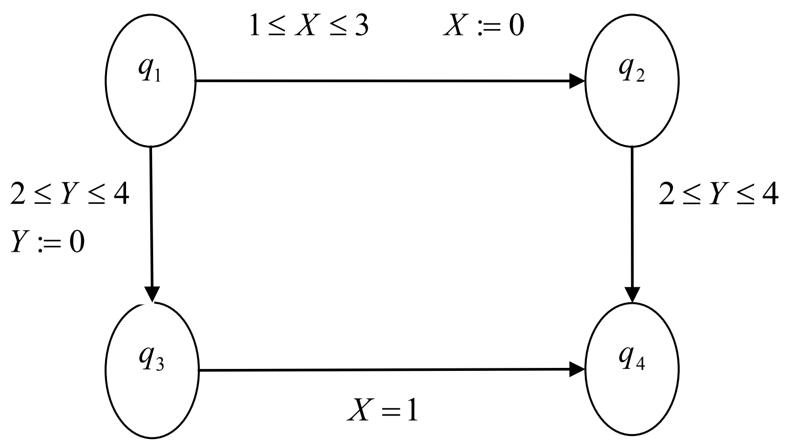
Αξίζει να σημειωθεί ότι οι τίτλοι "start" και "end" που χρησιμοποιήθηκαν για ορισμένες μεταβάσεις στις εκτελέσεις σε προηγούμενη ανάλυση, δεν υπάρχει ανάγκη να υφίστανται εδώ και συνεπώς αντί αυτών θα χρησιμοποιείται η διάρκεια των συγκεκριμένων μεταβάσεων που είναι 0. Αν και στον ορισμό θεωρήσαμε ότι όλοι οι χρονιστές αναπτύσσονται ομοιόμορφα με παράγωγο χρόνο τη μονάδα μέσα στις καταστάσεις, εντούτοις υπάρχουν καταστάσεις στις οποίες συγκεκριμένες τιμές χρονιστών δεν είναι σημαντικές επειδή σε όλες τις διαδρομές που προκύπτουν ξεκινώντας από αυτές τις καταστάσεις οι συγκεκριμένοι χρονιστές δεν ελέγχονται πριν μηδενιστούν. Τέτοια παραδείγματα αποτελούν ο χρονιστής X_1 στη κατάσταση $(\overline{p}_1, p_2)$ και ο χρονιστής X_2 στη κατάσταση $(p_1, \overline{p}_2)$. Έτσι θα λέμε ότι ένας χρονιστής είναι **μη ενεργός (inactive)** και για να το δείξουμε αυτό χρησιμοποιούμε το σύμβολο $\perp$.

Παράδειγμα: Παρακάτω παρουσιάζεται η εκτέλεση του αυτομάτου A'' του Σχήματος 3.7 με την υπόθεση ότι η διαδικασία P_1 ξεκινάει μετά από 0.20 χρονικές μονάδες και η διαδικασία P_2 ξεκινάει 2.5 χρονικές μονάδες αφότου ξεκινήσει η P_1:

$$(\overline{p}_1,\overline{p}_2,\bot,\bot)\xrightarrow{0.20}(p_1,\overline{p}_2,0,\bot)\xrightarrow{2.5}(p_1,\overline{p}_2,2.5,\bot)\xrightarrow{0}(p_1,p_2,2.5,0)\xrightarrow{0.5}$$
$$(p_1,p_2,3,0.5)\xrightarrow{0}(\underline{p}_1,p_2,\bot,0.5)\xrightarrow{1.5}(\underline{p}_1,p_2,\bot,2)\xrightarrow{0}(\underline{p}_1,\underline{p}_2,\bot,\bot)$$

Επίσης αξίζει να σημειώσουμε ότι από τον ορισμό των εκτελέσεων που δώσαμε, μας δίνεται η δυνατότητα να "διαιρέσουμε" χρονικά τα βήματα. Αυτό σημαίνει ότι για παράδειγμα το βήμα $(p_1,\overline{p}_2,0,\bot)\xrightarrow{2,5}(p_1,\overline{p}_2,2.5,\bot)$ μπορεί να γραφτεί ως $(p_1,\overline{p}_2,0,\bot)\xrightarrow{0,9}(p_1,\overline{p}_2,0.9,\bot)\xrightarrow{0,6}(p_1,\overline{p}_2,1.5,\bot)\xrightarrow{1,0}(p_1,\overline{p}_2,2.5,\bot)$ όπως και με κάθε άλλο τρόπο.

Ένα από τα πιο σημαντικά στοιχεία των χρονισμένων αυτομάτων είναι η ικανότητα που διαθέτουν στο να αναπαριστούν και να αναλύουν συστήματα στα οποία υπάρχει αβεβαιότητα ως προς το χρόνο. Συγκεκριμένα στο Σχήμα 3.8 παρατηρούμε ότι η μετάβαση από τη κατάσταση q_1 στη q_2 μπορεί να συμβεί όταν η τιμή του χρονιστή X λάβει οποιαδήποτε τιμή στο διάστημα [1,3].



Σχήμα 4.1: Ένα χρονισμένο αυτόματο

Αν και το σύνολο των εκτελέσεων που προκύπτει σε ένα τέτοιο αυτόματο είναι άπειρο, εντούτοις προβλήματα που έχουν να κάνουν με προσπέλαση και επαλήθευση είναι δυνατών να αντιμετωπιστούν με εφαρμογή κατάλληλων αλγορίθμων. Η βασική ιδέα για προσπέλαση στα χρονισμένα αυτόματα είναι η ακόλουθη:

1. Ένα σύνολο από προσπελάσιμους σχηματισμούς (reachable configurations) καταχωρούνται ως ενώσεις ***συμβολικών καταστάσεων (symbolic states)*** της μορφής (q,Z) όπου q είναι μια διακριτή κατάσταση και Z είναι ένα υποσύνολο του συνόλου των χρονιστών H.
2. Ο υπολογισμός των ακολούθων ή αλλιώς διάδοχων καταστάσεων μιας συμβολικής κατάστασης γίνεται σε δυο φάσεις. Πρώτα υπολογίζονται όλοι οι ακόλουθοι της (q,Z) που οδηγούν στη (q,Z'). Το Z' περιέχει όλες τις τιμές των χρονιστών που μπορούν να προσπελαστούν από το Z. Μετά το Z' διασταυρώνεται με το βοηθό μετάβασης κάθε μιας μετάβασης με σκοπό να προσδιορίσουμε το σχηματισμό στον οποίο η μετάβαση θα λάβει χώρα και εν συνεχεία εφαρμόζονται οι λειτουργίες επαναφοράς στους σχηματισμούς.

Ας εξετάσουμε το αυτόματο του Σχήματος 4.8. Ξεκινώντας από τη συμβολική κατάσταση $(q_1, X = Y = 0)$, και αφήνοντας το χρόνο να περάσει προσεγγίζουμε τη συμβολική κατάσταση $(q_1, X = Y \geq 0)$. Η τομή με το βοηθό της μετάβασης στο q_2 δίνει $(q_1, 1 \leq X = Y \leq 3)$ και ο μηδενισμός του X μας οδηγεί τελικά στη $(q_2, X = 0 \wedge 1 \leq Y \leq 3)$ από όπου ξεκινάμε ξανά κοκ. Ολόκληρος ο υπολογισμός του αυτομάτου απεικονίζεται στο Σχήμα 4.9. Το σημαντικότερο αποτέλεσμα αναφορικά με το χρονισμένο αυτόματο είναι ότι το σύνολο των τιμών των χρονιστών στις συμβολικές καταστάσεις ανήκει σε μια ειδική τάξη πολυέδρων (polyhedra) που λέγονται **ζώνες (zones)** και οι οποίες είναι πεπερασμένες για κάθε αυτόματο.

Ορισμός 4 (Ζώνες και συμβολικές καταστάσεις): Ζώνη είναι ένα υποσύνολο του H που περιλαμβάνει τα σημεία που ικανοποιούν μια σύζευξη από ανισότητες της μορφής $c_i - c_j \geq d$ ή $c_i \geq d$. Συμβολική κατάσταση είναι ένα ζεύγος (q,Z) όπου q είναι μια διακριτή κατάσταση και Z είναι μια ζώνη. Προκύπτει ένα σύνολο από σχηματισμούς $\{(q,Z) : z \in Z\}$.

Ορισμός 5 (Ακόλουθες): Έστω $A = (Q, C, s, f, \Delta)$ είναι ένα χρονισμένο αυτόματο και (q,Z) είναι μια συμβολική κατάσταση.

- Μια χρονικά ακόλουθη (time successor) της (q,Z) είναι ένα σύνολο από σχηματισμούς οι οποίες είναι προσπελάσιμες από την συμβολική κατάσταση (q,Z) εφόσον περάσει κάποια ποσότητα χρόνου

$$Post^{t}(q,Z)=\{(q,z+r1):z\in Z,r\geq 0\}$$

Λέμε ότι η (q,Z) είναι χρονικά κλειστή (time-closed) αν

$$(q,Z)=Post^{t}(q,Z)$$

- Μια δ-ακόλουθη μετάβαση της συμβολικής κατάστασης (q,Z) είναι ένα σύνολο από σχηματισμούς που είναι προσπελάσιμες από την (q,Z) αρκεί να λάβουμε τη μετάβαση $\delta=(q,\varphi,\rho,q')\in\Delta$:

$$Post^{\delta}(q,Z)=\{(q',\mathrm{Re}set_{p}(z)):z\in Z\cap\varphi\}$$

- Μια δ-ακόλουθη μιας χρονικά κλειστής συμβολικής κατάστασης (q,Z) είναι ένα σύνολο από σχηματισμούς που μπορούν να προσπελαστούν λαμβάνοντας μια μετάβαση δ που πραγματοποιείται με το πέρασμα του χρόνου:

$$Succ^{\delta}(q,Z)=Post^{t}(Post^{\delta}(q,Z))$$

- Οι ακόλουθες καταστάσεις της συμβολικής κατάστασης (q,Z) είναι το σύνολο όλων των δ-ακολούθων:

$$Succ(q,Z)=\bigcup_{\delta\in\Delta}(Succ^{\delta}(q,Z))$$

Έχοντας πλέον στη διάθεσή μας αυτές τις λειτουργίες (που μετασχηματίζουν ζώνες σε ζώνες) μπορούμε να επιλύσουμε προβλήματα προσπέλασης καταστάσεων σε χρονισμένα αυτόματα χρησιμοποιώντας αλγόριθμους γραφικής αναζήτησης (graph search algorithms) οι οποίοι εφαρμόζονται σε “γραφήματα προσομοίωσης” (simulation graph). Τα γραφήματα προσομοίωσης είναι γραφήματα που περιέχουν κόμβους που αναπαριστούν συμβολικές καταστάσεις και οι οποίες συνδέονται μεταξύ τους μέσω σχέσεων ακολούθων. Όπως θα δούμε στο επόμενο κεφάλαιο τα χρονισμένα αυτόματα που χρησιμοποιούνται για τη μοντελοποίηση Job-shop συστημάτων είναι μη κυκλικά ή άκυκλα (acyclic). Παρακάτω παρουσιάζεται ένας γενετικός αλγόριθμος ο οποίος υπολογίζει όλους τους προσπελάσιμους σχηματισμούς σε ένα τέτοιο μη κυκλικό αυτόματο ξεκινώντας από το σχηματισμό $(s,0)$.

Αλγόριθμος 1 (Ευθεία Προσπέλαση σε Άκυκλα Χρονισμένα Αυτόματα)

$Waiting := \{Post^t\{(s,0)\}\};$

while $Waiting \neq \emptyset$ **do**

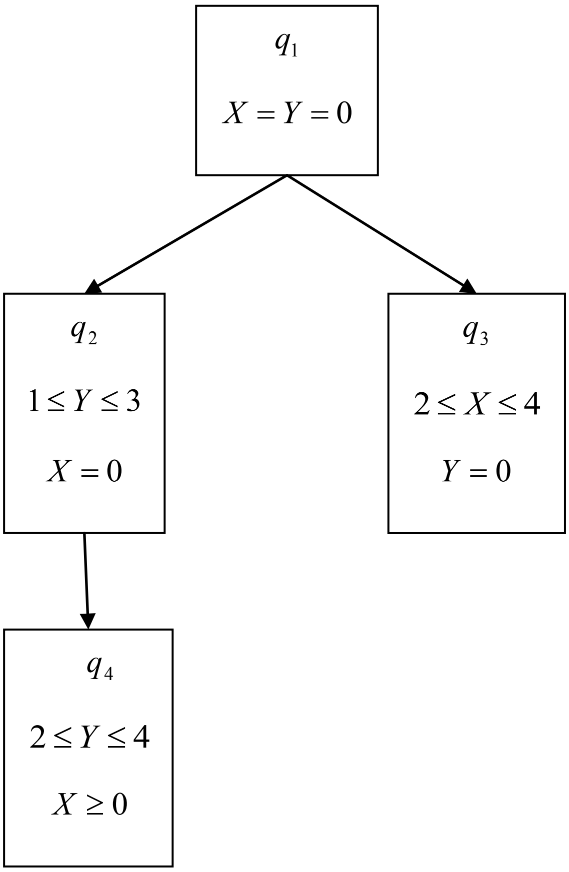
$Pick(q,Z) \in Waiting;$

For every $(q',Z') \in Succ(q,Z);$

Insert (q',Z') *into Waiting;*

Remove (q,Z) *from Waiting;*

End



Σχήμα 4.2: Υπολογισμός ευθείας προσπέλασης για το χρονισμένο αυτόματο του Σχήματος 4.1

Από τη στιγμή που το αυτόματο είναι άκυκλο ο αλγόριθμος θα ολοκληρωθεί όταν πλέον δεν θα υπάρχει λίστα στην οποία να υπάρχουν καταστάσεις προς εξερεύνηση. Παρόλα αυτά είναι σημαντικό να πραγματοποιούνται έλεγχοι διότι υπάρχουν περιπτώσεις που μπορούμε να έχουμε πολλά μονοπάτια που να οδηγούν στην ίδια διακριτή κατάσταση.

Εκτός της ευθείας προσπέλασης, τα προσπελάσιμα σύνολα μπορούν να υπολογιστούν με ανάστροφο τρόπο εφαρμόζοντας αλγορίθμους ανάστροφης προσπέλασης οι οποίοι υπολογίζουν τις προηγούμενες συμβολικές καταστάσεις μιας δεδομένης συμβολικής κατάσταση.

Ορισμός 6 (Προηγούμενες): Έστω $A = (Q, C, s, f, \Delta)$ είναι ένα χρονισμένο αυτόματο και έστω (q, Z) είναι μια συμβολική κατάσταση.

- Οι χρονικά προηγούμενες καταστάσεις μιας συμβολικής κατάστασης (q,Z) είναι ένα σύνολο από σχηματισμούς από τους οποίους μπορούμε να φτάσουμε στη συμβολική κατάσταση (q,Z) αν αφήσουμε το χρόνο να περάσει.

$$\mathrm{Pr}e^{t}(q,Z)=\{(q,v): v+r\cdot 1\in Z, r\geq 0\}$$

Λέμε ότι η (q,Z) είναι χρονικά κλειστή (time- closed) αν

$$(q,Z)=\mathrm{Pr}e^{t}(q,Z)$$

- Μια μετάβαση *δ* της προηγούμενης μιας συμβολικής κατάστασης (q,Z) είναι ένα σύνολο από σχηματισμούς από τους οποίους μπορούμε να φτάσουμε στη (q,Z) εφόσον ακολουθήσουμε αυτή τη μετάβαση $\delta=(q',\phi,p,q)\in\Delta$.

$$\mathrm{Pr}e^{\delta}(q,Z)=\{(q',v'): v'\in \mathrm{Re}set_{p}^{-1}(Z)\cap\phi\}$$

- Οι προηγούμενες καταστάσεις της συμβολικής κατάστασης (q,Z) είναι ένα σύνολο που περιλαμβάνει όλους τους σχηματισμούς από τους οποίους μπορεί κανείς να προσπελάσει τη (q,Z) λαμβάνοντας οποιαδήποτε μετάβαση δ που πραγματώνεται με το πέρασμα του χρόνου.

$$\mathrm{Pr}e(q,Z)=\bigcup_{\delta\in\Delta}\mathrm{Pr}e^{t}(\mathrm{Pr}e^{\delta}(q,Z))$$

Ο παρακάτω αλγόριθμος υπολογίζει το σύνολο των καταστάσεων από τους οποίους μπορεί να προσεγγιστεί η τελική κατάσταση *f*.

Αλγόριθμος 2 (Ανάστροφη Προσπέλαση σε Άκυκλα Χρονισμένα Αυτόματα)

$Waiting := \{(f, H)\};$

while $Waiting \neq \emptyset$ **do**

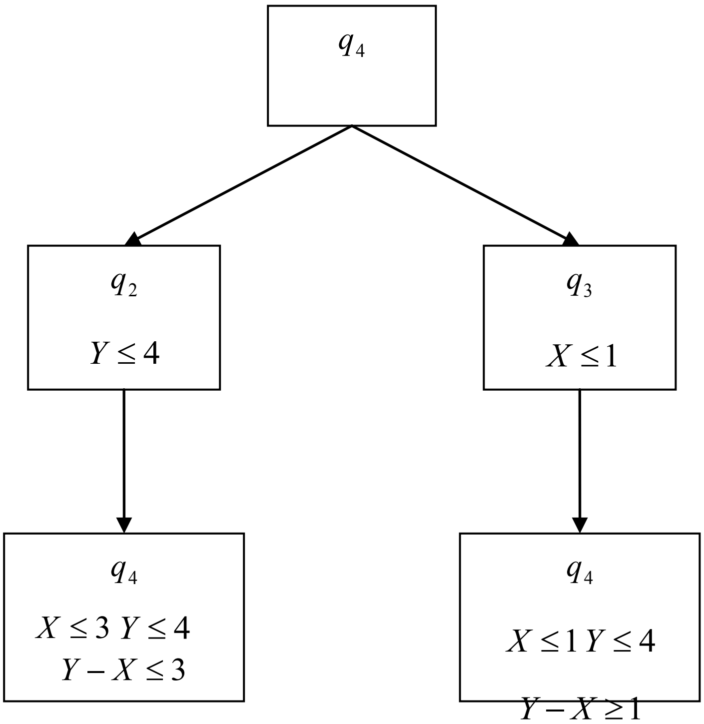
  Pick $(q, Z) \in Waiting;$

  For every $(q', Z') \in \mathrm{Pr}e(q, Z);$

    Insert (q', Z') *into Waiting;*

  Remove (q, Z) *from Waiting;*

end



Σχήμα 4.3: Υπολογισμός ανάστροφης προσπέλασης για το χρονισμένο αυτόματο του Σχήματος 4.1

4.2 ΝΤΕΤΕΡΜΙΝΙΣΤΙΚΟΣ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ JOB-SHOP ΣΥΣΤΗΜΑΤΩΝ

Σε αυτό το σημείο θα μοντελοποιήσουμε το πρόβλημα του χρονοπρογραμματισμού συστημάτων κατά παραγγελία (Job-shop scheduling problem) χρησιμοποιώντας τη θεωρία των άκυκλων χρονισμένων αυτομάτων (acyclic timed automata) που παρουσιάστηκε στο προηγούμενο κεφάλαιο. Στόχος αυτής της προσπάθειας είναι να πετύχουμε το βέλτιστο χρονοπρόγραμμα, δηλαδή στην ουσία να βρούμε τη συντομότερη διαδρομή στο αυτόματο από πλευράς χρόνου. Στη προσπάθεια για την εύρεση της συντομότερης διαδρομής στα χρονισμένα αυτόματα, παρουσιάζονται αλγόριθμοι (ευρετικοί και άλλοι) .

Ορισμός 7 (Χαρακτηριστικά Job-shop συστήματος): Έστω M είναι ένα πεπερασμένο σύνολο πόρων (μηχανές). Μια εργασία J πάνω σε ένα σύνολο μηχανών M προσδιορίζεται ως μια τριάδα $J=(k,\mu,\mathrm{d})$, όπου $k \in \aleph$ είναι ο αριθμός των βημάτων (διεργασιών) της J, $\mu:\{1,\ldots,\mathrm{k}\}\rightarrow \mathrm{M}$ δείχνει ποια μηχανή χρησιμοποιείται σε κάθε βήμα και $d:\{1,\ldots,k\}\rightarrow \aleph$ δείχνει τη διάρκεια κάθε βήματος. Σε ένα σύστημα Job-shop υπάρχει ένα σύνολο εργασιών $J=\{J^1,\ldots,J^n\}$ με $J^i=(k^i,\mu^{\mathrm{i}},d^i)$ και $i=1,\ldots,n$.

Επίσης υποθέτουμε ότι κάθε μηχανή χρησιμοποιείται από μια και μόνο εργασία κάθε φορά, όπως επίσης και ότι ο χρόνος T παίρνει τιμές στο σύνολο $\Re_+$ ενώ $J=\{1,\ldots,n\}$ και $K=\{1,\ldots,k\}$.

Ορισμός 8 (Εφικτά χρονοπρογράμματα): Ένα εφικτό χρονοπρόγραμμα ενός συστήματος Job-shop που αποτελείται από n-εργασίες $J=\{J^1,\ldots,J^n\}$ είναι μια σχέση $\mathrm{S}\subseteq \mathrm{J}\times\mathrm{K}\times\mathrm{T}$ τέτοια ώστε $(i,j,t)\in S$ και δείχνει ότι η J^i εργασία εκτελεί το j^{th} βήμα σε κάποιο χρόνο t και απασχολεί τη μηχανή $\mu^{\mathrm{i}}(j)$. Ένα πρόγραμμα παραγωγής πρέπει να ικανοποιεί τις παρακάτω συνθήκες:

1. Βήματα που αναφέρονται στην ίδια εργασία εκτελούνται με την ίδια σειρά (ταξινόμηση/ ordering) . Αν $(i,j,t)\in S$ και $(i,j',t')\in S$ τότε $j\le j'$ και $t\le t'$.

2. Κάθε βήμα εκτελείται συνεχώς μέχρι την πλήρη ολοκλήρωσή του (κάλυψη και μη-προεκτόπιση /covering and non-preemption). Για κάθε $i\in J$ και $j\in K$ το σύνολο $\{t:(i,j,t)\in S\}$ είναι ένα μη προεκτοπιστικό (non-empty) σύνολο της μορφής $[r,r+d]$ για κάποιο $r\in T$ και $d=d^i(j)$.

3. Δύο διεργασίες δύο διαφορετικών εργασιών οι οποίες εκτελούνται την ίδια χρονική στιγμή δεν χρησιμοποιούν την ίδια μηχανή (αμοιβαία απόκλιση/ mutual exclusion). Για κάθε *i*, $i\in J$, $j\in K$, $t\in T$, αν $(i,j,t)\in S$ και $(i',j',t)\in S$ τότε $\mu^{\mathrm{i}}(j)\neq\mu^{\mathrm{i}'}(j')$.

- Ο χρόνος έναρξης της διεργασίας *j* της εργασίας *i* είναι

$$s(i,j)=\min_{(i,j,t)\in S} t$$

- Η διάρκεια του χρονοπρογράμματος είναι το μέγιστο *t* όλων των βημάτων $(i,j,t) \in S$.

- Το βέλτιστο χρονοπρόγραμμα των εργασιών *J* ενός συστήματος Job-shop είναι εκείνο το χρονοπρόγραμμα με τη μικρότερη χρονική διάρκεια.

Από τους προηγούμενους ορισμούς που δώσαμε αναφορικά με τα χρονοπρογράμματα μπορούμε να εξάγουμε δυο συναρτήσεις:

1. Συνάρτηση κατανομής των μηχανών *α* (The machine allocation function): $\mathrm{M} \times \mathrm{T} \to J$
   Η συνάρτηση αυτή ορίζεται ως *α*(*m,t*)=*i* αν $(i,j,t) \in S$ και $\mu^{\mathrm{i}}(j) = m$. Δείχνει ποια εργασία απασχολεί ποια μηχανή και για πόσο χρόνο.

2. Συνάρτηση προόδου των εργασιών *β* (The task progress function): $J \times T \to M$
   Η συνάρτηση αυτή ορίζεται ως *β*(*i,t*)=*m* αν $(i,j,t) \in S$ και $\mu^{\mathrm{i}}(j) = m$. Δείχνει ποια μηχανή χρησιμοποιείται από ποια εργασία σε δεδομένο χρόνο.

Στις προηγούμενες συναρτήσεις υπάρχει η πιθανότητα μια μηχανή ή μια εργασία να είναι ***αδρανής (idle)*** σε συγκεκριμένη χρονική στιγμή (χρησιμοποιούμε το σύμβολο $\perp$).

Συγκεκριμένα:

- μια μηχανή *m* είναι αδρανής σε χρόνο *t*, $\alpha(m,t) = \perp$ αν

$$\forall i, j \text{ s.t. } \mu^{\mathrm{i}}(j) = m \ \ (i,j,t \notin S$$

- μια εργασία *i* είναι αδρανής σε χρόνο *t*, $\beta(i,t) = \perp$ αν

$$\forall j \text{ s.t. } \mu^{i}(j) = m \ \ (i,j,t) \notin S$$

- το βήμα *j* μιας εργασίας *i* μπορεί να πραγματοποιηθεί σε χρόνο *t* αν

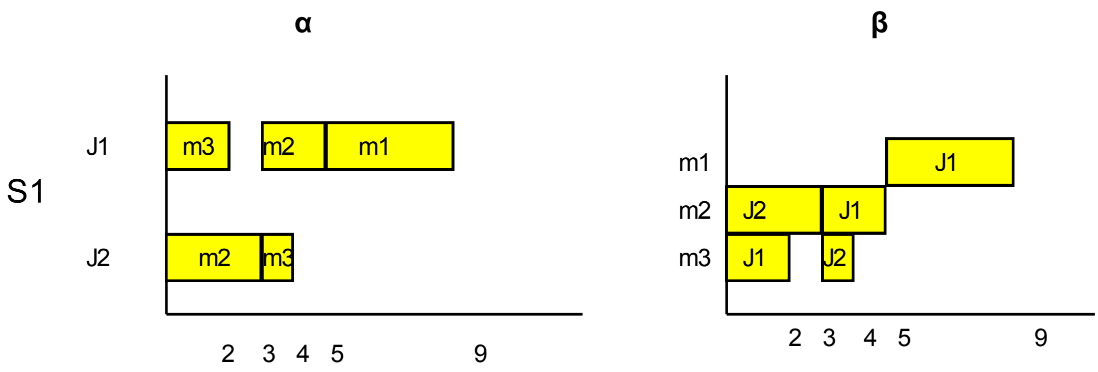
$$s(i,j-1)+d_i(j-1)<t<s(i,j) \text{ και } \alpha(\mathrm{m},\mathrm{t})=\perp$$

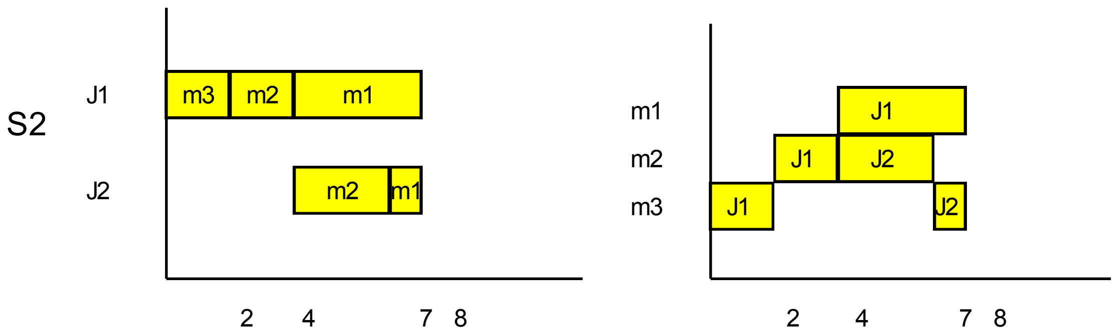
Παράδειγμα: Ας θεωρήσουμε ότι έχουμε ένα σύνολο τριών μηχανών $M=(m_1,m_2,m_3)$ και δύο εργασιών $J=\{J^1,J^2\}$ με χαρακτηριστικά

$J^1=(m_3,2),(m_2,2),(m_1,4)$ και

$J^2=(m_2,3),(m_3,1)$

Δύο πιθανά χρονοπρογράμματα απεικονίζονται στο Σχήμα 4.1. Η απεικόνιση γίνεται τόσο με βάση τη συνάρτηση κατανομής των μηχανών *α* όσο και με τη χρήση της συνάρτησης προόδου των εργασιών *β*. Η διάρκεια του S_1 είναι 9 χρονικές μονάδες, ενώ του S_2 είναι 8 χρονικές μονάδες.





Σχήμα 4.4: Τα δύο χρονοπρογράμματα S_1 κα S_2 χρησιμοποιώντας τις συναρτήσεις κατανομής των μηχανών και προόδου των εργασιών

Αξίζει να σημειωθεί ότι μια εργασία μπορεί να είναι αδρανής σε χρόνο *t* ακόμα και αν τόσο οι προηγούμενοι περιορισμοί που αναφέρθηκαν ικανοποιούνται όσο και αν η μηχανή που απαιτείται είναι διαθέσιμη. Χαρακτηριστικό παράδειγμα αποτελεί το χρονοπρόγραμμα S_2 όπου αν και η μηχανή m_2 είναι διαθέσιμη σε χρόνο *t*=0, η εργασία J^2 δεν εκτελείται σε αυτή τη μηχανή και παραμένει αδρανής μέχρι και το χρόνο *t*=4. Αν εκτελούνταν η J^2 στη m_2 αμέσως, τότε όπως φαίνεται στο Σχήμα 4.4 η J^1 θα άρχιζε να εκτελείται στη m_2 σε χρόνο *t*=3 και όχι *t*=2 και εν συνεχεία στη m_1 σε χρόνο *t*=5 και συνεπώς θα λαμβάναμε το χρονοπρόγραμμα S_1 με συνολικό χρόνο εκτέλεσης *t*=9 χρονικές μονάδες που είναι και μεγαλύτερος.

Η ανάγκη που υπάρχει να πετυχαίνουμε πάντα το βέλτιστο χρονοπρόγραμμα, περιμένοντας πολλές φορές αντί να ξεκινήσουμε αμέσως την εκτέλεση των εργασιών αυξάνει το σύνολο των πιθανών λύσεων που δύναται να προκύψουν και είναι προς διερεύνηση και στην πραγματικότητα αποτελεί την σημαντικότερη αιτία πολυπλοκότητας στον χρονοπρογραμματισμό παραγωγής.

4.3 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΜΕ ΤΗ ΧΡΗΣΗ ΧΡΟΝΙΣΜΕΝΩΝ ΑΥΤΟΜΑΤΩΝ

4.3.1 Μοντελοποίηση εργασιών

Κατασκευάζουμε για κάθε εργασία ξεχωριστά $J=(k,m,d)$ ένα χρονισμένο αυτόματο με ένα χρονιστή *c* και 2k+1 καταστάσεις. Στο κεφάλαιο 4 είπαμε ότι μια διαδικασία P μπορεί: α) να περιμένει πριν ξεκινήσει να εκτελείται $(\overline{p})$, β) να εκτελείται (*p*) και γ) να έχει ολοκληρωθεί αφού βρισκόταν σε κατάσταση εκτέλεσης $(\underline{p})$. Κάτι ανάλογο θεωρούμε και εδώ, όπου για κάθε βήμα *j* τέτοιο ώστε $\mu(j)=\mathrm{m}$ θα υπάρχουν δύο δυνατές καταστάσεις σε ένα χρονισμένο αυτόματο. Η κατάσταση $\overline{m}$ δείχνει ότι η εργασία αναμένει πριν ξεκινήσει η εκτέλεση του συγκεκριμένου βήματος και η κατάσταση *m* η οποία δείχνει ότι εκτελείται το δεδομένο βήμα της εργασίας. Η αρχική κατάσταση είναι η $\overline{\mu}(1)$ όπου η εργασία *J* δεν έχει ξεκινήσει ακόμα, ενώ η τελική κατάσταση είναι η *f* όπου η εργασία έχει ολοκληρώσει την εκτέλεση όλων των βημάτων. Στις καταστάσεις εκείνες στις οποίες οι εργασίες δεν έχουν ξεκινήσει την εκτέλεση των βημάτων τους ($\overline{m}$) οι αντίστοιχοι χρονιστές *c* είναι ακόμα ***μη ενεργοί (inactive)*** και μπαίνοντας σε διαδικασίες εκτέλεσης των βημάτων (*m*) μηδενίζονται. Το αυτόματο μπορεί να αφήσει την κατάσταση *m* μόνο αφού περάσει χρόνος $d(j)$ (διάρκεια βήματος) και αυτό γίνεται ελέγχοντας αν η τιμή του χρονιστή *c* είναι μεγαλύτερη ή ίση με $d(j)$.

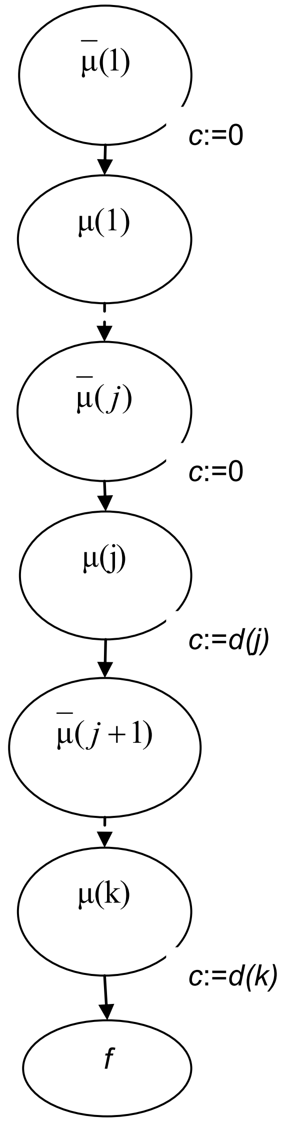
Έστω $\overline{M}=\{\overline{m}:m\in M\}$ και έστω $\overline{\mu}:\mathrm{K}\rightarrow\overline{\mathrm{M}}$ είναι μια βοηθητική συνάρτηση τέτοια ώστε $\overline{\mu}(j)=\overline{m}$ για οποιαδήποτε $\mu(j)=\mathrm{m}$.

Ορισμός 9 (Χρονισμένο αυτόματο για μια εργασία): Έστω μια εργασία $J=(k,m,d)$. Το σχετιζόμενο με την εργασία αυτή αυτόματο είναι $A=(Q,\{c\},\Delta,\mathrm{s},\mathrm{f})$ με $Q=P\cup\overline{P}\cup\{f\}$ όπου $P=\{\mu(1,\ldots,\mu(\mathrm{k})\}$ και $\overline{P}=\{\overline{\mu}(1),\ldots,\overline{\mu}(k)\}$. Η αρχική κατάσταση είναι $\overline{\mu}(1)$ ενώ η σχέση μετάβασης Δ περιλαμβάνει

$(\overline{\mu}(j),true,\mu(j))$ *j*=1,…,k

$(\overline{\mu}(j),c=d(j),\varnothing,\overline{\mu}(j+1))$ *j*=1,…,k+1

$(\mu(k),c=d(k),\varnothing,f)$



Σχήμα 4.5: Το αυτόματο που αντιστοιχεί στην εργασία $J=(k,m,d)$

4.3.2 Μοντελοποίηση των χαρακτηριστικών ενός Jop-shop συστήματος

Στη προηγούμενη ενότητα παρουσιάσαμε πως κατασκευάζεται ένα χρονισμένο αυτόματο για κάθε μια εργασία. Για να φτιάξουμε τώρα ένα χρονισμένο αυτόματο για ένα σύστημα Job-shop που μπορεί να περιλαμβάνει πολλές εργασίες θα πρέπει να συνθέσουμε τα αυτόματα των διαφόρων εργασιών. Στη προσπάθεια αυτή το στοιχείο που θα πρέπει να προσεχθεί ιδιαίτερα είναι να ισχύει η τρίτη συνθήκη που αναφέραμε στον ορισμό 9, δηλαδή να τηρείται ο περιορισμός της αμοιβαίας απόκλισης (δυο βήματα διαφορετικών εργασιών τα οποία εκτελούνται την ίδια χρονική στιγμή δεν χρησιμοποιούν την ίδια μηχανή). Με άλλα λόγια δεν θα πρέπει να υπάρχουν ***αλληλοσυγκρουόμενες καταστάσεις (conflicting states)*** όπου δύο ή περισσότερα αυτόματα βρίσκονται σε μια συγκεκριμένη κατάσταση στην οποία αντιστοιχεί ο ίδιος πόρος *m*. Παρακάτω δίνουμε τον ορισμό των αλληλοσυγκρουόμενων καταστάσεων.

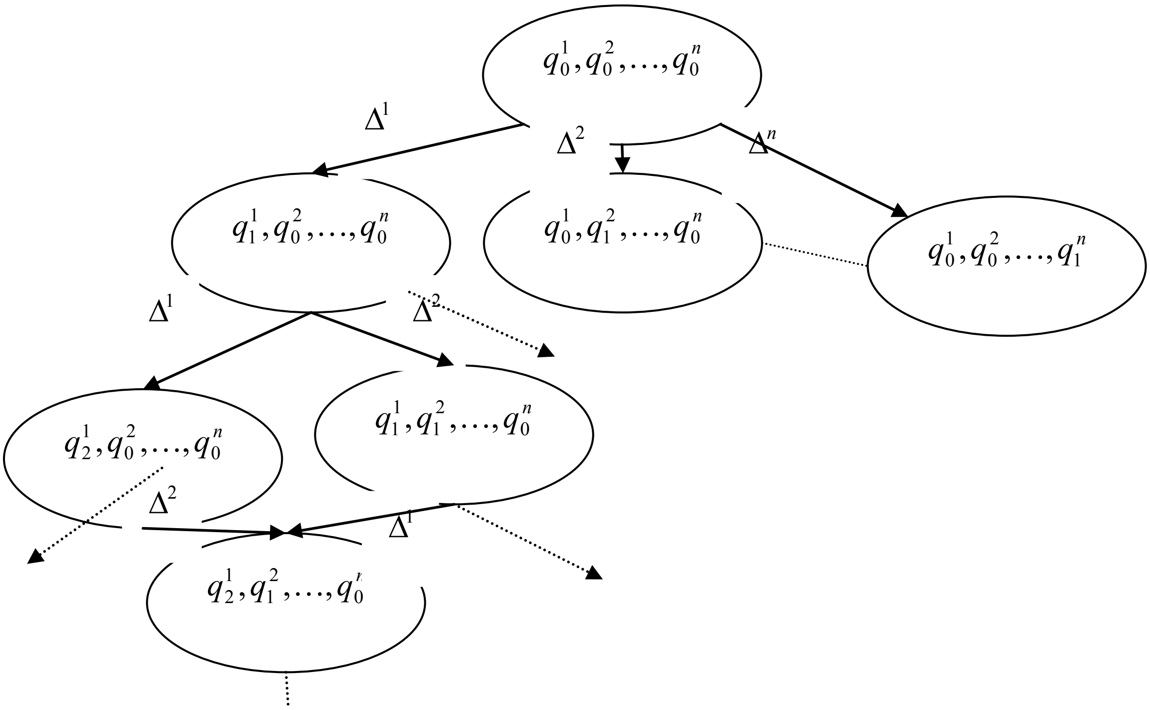
Ορισμός 10 (Αλληλοσυγκρουόμενες καταστάσεις): Μια κατάσταση $q=(q^1,\ldots,q^n)\in Q^n$ θεωρείται αλληλοσυγκρουόμενη αν περιλαμβάνει δυο στοιχεία q^α και q^β τέτοια ώστε $q^\alpha=q^\beta=m\in M$.

Έστω $J=\{J^1,\ldots,J^n\}$ είναι ένα Job-shop σύστημα και $A^i=(Q^i,\{c_i\},\Delta^i,s^i,f^i)$ είναι ένα χρονισμένο αυτόματο για μια συγκεκριμένη εργασία J^i. Αν συνθέσουμε όλα τα αυτόματα των εργασιών θα αποκτήσουμε το χρονισμένο αυτόματο $A=(Q,C,\Delta,\mathrm{s},\mathrm{f})$ που περιέχει *n*- χρονιστές.

Ορισμός 11 (Σύνθεση Αμοιβαίας Απόκλισης): Έστω $J=\{J^1,\ldots,J^n\}$ είναι ένα Job-shop σύστημα και έστω $A^i=(Q^i,\{c_i\},\Delta^i,s^i,f^i)$ είναι ένα αυτόματο που αντιστοιχεί στην εργασία J^i με *i*=1,2,…,*n*. Η σύνθεση αμοιβαίας απόκλισής του είναι το αυτόματο $A=(Q,C,\Delta,\mathrm{s},\mathrm{f})$ τέτοιο ώστε Q είναι ο περιορισμός των $Q^1\times Q^2\times\ldots Q^n$ μη αλληλοσυγκρουόμενων καταστάσεων, $C=C^1\cup C^2\cup\ldots C^m$, $s=(s^1,s^2,\ldots,s^n)$, $f=(f^1,f^2,\ldots,f^n)$ ενώ η σχέση μετάβασης περιλαμβάνει όλες τις πλειάδες της μορφής:

$$((q^1,q^2,\ldots,q^n),\varphi,\rho,(\mathrm{q}^1,q^2,\ldots,q^n))$$

τέτοιες ώστε $(q^{\alpha},\varphi,\rho,p^{a})\in\Delta^{\alpha}$ για κάποιο α και επιπλέον τόσο οι $(q^{1},q^{2},\ldots,q^{a},\ldots,q^{n})$ όσο και οι $(q^{1},q^{2},\ldots,p^{a},\ldots,q^{n})$ είναι μη αλληλοσυγκρουόμενες.



Σχήμα 4.6: Το χρονισμένο αυτόματο που αντιστοιχεί στα χαρακτηριστικά ενός Job-shop συστήματος $J=\{J^{1},\ldots,J^{n}\}$

Όπως μπορεί να διαπιστώσει κανείς από τον ορισμό και το Σχήμα 4.6 κάθε μεμονωμένη μετάβαση στο *A* αντιστοιχεί σε μια μετάβαση σε ***ένα*** και μόνο αυτόματο και το γεγονός αυτό ονομάζεται ***σημασιολογική παρεμβολή (interleaving semantics)***. Αν για παράδειγμα σε ένα χρονοπρόγραμμα δυο διαφορετικά αυτόματα πραγματοποιήσουν δυο μεταβάσεις δ^{1} και δ^{2} ταυτόχρονα τότε θα προκύψουν δύο αντίστοιχες εκτελέσεις στο αυτόματο και οι οποίες είναι:

$$q\xrightarrow{\delta_1}q'\xrightarrow{\delta_2}p \text{ και } q\xrightarrow{\delta_2}q''\xrightarrow{\delta_1}p \text{ με } q'\neq q''$$

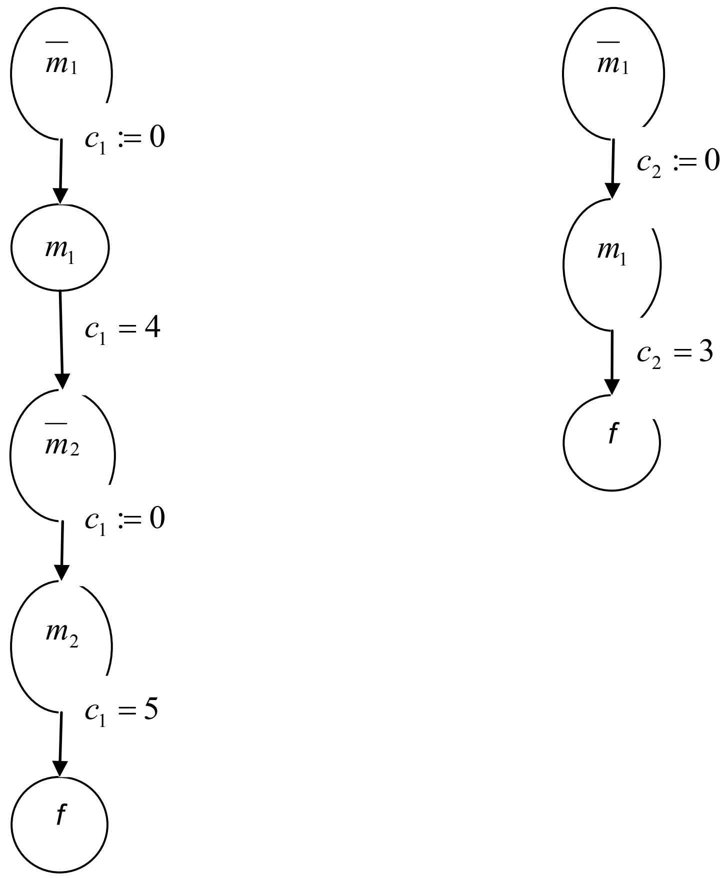
Παράδειγμα: Έστω ότι έχουμε δύο μηχανές $M=\{m_1,m_2\}$ και δύο εργασίες $J^{1}=(m_1,4),(m_2,5)$ και $J^{2}=(m_1,3)$.

Τα αυτόματα των δύο εργασιών απεικονίζονται στο Σχήμα 4.7 ενώ η σύνθεση και των δύο μαζί παρουσιάζεται στο Σχήμα 4.8.

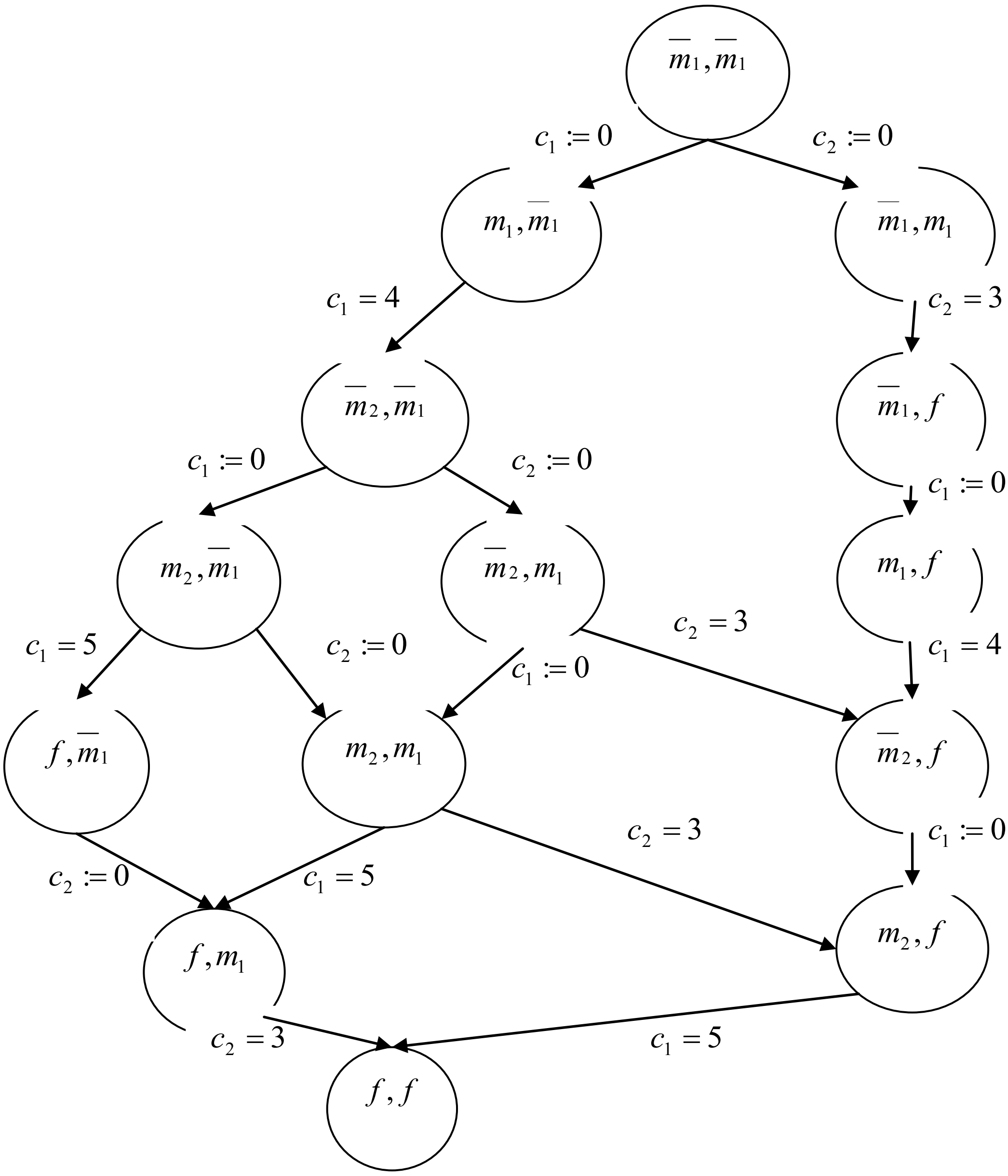
Το Job-shop χρονισμένο αυτόματο που προκύπτει είναι άκυκλο και σε σχήμα διαμαντιού. Επίσης περιλαμβάνει μια αρχική κατάσταση, στην οποία καμία από τις εργασίες δεν έχει ξεκινήσει και μια τελική κατάσταση όπου όλες οι εργασίες έχουν ολοκληρωθεί. Θα θεωρούμε ως ***ολοκληρωμένη (complete)*** την εκτέλεση ενός τέτοιου αυτομάτου αν ξεκινάει από την *s* (αρχική κατάσταση) και τελειώνει στην *f* (τελική κατάσταση).

Όλες οι μεταβάσεις σε ένα ολικό χρονισμένο αυτόματο δείχνουν ότι ένα στοιχείο (μηχανή ή εργασία) είτε μετακινείται από μια ενεργή κατάσταση σε μια μη ενεργή (υποβοηθείται από τη συνθήκη της μορφής $c_i = d$), όταν δηλαδή η τιμή του χρονιστή c_i είναι ίση με τη διάρκεια του βήματος *d* , είτε μετακινείται από μια μη ενεργή κατάσταση σε μια ενεργή (φαίνεται από την επαναφορά-μηδενισμό του χρονιστή $c_i = 0$ και ονομάζεται αρχική$_\iota$ *μετάβαση* (*$start_i$ transition*).

Δύο μεταβάσεις που εξέρχονται από την ίδια κατάσταση δείχνουν ότι ο χρονοπρογραμματιστής πρέπει να επιλέξει ποια από τις δύο θα ακολουθήσει. Συγκεκριμένα στο παράδειγμά μας, οι δύο μεταβάσεις που εξέρχονται από την αρχική κατάσταση $(\overline{m}_1, \overline{m}_1)$ αναπαριστούν την απόφασή μας για το ποια εργασία θα απασχολήσει πρώτη τη μηχανή m_1. Αν θα είναι δηλαδή η εργασία J^1, οπότε μετακινούμαστε στη κατάσταση $(m_1, \overline{m}_1)$, ή η εργασία J^2 με τη μετακίνηση στη κατάσταση $(\overline{m}_1, m_1)$. Επιπλέον ο χρονοπρογραμματιστής θα πρέπει να αποφασίσει αν μια εργασία θα ξεκινήσει ή θα μείνει αδρανής (idle). Στο παράδειγμά μας αυτό φαίνεται από τις δύο εξερχόμενες μεταβάσεις από την κατάσταση $(m_2, \overline{m}_1)$. Σε αυτή τη περίπτωση ο χρονοπρογραμματιστής είτε ξεκινάει την εργασία J^2 στη μηχανή m_1, είτε αφήνει το χρόνο να περάσει μέχρι ο χρονιστής c_1 να ικανοποιεί το βοηθό μετάβασης (guard) και να επιτευχθεί με αυτό τον τρόπο η μετακίνηση στη κατάσταση $(f, \overline{m}_1)$ που σημαίνει ότι η επεξεργασία της εργασίας J^1 στη μηχανή m_1 έχει ολοκληρωθεί. Από την άλλη μεριά μερικές επαναλήψεις στις διαδρομές που προκύπτουν είναι τεχνικώς γενόμενες εξαιτίας της παρεμβολής. Για παράδειγμα, οι δύο διαδρομές που εξέρχονται από την κατάσταση $(\overline{m}_2, \overline{m}_1)$ και φτάνουν στη κατάσταση (m_2, m_1), είναι μερικώς ισοδύναμες. Αξίζει να γίνει η υπενθύμιση ότι τα γραφήματα που απεικονίζουν μεταβάσεις μπορεί να είναι παραπλανητικά επειδή δύο ή περισσότερες μεταβάσεις που εισέρχονται στην ***ίδια*** διακριτή κατάσταση, πχ μεταβάσεις προς τη (m_2, f), μπορεί να εισέρχονται σε αυτή με διαφορετικές τιμές χρονιστών και έτσι να οδηγούν σε διαφορετικές διάρκειες (συνέχειες).



Σχήμα 4.7: Τα δύο αυτόματα που αντιστοιχούν στις εργασίες $J^1 = (m_1,4),(m_2,5)$ και $J^2 = (m_1,3)$



Σχήμα 4.8: Το ολικό χρονισμένο αυτόματο για τις δύο εργασίες J^1 και J^2

4.4 ΕΚΤΕΛΕΣΕΙΣ ΚΑΙ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΑ

Σε αυτή την ενότητα θα δείξουμε τη στενή σχέση που υπάρχει μεταξύ των εφικτών χρονοπρογραμμάτων και των εκτελέσεων ενός αυτομάτου.

Ορισμός 12 (Εξαγόμενα χρονοπρογράμματα): Το εξαγόμενο χρονοπρόγραμμα S_ξ μιας εκτέλεσης *ξ* είναι ένα χρονοπρόγραμμα όπου για κάθε *i,j* *s*(*i,j*)=*t* όπου *t* είναι ο συνολικός χρόνος στη *ξ* και το αυτόματο A^i πραγματοποιεί μια αρχική μετάβαση από τη $\overline{\mu}^{i}(j)$ στη $\mu^{i}(j)$.

Αξίωμα 1: Έστω *A* είναι ένα αυτόματο ενός Job-shop συστήματος με χαρακτηριστικά σύμφωνα με τους ορισμούς που δόθηκαν. Τότε:

1. Για κάθε ολοκληρωμένη εκτέλεση *ξ* του αυτομάτου *A*, το εξαγόμενο χρονοπρόγραμμα (S_ξ) είναι εφικτό για το *J*.
2. Για κάθε εφικτό χρονοπρόγραμμα *S* για το *J*, υπάρχει μια ολοκληρωμένη εκτέλεση *ξ* του αυτομάτου *A* τέτοια ώστε S_ξ=*S*.

Απόδειξη: Η απόδειξη γίνεται μέσω της επαγωγής πάνω στη διάρκεια της εκτέλεσης και του χρονοπρογράμματος. Ένα μερικό (partial) χρονοπρόγραμμα S' είναι ένα χρονοπρόγραμμα του *S* περιορισμένο όμως σε ένα διάστημα [0,*t*]. Μια επί μέρους εκτέλεση είναι μια εκτέλεση που δεν προσπελάζει την *f*. Το ***τμήμα*** του χρονοπρογράμματος σε χρόνο *t* εκφράζεται από την τετράδα $(\overline{P}, P, \underline{P}, e)$ όπου $\overline{P}$ είναι τα βήματα που αναμένεται να γίνουν, *P* είναι τα βήματα που πραγματοποιούνται τώρα και $\underline{P}$ είναι τα βήματα που έχουν ολοκληρωθεί. Το *e* είναι μια συνάρτηση από το *P* στο $\Re_+$ και δείχνει το χρόνο που έχει περάσει από την έναρξη κάθε ενεργής κατάστασης.

$$\overline{P} = \{(i,j) : s(i,j) > t\}$$

$$P = \{(i,j) : 0 \le t - s(i,j) \le d^i(j)\}$$

$$\underline{P} = \{(i,j) : d^i(j) \le t - s(i,j)\}$$

και *e*(*i,j*)=*t*-*s*(*i,j*). Στη συνέχεια θα ορίσουμε την αντιστοιχία που υπάρχει μεταξύ των σχηματισμών (configurations) ενός αυτομάτου και των τμημάτων ενός χρονοπρογράμματος. Έστω $((q_1, q_2, \ldots, q_n), (v_1, v_2, \ldots, v_n), t)$ είναι μια εκτεταμένη σύνθεση. Το σχετιζόμενό του τμήμα ορίζεται αν ορίσουμε για κάθε $i \in J$

$$q_i = \overline{\mu}^i(j) \quad \text{αν και μόνο αν} \quad \begin{array}{l} \forall j' < j(i,j) \in \underline{P} \wedge \\ \forall j' \geq j(i,j0 \in \overline{P} \end{array}$$

$$q_i = \overline{\mu}^i(j) \quad \text{αν και μόνο αν} \quad \begin{array}{l} \forall j' < j(i,j) \in \underline{P} \wedge \\ \forall j' > j(i,j) \in \overline{P} \wedge \\ (i,j) \in P \wedge \\ v_i = t - s(i,j) \end{array}$$

Η επαγωγική υπόθεση που γίνεται είναι ότι κάθε επί μέρους εκτέλεση που προσπελαύνει την κατάσταση (*q,v,t*) αντιστοιχεί σε ένα επί μέρους χρονοπρόγραμμα το οποίο είναι εφικτό μέχρι να περάσει χρόνος *t* και του οποίου το τμήμα ταιριάζει με την κατάσταση (*q,v,t*). Αυτό είναι ορθό σε αρχική κατάσταση και σε μηδενικό χρόνο (*t*=0) και μπορεί εύκολα να δειχθεί μέσω διακριτών μεταβάσεων και με το πέρασμα του χρόνου.

Πόρισμα:(Χρονοπρογραμματισμός Job- shop συστημάτων και χρονισμένα αυτόματα): Το πρόβλημα του άριστου χρονοπρογραμματισμού Job- shop συστημάτων μπορεί να περιοριστεί σε ένα πρόβλημα εύρεσης της συντομότερης διαδρομής σε ένα χρονισμένο αυτόματο.

4.5 ΕΥΡΕΣΗ ΤΗΣ ΣΥΝΤΟΜΟΤΕΡΗΣ ΔΙΑΔΡΟΜΗΣ ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ ΑΛΓΟΡΙΘΜΟΥΣ ΠΡΟΣΠΕΛΑΣΗΣ ΣΤΑ ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ

Στην ενότητα αυτή θα δείξουμε πως χρησιμοποιώντας τον αλγόριθμο ευθείας προσπέλασης μπορούμε να βρούμε τη συντομότερη διαδρομή και να επιλύσουμε προβλήματα που έχουν να κάνουν με τον χρονοπρογραμματισμό Job-shop συστημάτων.

Αν στο χρονισμένο αυτόματο *A* προσθέσουμε το χρονιστή (ρολόι) *t* που μετράει τον συνολικό χρόνο τότε θα έχουμε το διευρυμένο χρονισμένο αυτόματο A'. Αναλόγως από το A' θα προκύψουν δύο νέες διευρυμένες εκτελέσεις ξ_1' και ξ_2' που αντιστοιχούν στις παλαιές ξ_1 και ξ_2 και οι οποίες είναι:

$$\begin{aligned} &(\overline{m}_1,\overline{m}_1,\perp,\perp,0)\xrightarrow{0}(m_1,\overline{m}_1,0,\perp,0)\xrightarrow{4}(m_1,\overline{m}_1,4,0,4)\xrightarrow{0}\\ \xi_1':\ &(\overline{m}_2,\overline{m}_1,\perp,\perp,4)\xrightarrow{0}(m_2,\overline{m}_1,0,\perp,4)\xrightarrow{0}(m_2,m_1,0,0,4)\xrightarrow{3}\\ &(m_2,m_1,3,3,7)\xrightarrow{0}(m_2,f,3,\perp,7)\xrightarrow{2}(m_2,f,5,\perp,9)\xrightarrow{0}(f,f,\perp,\perp,9) \end{aligned}$$

$$\begin{aligned} &(\overline{m}_1,\overline{m}_1,\perp,\perp,0)\xrightarrow{0}(\overline{m}_1,m_1,\perp,0,0)\xrightarrow{3}(\overline{m}_1,m_1,\perp,3,3)\xrightarrow{0}\\ \xi_2':\ &(m_1,f,\perp,\perp,3)\xrightarrow{0}(m_1,f,0,\perp,3)\xrightarrow{4}(m_1,f,4,\perp,7)\xrightarrow{0}\\ &(\overline{m}_2,f,\perp,\perp,7)\xrightarrow{0}(m_2,f,0,\perp,7)\xrightarrow{5}(m_2,f,,5,\perp,12)\xrightarrow{0}(f,f,\perp,\perp,12) \end{aligned}$$

Η τιμή του επιπρόσθετου χρονιστή *t* σε κάθε προσπελάσιμο σχηματισμό δείχνει το συνολικό χρόνο που έχει περάσει μέχρι εκείνη τη στιγμή και έτσι στη τελική κατάσταση (*f,f*) φανερώνει τη συνολική διάρκεια της εκτέλεσης που στις συγκεκριμένες περιπτώσεις είναι 9 και 12 χρονικές μονάδες αντίστοιχα για τις ξ_1 και ξ_2.

Συνεπώς για να βρεθεί η συντομότερη διαδρομή, δηλαδή αυτή με τη μικρότερη συνολική διάρκεια, χρειάζεται να υπολογίσουμε και να συγκρίνουμε τις τιμές όλων των χρονιστών *t* όλων των προσπελάσιμων σχηματισμών που είναι της μορφής (*f,v,t*). Το σύνολο αυτό μπορεί να βρεθεί αν εφαρμοστεί ο αλγόριθμος ευθείας προσπέλασης για άκυκλα χρονισμένα αυτόματα που παρουσιάσαμε νωρίτερα.

Σημείωση: Επειδή στη προσπάθεια αυτή προκύπτουν πολλές συμβολικές καταστάσεις που πρέπει να διερευνηθούν για τη μείωση του αριθμού τους χρησιμοποιείται το inclusion test. Το test αυτό βασίζεται στο γεγονός ότι $Z \subseteq Z'$ που σημαίνει ότι $Succ^{\delta}(q,Z) \subseteq Succ^{\delta}(q,Z')$ για κάθε $\delta \in \Delta$. Έτσι όταν μια νέα συμβολική κατάσταση (q,Z) δημιουργείται, αυτή συγκρίνεται με οποιαδήποτε άλλη (q,Z') από αυτές που είναι στη λίστα προς εξερεύνηση. Αν $Z \subseteq Z'$ τότε η (q,Z) δεν εισέρχεται στη λίστα ενώ αν $Z' \subseteq Z$, η (q,Z') μετακινείται από τη λίστα. Επιτρέποντας στο Job- shop αυτόματο να μένει απεριόριστα σε κάθε κατάσταση κάνει τις προς εξερεύνηση ζώνες «κλειστές προς τα άνω» (“upward-closed”) αναφορικά με το συνολικό χρόνο και έτσι αυξάνεται σημαντικά η αποτελεσματικότητα του inclusion test.

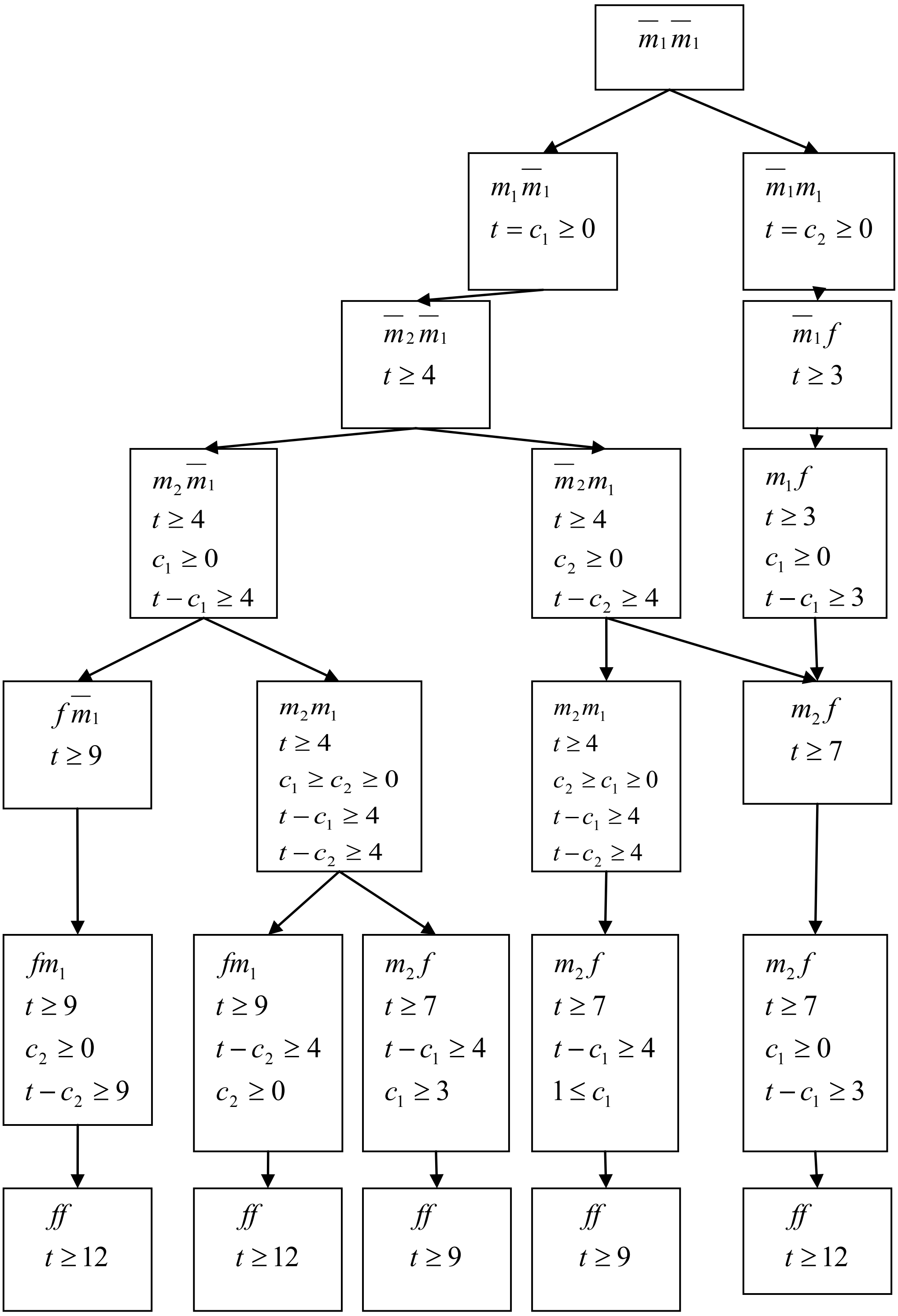
Από κάθε τελική συμβολική κατάσταση (f,Z) στο γράφημα προσομοίωσης μπορούμε να εξάγουμε τη συνάρτηση $G(f,Z)$ που δείχνει τη διάρκεια των ελαχίστων χρονικά εκτελέσεων δια μέσου όλων των εκτελέσεων που μοιράζονται την ίδια διαδρομή:

$$G(f,Z) = \min\{t : (v,t) \in Z\}$$

Η διάρκεια του βέλτιστου χρονοπρογράμματος είναι:

$$t^* = \min\{G(f,Z) : (f,Z) \textit{ που είναι προσπελάσιμες στο } A'\}$$

Το βέλτιστο χρονοπρόγραμμα είναι αυτό με διάρκεια t^* και για να το βρούμε θα εκτελέσουμε έναν αλγόριθμο προσπέλασης στο A'.



Σχήμα 4.9: Το γράφημα προσομοίωσης του διευρυμένου Job- shop χρονισμένου αυτομάτου του Σχήματος 4.8

ΚΕΦΑΛΑΙΟ 5

Η βιβλιοθήκη αλγορίθμων LiSA

5. Η ΒΙΒΛΙΟΘΗΚΗ ΑΛΓΟΡΙΘΜΩΝ LISA

5.1 ΕΙΣΑΓΩΓΗ

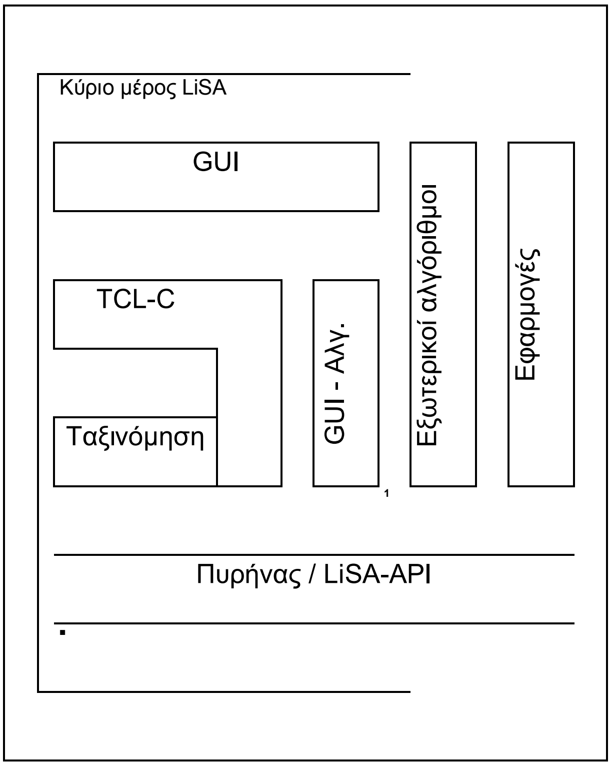
Η βιβλιοθήκη αλγορίθμων χρονοπρογραμματισμού LiSA αποτελεί επιστημονικό έργο μιας ερευνητικής προσπάθειας η οποία υποστηρίχθηκε από το Υπουργείο Πολιτισμού του ομόσπονδου κράτους της Σαξονίας-Άνχαλτ στο πλαίσιο των προγραμμάτων "Λατινικά Τετράγωνα στη Θεωρία Χρονοπρογραμματισμού" (Οκτώβριος 1997-Σεπτέμβριος 1999) και "LiSA - Μια βιβλιοθήκη αλγορίθμων χρονοπρογραμματισμού" (Νοέμβριος 1999 - Οκτώβριος 2001).

Το λογισμικό LiSA παρέχει ένα ολοκληρωμένο πλαίσιο για την επίλυση αιτιοκρατικών προβλημάτων χρονοπρογραμματισμού αξιοποιώντας ευρετικές και προσεγγιστικές μεθόδους. Στη συνέχεια περιγράφεται το τεχνικό υπόβαθρο του πακέτο και γίνεται μια σύντομη εισαγωγή στην ταξινόμηση και στα μοντέλα που χρησιμοποιούνται. Περιλαμβάνεται μια επισκόπηση των διαθέσιμων αλγορίθμων, οι οποίοι μπορούν επίσης να χρησιμοποιηθούν κι εκτός του LiSA. Τέλος δίνεται ένα χαρακτηριστικό παράδειγμα για την κατανόηση της λειτουργίας του LISA και παράλληλα επεξηγείται η δομή του και πώς μπορούν να εισαχθούν νέοι αλγόριθμοι στο πακέτο. (http://lisa.math.uni-magdeburg.de/index.php)

5.2 ΤΙ ΕΙΝΑΙ ΤΟ LISA

Το LiSA είναι ένα πακέτο λογισμικού για την επίλυση αιτιοκρατικών προβλημάτων χρονοπρογραμματισμού. Σε ένα πρόβλημα χρονοπρογραμματισμού ένα σύνολο εργασιών θα πρέπει να υποβληθεί σε επεξεργασία σε ένα σύνολο μηχανών κάτω από διαφορετικούς πρόσθετους περιορισμούς, έτσι ώστε να ελαχιστοποιηθεί μια αντικειμενική συνάρτηση. Στη βιβλιογραφία ένα τέτοιο πρόβλημα χρονοπρογραμματισμού δίνεται συνήθως από την τριάδα α | β | γ, όπου το α περιγράφει το περιβάλλον μηχανής, το β δίνει τους πρόσθετους περιορισμούς και το γ αντιπροσωπεύει την αντικειμενική συνάρτηση.

Το λογισμικό LiSA έχει αρθρωτή δομή. Το κύριο μέρος του LiSA περιέχει όλους τους βασικούς αλγόριθμους που συνδέονται με το μοντέλο, τις διαδικασίες εισαγωγής και εξαγωγής και τις μονάδες του γραφικού περιβάλλοντος εργασίας χρήστη. Επιπλέον, εδώ θα διαχειρίζονται τα στοιχεία του προγράμματος και θα συντονίζεται η συνεργασία με εξωτερικούς αλγόριθμους. Μπορεί επίσης να καθοριστεί η κατάσταση πολυπλοκότητας ενός προβλήματος στα σύμβολα α | β | γ. Όλοι οι αλγόριθμοι που χρησιμοποιούνται στο LiSA για την επίλυση προβλημάτων χρονοπρογραμματισμού ενσωματώνονται σε εξωτερικές μονάδες. Οι μονάδες αυτές είναι αυτόνομα δυαδικά στοιχεία, με μια κοινή διασύνδεση γραμμής εντολών. Επικοινωνούν με το κύριο πρόγραμμα μέσω αρχείων. Έτσι, μπορούν να χρησιμοποιηθούν τόσο μέσα από το GUI του LiSA όσο και ανεξάρτητα, χωρίς GUI σε κατάσταση λειτουργίας δέσμης. Η εισαγωγή νέου αλγόριθμου στο LiSA πραγματοποιείται μέσω ενός αρχείου περιγραφής αλγόριθμου και του αντίστοιχου αρχείου βοήθειας σε μορφή html.



Σχήμα 1: Δομή LiSA

5.2 ΑΛΓΟΡΙΘΜΟΙ LISA

5.2.1 Αλγόριθμοι Γενικής Χρήσης

- Branch-and-Bound

Ένας αλγόριθμος Branch-and-Bound (κλάδος και όριο) για συνήθεις αντικειμενικές συναρτήσεις επιλύει τα περισσότερα προβλήματα σε συστήματα παραγωγής open, job ή flow shop. Ο αλγόριθμός μας εφαρμόζει μία τεχνική εισαγωγής, δείτε BRASEL και BRASEL ET AL. και παρεχόμενα από το χρήστη επάνω όριο και κάτω όρια. Το επάνω όριο μπορεί επίσης να προκύψει από προηγουμένως εφαρμοσμένες ευρετικές μεθόδους. Με δεδομένη την ύπαρξη ενός σωστού συνόλου ορίων, ο αλγόριθμος μπορεί να επιταχυνθεί σημαντικά.

-Εποικοδομητικοί ευρετικοί αλγόριθμοι

Μπορούν να χρησιμοποιηθούν κανόνες απόδοσης προτεραιότητας για διάφορους σκοπούς και πολλούς πρόσθετους περιορισμούς. Βήμα προς βήμα προστίθεται μια νέα λειτουργία με τη βοήθεια μιας δεδομένης στρατηγικής.

Κανόνας	Στρατηγική επιλογής της επόμενης λειτουργίας
RAND	τυχαία
FCFS	εξυπηρέτηση με βάση τη σειρά άφιξης στο σύστημα
EDD	νωρίτερη ημερομηνίας περάτωσης πρώτα
LQUE	με μικρότερη διαφορά ημερομηνίας περάτωσης και (χρόνου επεξεργασίας + ουράς)
SPT	συντομότερος χρόνος επεξεργασίας πρώτα
WSPT	συντομότερος σταθμισμένος χρόνος επεξεργασίας πρώτα
ECT	(εφικτός) χρόνος νωρίτερης ολοκλήρωσης πρώτα
WI	μεγαλύτερο βάρος πρώτα
LPT	μεγαλύτερος χρόνος επεξεργασίας πρώτα

Κανόνες απόδοσης προτεραιότητας

-Επαναληπτικοί αλγόριθμοι

Στο LiSA υπάρχουν πολλοί αλγόριθμοι αναζήτησης γειτονιάς. Κάθε κορυφή ενός γραφήματος γειτονιάς αντιστοιχεί σε μια λύση (ακολουθία και χρονοπρόγραμμα) και σταθμίζεται ως προς την τιμή αντικειμενικής συνάρτησης του εξεταζόμενου προβλήματος. Το σύνολο τόξων ή ακμών είναι διαφορετικό σε πολλές γειτονιές. Χρειαζόμαστε μια πρώτη λύση, η οποία μπορεί να βρεθεί με μια απλή εποικοδομητική ευρετική μέθοδο. Στη συνέχεια ξεκινούμε μια βηματική περιήγηση (επαναληπτική αναζήτηση) στο γράφημα γειτονιάς για να βρούμε μια καλύτερη λύση.

Μέθοδος	Περιγραφή
Επαναληπτική βελτίωση	μετάβαση σε μια καλύτερη γειτονική λύση - μετά την απαρίθμηση όλων των γειτόνων ή - εάν βρεθεί το πρώτο καλύτερο χρονοπρόγραμμα στη γειτονιά
Προσομοιωμένη ανόπτηση	επιτρέπει τη μετάβαση σε χειρότερη λύση με συγκεκριμένη πιθανότητα, η οποία μειώνεται βήμα προς βήμα
Αποδοχή ορίου	επιτρέπει τη μετάβαση σε έναν χειρότερο γείτονα με ένα όριο, το οποίο σταδιακά μειώνεται
Απαγορευμένη αναζήτηση	δημιουργεί απαγορευμένες λίστες, για την αποφυγή επιστροφής σε λύσεις που έχουν ήδη προσπελαστεί

Στρατηγικές αναζήτησης γειτονιών

Στον πίνακα 2 εξηγούνται οι διάφορες μέθοδοι επαναληπτικής αναζήτησης που εφαρμόζονται. Ο πίνακας 3 περιλαμβάνει τις γειτονιές που είναι διαθέσιμες στο LISA.

Γειτονιά	Περιγραφή
API	εναλλάσσει δύο (άμεσα) γειτονικές λειτουργίες σε μια μηχανή
SHIFT	μετατοπίζει αυθαίρετα μια λειτουργία σε μια μηχανή
CR_API	εναλλάσσει δύο γειτονικές λειτουργίες σε μια μηχανή, κρίσιμες για τον στόχο C_{max}
3_CR	επεκτείνει το CR_ API, αντιμεταθέτει προκάτοχο και διάδοχο, καθώς και άλλες λειτουργίες
BL_SHIFT	μετατοπίζει ένα μπλοκ λειτουργιών σε μια μηχανή, κρίσιμο για το στόχο C_{max}
BL_API CR_SHIFT	εναλλάσσουν λειτουργίες σε μια μηχανή κατά τέτοιο τρόπο ώστε να καταστρέφεται μία κρίσιμη διαδρομή του χρονοπρογράμματος (ιδέα προσέγγισης μπλοκ)

Γειτονιές

5.2.2 Ειδικοί Αλγόριθμοι

Μερικοί αλγόριθμοι του λογισμικού LISA έχουν σχεδιαστεί για ειδικές, σαφώς καθορισμένες κατηγορίες προβλημάτων.
Ο πρώτος πίνακας που ακολουθεί περιέχει ακριβείς αλγόριθμους, ενώ ο δεύτερος μια επισκόπηση των ευρετικών αλγορίθμων.

Τύπος προβλήματος	Περιγραφή
$1\|\| L_{max}$	Κλάδος και όριο
$1\|\| \sum w_i T_i$	Ο κανόνας Smith, λειτουργεί όπως ο κανόνας WSPT
$1\|\| L_{max}$	Κανόνας ERD, νωρίτερη ημερομηνία αποδέσμευσης πρώτα
$F2\|\| C_{max}$	Κανόνας Johnson
$J2\|\| C_{max}$	Κανόνας Jackson
$O2\|\| C_{max}$	Αλγόριθμος Gonzales / Sahni
$P2\|\| C_{max}$	Ψευδοπολυωνυμικός αλγόριθμος (δεν υπάρχει ακόμη απεικόνιση)
$J\|\| C_{max}$	Αρχικός αλγόριθμος κλάδου και ορίου Brucker
$O\|pmtn\| C_{max}$	Gonzales / Sahni (δεν υπάρχει ακόμη απεικόνιση)
$O2\|\| C_{max}$	Κανόνας LAPT, μεγαλύτερος χρόνος εναλλακτικής επεξεργασίας - επεξεργάζεται την εργασία με τον μεγαλύτερο χρόνο επεξεργασίας στην άλλη μηχανή πρώτα

Ακριβείς αλγόριθμοι για ειδικά προβλήματα

Τύπος προβλήματος	Περιγραφή	Λογοτεχνία
$F\|\| C_{max}$	Ευρετική μέθοδος συστήματος παραγωγής συνεχούς ροής Dannenbring	[8]
$J\|\| C_{max}$	Ευρετική μέθοδος μετατόπισης συμφόρησης	[11]
$O\|\| C_{max}$ and $O\|\| \sum C_i$	Αντιστοίχιση ευρετικών μεθόδων	[4]
$O\|\| C_{max}$ and $F\|\| C_{max}$	Αναζήτηση δέσμης	[4]

Ευρετικές μέθοδοι για ειδικά προβλήματα

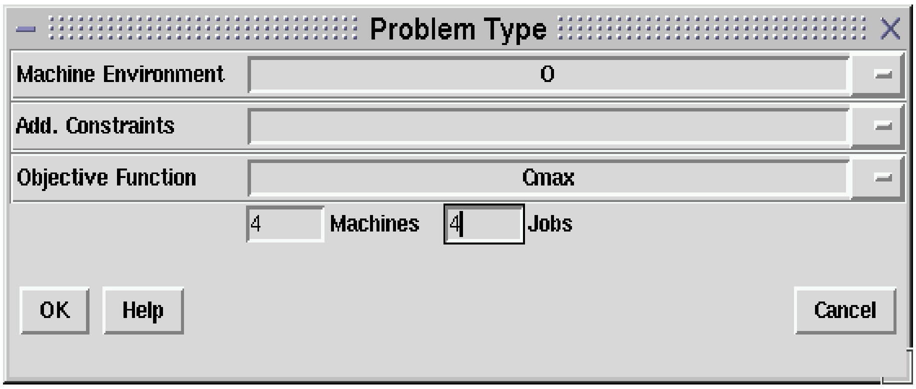
Σημειώνεται ότι στην έκδοση 2.3 του LiSA η οποία χρησιμοποιήθηκε και στο πλαίσιο της παρούσας εργασίας εξακολουθούν να παραμένουν άλυτα προβλήματα χρονοπρογραμματισμού με περιορισμούς προτεραιότητας ενώ απουσιάζουν βασικοί αλγόριθμοι για απεικόνιση προβλημάτων αναστολής και αποτελεσματικές δομές στοιχείων για την απεικόνιση προβλημάτων παράλληλων μηχανών.

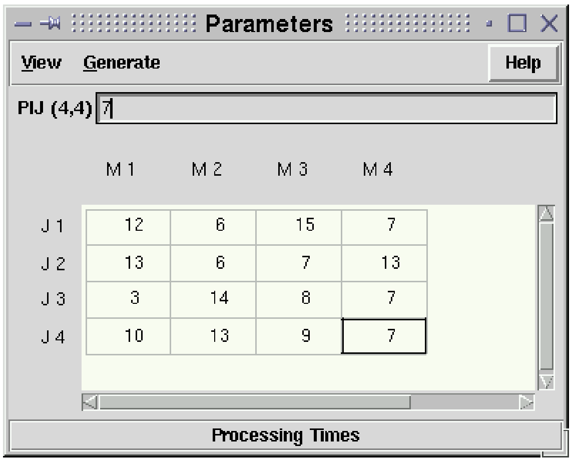
5.3 ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ LISA (ΠΑΡΑΔΕΙΓΜΑ)

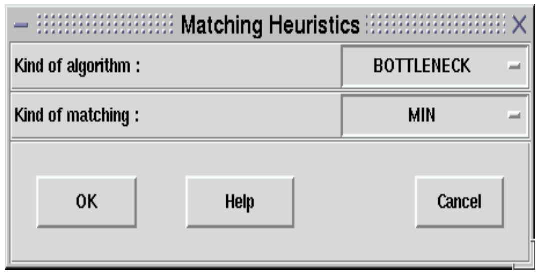
Υποθέτουμε ένα πρόβλημα ανοικτού παραγωγικού συστήματος με m = 4 μηχανές, n = 4 εργασίες και ελαχιστοποίηση του συνολικού χρόνου ολοκλήρωσης χωρίς πρόσθετο περιορισμό. Ο χρόνος επεξεργασίας δίνεται από τον ακόλουθο πίνακα *PT:*

$$PT = \begin{vmatrix} 12 & 6 & 15 & 7 \\ 13 & 6 & 7 & 13 \\ 3 & 14 & 8 & 7 \\ 10 & 13 & 9 & 7 \end{vmatrix}$$

Ποια βήματα απαιτούνται; Μετά την εκκίνηση του LiSA επιλέγουμε από το μενού File [Αρχείο]το κουμπί New [Νέο]. Ανοίγει το παράθυρο Problem Type [Τύπος προβλήματος] και εισάγετε το πρόβλημά σας στη δήλωση α | β | γ , δηλαδή το O || Cmax. Μην ξεχάσετε να εισάγετε τα n = 4 και m = 4. Τώρα το λογισμικό LiSA μας παρέχει όλες τις μονάδες για το υπό εξέταση πρόβλημα. Ξεκινάμε με την εισαγωγή του χρόνου επεξεργασίας (κουμπιά Edit [Επεξεργασία], Parameter [Παράμετρος], Generate [Δημιουργία], Processing times [Χρόνος επεξεργασίας]). Μπορείτε να το κάνετε με το χέρι ή χρησιμοποιώντας τη δυνατότητα τυχαίας δημιουργίας.

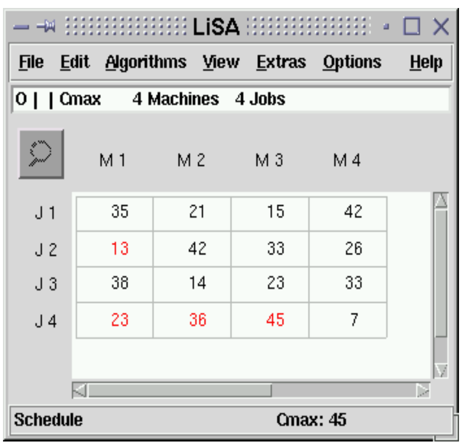


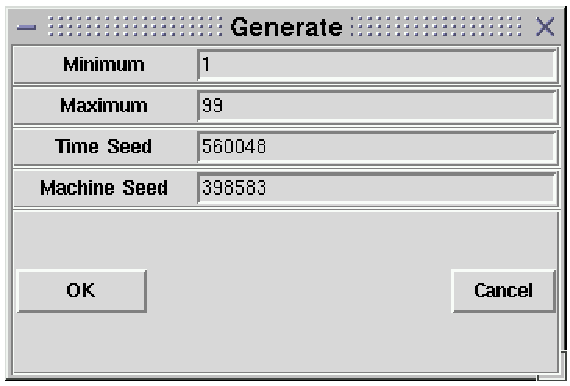


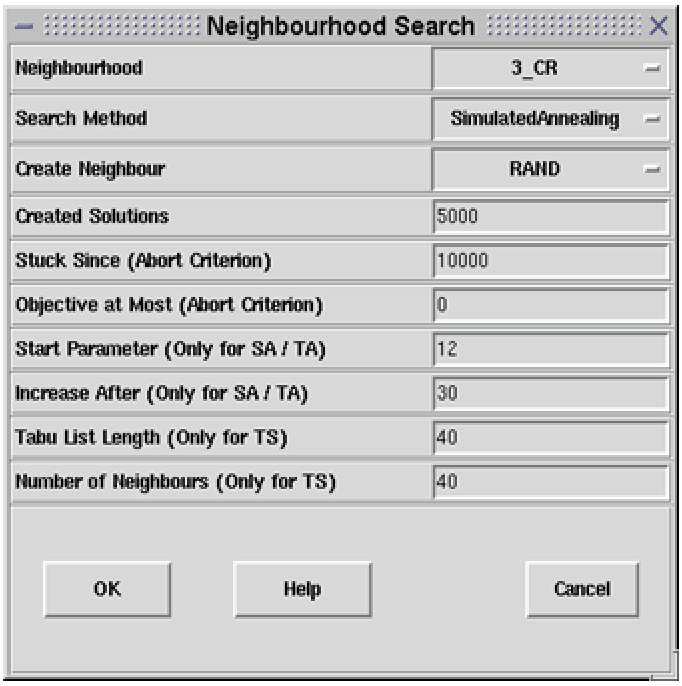


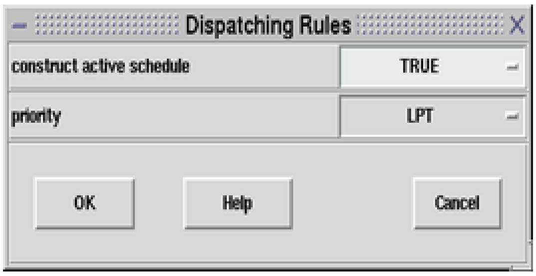
Σχήμα 2: Εισαγωγή του παραδείγματος

Στο μενού **Algorithms [**Αλγόριθμοι**]** υπάρχουν ακριβείς και ευρετικοί αλγόριθμοι για το συγκεκριμένο πρόβλημα. Το σχήμα 3 δείχνει κάποιες δυνατότητες. Για παράδειγμα, μπορεί να εφαρμοστεί ο κανόνας LPT (μεγαλύτερος χρόνος επεξεργασίας πρώτα). Μετά την κατασκευή του πρώτου χρονοπρογράμματος είμαστε σε θέση να χρησιμοποιήσουμε όλους τους περιεχόμενους αλγόριθμους αναζήτησης γειτονιάς, εδώ επιλέγεται η προσομοιωμένη ανόπτηση με γειτονιά 3_CR. Ορισμένες παράμετροι μπορούν να αλλάξουν, ακολουθήστε την περιγραφή του LiSA. Τέλος, υπάρχουν οι αντίστοιχες ευρετικές μέθοδοι για το πρόβλημά σας: βήμα προς βήμα όλες οι λειτουργίες που ανήκουν σε μια βέλτιστη αντιστοίχιση σε σχέση με τον πίνακα χρόνου επεξεργασίας μπορούν να επεξεργαστούν ταυτόχρονα, εδώ με αντικειμενική συνάρτηση ελάχιστης συμφόρησης.









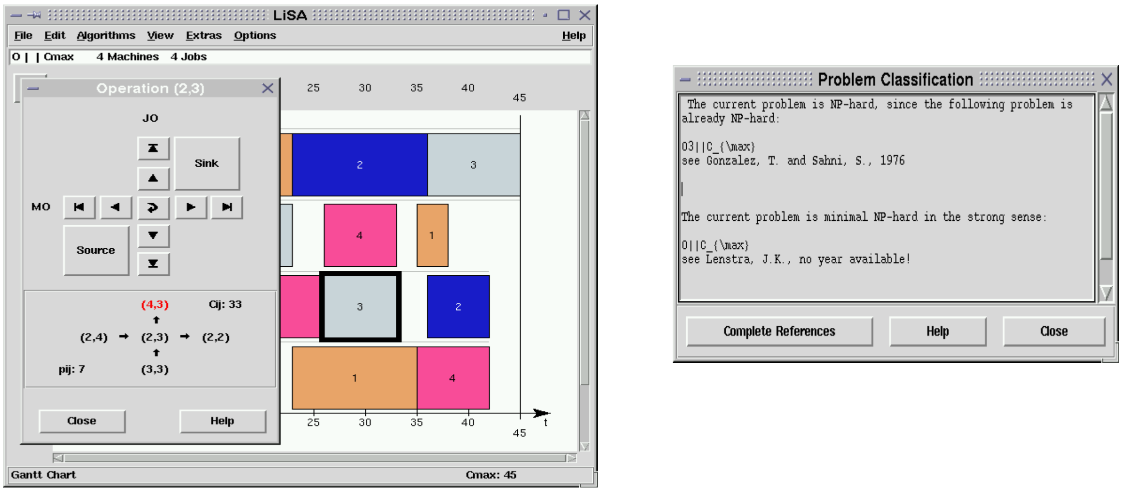
Σχήμα 3: Ευρετικοί αλγόριθμοι για το υπό εξέταση πρόβλημα

Συνήθως το αίτημα ενός αλγορίθμου παράγει το διάγραμμα Gantt του δημιουργούμενου χρονοπρογράμματος, αλλά υπάρχουν και άλλες δυνατότητες παραγωγής, δείτε το σχήμα 4. Εδώ μπορείτε να επιλέξετε από το μενού View [Προβολή] τον πίνακα ακολουθίας ή το γράφημα ακολουθίας, το χρονοπρόγραμμα περιγράφεται ως πίνακας χρόνου ολοκλήρωσης ή ως διάγραμμα Gantt. Σημειώστε ότι στο μενού Options [Επιλογές] μπορούν να επιλεγούν κάποιες επιλογές του διαγράμματος Gantt, μπορεί, για παράδειγμα, αυτό να είναι προσανατολισμένο στη μηχανή ή προσανατολισμένο στην εργασία ή μπορεί να επισημανθεί η κρίσιμη διαδρομή. Επιπλέον, μπορούν να αλλάξουν ορισμένα χρώματα για τις λειτουργίες. Εάν ο αριθμός των εργασιών ή των μηχανών είναι μεγάλος, δηλαδή το διάγραμμα Gantt είναι υπερβολικά πολύπλοκο, βοηθά η χρήση του ζουμ.

Το LiSA διαθέτει κάποιες πρόσθετες δυνατότητες, δύο από αυτές περιλαμβάνονται στο σχήμα 5. Η πιο σημαντική πρόσθετη δυνατότητα είναι η μονάδα πολυπλοκότητας. Κάθε φορά που το LiSA έχει τη δήλωση α | β | γ ενός προβλήματος καθορίζει την κατάσταση πολυπλοκότητας (κουμπιά Extras [Πρόσθετα], Problem Classification, [Ταξινόμηση προβλήματος]).

Σχήμα 4: Εξαγωγή LiSA

Επιπλέον, ο χρήστης μπορεί επίσης να λάβει μια πλήρη αναφορά πατώντας το αντίστοιχο κουμπί. Μια άλλη πρόσθετη δυνατότητα είναι η μονάδα χειρισμού. Αν θέλετε να χειριστείτε το χρονοπρόγραμμα μπορείτε να το κάνετε πατώντας το ποντίκι (δεξιά) σε μια ορισμένη λειτουργία του διαγράμματος Gantt. Αυτή η λειτουργία μπορεί να μετακινηθεί κατά μία θέση νωρίτερα ή αργότερα και στα δύο, στη σειρά μηχανών ή στη σειρά εργασιών, αντίστοιχα. Το LiSA ειδοποιεί, εάν ο νέος συνδυασμός σειρών μηχανών και σειρών εργασιών δεν είναι εφικτός. Κάθε μετακίνηση μιας λειτουργίας σε έναν αφετηριακό κόμβο ή σε ένα τερματικό κόμβο του γραφήματος ακολουθίας παράγει και πάλι μια εφικτή ακολουθία και παρουσιάζεται το αντίστοιχο διάγραμμα Gantt. (http://lisa.math.uni-magdeburg.de/index.php)



Σχήμα 5: Πρόσθετα του LiSA

ΚΕΦΑΛΑΙΟ 6

Συγκριτική Επίλυση Προβλημάτων Χρονοπρογραμματισμού

6. ΣΥΓΚΡΙΤΙΚΗ ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

6.1 ΕΙΣΑΓΩΓΗ

Στόχος της εργασίας ήταν εξαρχής να γίνει μια σύγκριση των αποτελεσμάτων της επίλυσης προβλημάτων χρονοπρογραμματισμού με χρονισμένα αυτόματα και με προσεγγιστικούς και ευρετικούς και αλγορίθμους. Για αυτό το σκοπό δημιουργήσαμε 100 προβλήματα παρόμοιας δομής τα οποία επιλύσαμε και με τους δύο τρόπους. Σκοπός μας είναι να εξάγουμε χρήσιμα συμπεράσματα για το ποιες μέθοδοι και με ποιες προτεραιότητες είναι αποτελεσματικότερες και θα μας οδηγήσουν στη βέλτιστη λύση. Εκτός αυτού, με την επίλυση των προβλημάτων ίσως εκμαιεύσουμε πληροφορίες όσον αφορά μειονεκτήματα και πλεονεκτήματα των μεθόδων προκειμένου στο μέλλον να οδηγηθούμε σε πιο πλήρες μοντέλο επίλυσης που να προκύπτει από πιθανό συνδυασμό τους, εκμεταλλευόμενοι τα δυνατά σημεία της κάθε μεθόδου.

6.2 ΜΟΡΦΗ ΠΡΟΒΛΗΜΑΤΩΝ

Η γενική μορφή των προβλημάτων που θα επιλυθούν είναι προβλήματα χρονοπρογραμματισμού με συντελεστές 10 μηχανές και 10 εργασίες (Jobshop problems 10x10). Έχουμε διαμορφώσει 5 σετ προβλημάτων σε κάθε ένα από τα οποία η σειρά των εργασιών στις μηχανές είναι σταθερή όπως φαίνεται στους πίνακες που ακολουθούν. Ο χρόνος της εργασίας στη κάθε μηχανή διαφέρει σε όλες τις δοκιμές και προσδιορίστηκαν από μια random συνάρτηση επιλογής αριθμών από 1 έως 9. Έτσι προέκυψαν τελικά 100 διαφορετικά προβλήματα που επιλύθηκαν τόσο με τη χρήση του εργαλείου UPPAAL (χρονισμένα αυτόματα) όσο και με τη χρήση του προγράμματος LISA

Παρακάτω παρατίθενται οι πίνακες γενικής μορφής προβλήματος για το κάθε σετ. Συγκεκριμένα σε κάθε πίνακα αναφέρεται ως επικεφαλίδα ο αριθμός του σετ προβλημάτων που έχει τη δεδομένη σειρά εργασιών στις μηχανές. Στον οριζόντιο άξονα αναγράφεται από το 0 έως το 9 ο αριθμός της μηχανής που πραγματοποιείται η κάθε εργασία, ενώ στον κάθετο άξονα αναγράφονται από το 0 έως το 9 οι 10 διαφορετικές εργασίες. Στο εσωτερικό του πίνακα αναγράφεται τέλος η σειρά της συγκεκριμένης μηχανής στο συνολικό κύκλο ολοκλήρωσης της συγκεκριμένης εργασίας.

SET1										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	1	9	4	8	2	5	7	10	6	3
J1	3	5	2	6	8	1	4	7	10	9
J2	3	10	6	1	2	5	7	9	4	8
J3	3	6	5	7	9	10	2	8	4	1
J4	4	1	6	3	7	8	10	9	2	5
J5	4	5	2	1	6	7	8	9	10	3
J6	9	6	2	1	3	4	7	10	8	5
J7	1	4	2	5	3	6	7	8	9	10
J8	8	2	5	6	3	1	4	7	9	10
J9	1	7	8	4	9	10	2	3	5	6

SET2										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	6	4	7	8	1	5	10	3	2	9
J1	3	4	1	2	5	7	9	10	6	8
J2	8	9	10	4	6	5	3	7	2	1
J3	10	6	8	9	1	2	4	3	5	7
J4	3	2	6	9	1	8	5	4	7	10
J5	5	9	1	4	7	8	10	3	6	2
J6	2	4	7	8	9	1	3	5	6	10
J7	3	4	5	9	10	8	1	7	2	6
J8	3	1	9	4	5	6	8	2	7	10
J9	7	1	2	3	8	9	10	4	6	5

SET3										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	3	7	8	9	1	10	5	2	4	6
J1	3	4	6	8	9	5	2	10	7	1
J2	2	4	5	6	8	9	10	7	1	3
J3	8	9	10	7	3	4	2	5	6	1
J4	7	6	8	10	9	2	5	4	3	1
J5	1	2	3	4	5	6	7	9	8	10
J6	1	10	7	8	6	4	9	3	5	2
J7	2	1	5	10	8	3	4	6	9	7
J8	1	4	5	2	6	7	3	8	9	10
J9	4	7	1	3	6	8	5	2	9	10

SET4										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	9	10	7	4	2	1	3	6	5	8
J1	1	7	8	9	10	5	2	3	4	6
J2	1	3	6	7	2	8	9	4	5	10
J3	2	3	7	8	6	4	5	1	9	10
J4	9	10	3	7	6	1	4	5	2	8
J5	5	8	7	9	6	10	4	3	1	2
J6	2	1	3	6	4	7	8	9	10	5
J7	8	9	4	1	2	6	7	3	5	10
J8	2	10	3	4	1	6	5	7	8	9
J9	3	5	4	7	9	2	10	8	1	6

SET5										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	10	4	5	6	1	2	3	7	8	9
J1	1	7	4	5	8	2	9	3	6	10
J2	7	8	10	2	9	4	3	5	6	1
J3	3	4	5	6	7	8	9	10	1	2
J4	7	4	10	2	9	1	8	3	6	5
J5	7	5	6	3	2	8	1	9	4	10
J6	1	2	3	4	5	6	7	9	10	8
J7	4	2	5	8	6	10	7	9	3	1
J8	6	7	3	8	9	10	2	4	5	1
J9	1	9	3	7	8	10	4	5	6	2

Στους παρακάτω πίνακες παρουσιάζονται οι άλλες παράμετροι των προβλημάτων μας, δηλαδή οι χρόνοι επεξεργασίας των εργασιών στη κάθε μηχανή. Όπως εξηγήσαμε νωρίτερα οι χρόνοι αυτοί προέκυψαν από μια random συνάρτηση επιλογής αριθμών από το 0 έως το 9. Όπως και πριν, η επικεφαλίδα αναφέρει τον αριθμό της δοκιμής αλλά και το σετ στο οποίο ανήκει. Ο οριζόντιος άξονας αναφέρει τον αριθμό των μηχανών από το 0 έως το 9, ενώ ο κάθετος τις αντίστοιχες εργασίες. Στο εσωτερικό του πίνακα αναφέρεται ο χρόνος επεξεργασίας της συγκεκριμένης εργασίας στην αντίστοιχη μηχανή. Εδώ ενδεικτικά παρουσιάζονται οι δοκιμές 1 έως 10 από το δεύτερο σετ.

SET 2-1										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	4	2	8	1	9	8	4	8	9	2
J1	8	1	8	7	1	6	5	8	4	7
J2	9	2	3	7	7	3	9	6	3	9
J3	2	8	9	2	5	5	2	3	8	3
J4	3	7	4	3	8	6	7	8	1	7
J5	5	3	5	3	3	5	4	3	7	3
J6	5	5	4	7	9	9	8	6	5	3
J7	2	2	7	7	7	2	2	3	8	8
J8	5	1	6	3	9	6	5	6	3	1
J9	2	4	8	8	3	5	3	5	7	1

SET 2-2										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	6	4	6	8	1	3	3	5	8	3
J1	2	8	5	8	6	1	4	5	1	1
J2	1	4	2	7	1	1	6	4	2	7
J3	3	2	5	5	4	2	4	9	2	9
J4	7	4	4	5	7	6	6	5	3	7
J5	7	9	1	1	1	2	6	8	1	5
J6	1	3	7	2	6	7	7	1	9	5
J7	6	9	1	1	4	1	9	5	9	5
J8	5	1	5	1	4	6	7	1	1	4
J9	4	4	1	6	9	3	9	9	9	3

SET 2-3										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	2	7	1	6	6	1	8	7	7	2
J1	6	6	1	6	5	8	8	8	7	5
J2	5	1	9	4	2	1	9	4	6	5
J3	6	7	7	4	9	2	6	9	8	8
J4	4	9	2	9	9	5	6	8	1	8
J5	2	9	4	4	9	4	4	3	6	9
J6	1	1	6	9	9	3	7	6	4	5
J7	4	9	1	3	6	8	7	9	6	4
J8	4	4	8	3	1	8	8	4	8	4
J9	1	5	1	7	6	4	6	3	2	4

SET 2-4										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	4	2	8	1	9	8	4	8	9	2
J1	8	1	8	7	1	6	5	8	4	7
J2	9	2	3	7	7	3	9	6	3	9
J3	2	8	9	2	5	5	2	3	8	3
J4	3	7	4	3	8	6	7	8	1	7
J5	5	3	5	3	3	5	4	3	7	3
J6	5	5	4	7	9	9	8	6	5	3
J7	2	2	7	7	7	2	2	3	8	8
J8	5	1	6	3	9	6	5	6	3	1
J9	2	4	8	8	3	5	3	5	7	1

SET 2-5										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	2	5	8	7	3	4	7	2	9	1
J1	9	8	2	4	7	1	1	2	3	8
J2	7	5	7	7	3	6	4	8	8	4
J3	7	1	7	4	4	3	6	5	2	5
J4	9	5	1	3	1	2	6	4	6	6
J5	8	8	7	2	4	8	4	3	7	3
J6	6	1	3	7	5	1	1	8	7	4
J7	6	1	8	7	9	5	9	9	7	7
J8	4	3	1	1	4	2	8	9	7	3
J9	8	1	8	1	3	9	2	6	3	8

SET 2-6										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	2	4	2	5	8	5	3	5	2	3
J1	3	6	1	6	4	7	3	1	5	9
J2	3	4	5	8	6	1	5	4	7	6
J3	5	9	4	2	3	8	5	8	8	9
J4	8	6	7	5	4	7	8	5	2	4
J5	7	1	2	5	2	4	6	4	4	5
J6	9	9	9	2	1	8	4	4	9	1
J7	6	8	6	7	4	1	9	8	2	1
J8	7	9	2	6	6	6	9	8	8	8
J9	3	4	6	2	2	6	9	6	6	8

SET 2-7										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	6	4	6	8	1	3	3	5	8	3
J1	2	8	5	8	6	1	4	5	1	1
J2	1	4	2	7	1	1	6	4	2	7
J3	3	2	5	5	4	2	4	9	2	9
J4	7	4	4	5	7	6	6	5	3	7
J5	7	9	1	1	1	2	6	8	1	6
J6	1	3	7	2	6	7	7	1	9	5
J7	6	9	1	1	4	1	9	5	9	5
J8	5	1	5	1	4	6	7	1	1	4
J9	4	4	1	6	9	3	9	9	9	3

SET 2-8										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	2	7	1	6	6	1	8	7	7	2
J1	6	6	1	6	5	8	8	8	7	5
J2	5	1	9	4	2	1	9	4	6	5
J3	6	7	7	4	9	2	6	9	8	8
J4	4	9	2	9	9	5	6	8	1	8
J5	2	9	4	4	9	4	4	3	6	9
J6	1	1	6	9	9	3	7	6	4	5
J7	4	9	1	3	6	8	7	9	6	4
J8	4	4	8	3	1	8	8	4	8	4
J9	1	5	1	7	6	4	6	3	2	4

SET 2-9										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	4	2	8	1	9	8	4	8	9	2
J1	8	1	8	7	1	6	5	8	4	7
J2	9	2	3	7	7	3	9	6	3	9
J3	2	8	9	2	5	5	2	3	8	3
J4	3	7	4	3	8	6	7	8	1	7
J5	5	3	5	3	3	5	4	3	7	3
J6	5	5	4	7	9	9	8	6	5	3
J7	2	2	7	7	7	2	2	3	8	8
J8	5	1	6	3	9	6	5	6	3	1
J9	2	4	8	8	3	5	3	5	7	1

SET 2-10										
	M0	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**
J0	2	5	8	7	3	4	7	2	9	1
J1	9	8	2	4	7	1	1	2	3	8
J2	7	5	7	7	3	6	3	9	8	4
J3	7	1	7	4	4	3	6	5	2	5
J4	9	5	1	3	1	2	6	4	6	6
J5	8	8	7	2	4	8	4	3	7	3
J6	6	1	3	7	5	1	1	8	7	4
J7	6	1	8	7	9	5	9	9	7	7
J8	4	3	1	1	4	2	8	9	7	3
J9	8	1	8	1	3	9	2	6	3	8

6.3 ΠΑΡΟΥΣΙΑΣΗ ΕΝΔΕΙΚΤΙΚΩΝ ΛΥΣΕΩΝ ΠΡΟΒΛΗΜΑΤΩΝ

Σε αυτή την ενότητα θα παρουσιαστούν ενδεικτικά μέσα από εικόνες οι λύσεις των προβλημάτων που αναλύθηκαν παραπάνω. Μετά από τυχαία επιλογή καταλήξαμε στη δοκιμή με αριθμό 5 από το δεύτερο σετ. Παρακάτω φαίνεται η λύση του συγκεκριμένου προβλήματος με το εργαλείο UPPAAL και με το εργαλείο LISA στο οποίο γίνεται ιδιαίτερη αναφορά στις 3 διαφορετικές μεθόδους επίλυσης:

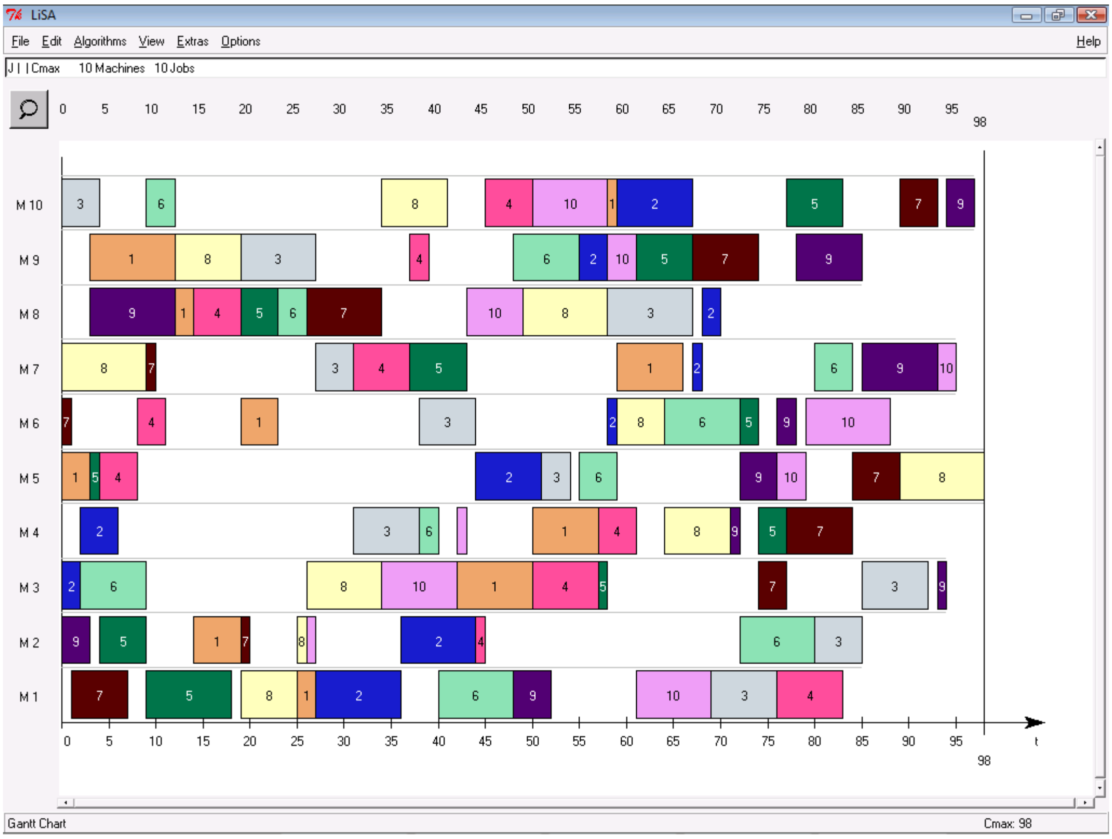
- Brucker ‘s Job-shop B&B
- Shifting Bottleneck
- Dispatching Rules (με προτεραιότητες FCFS, LQUE, LPT, EDD, WI, SPT, WSPT, ECT)

LISA- Brucker 's Job-shop B&B

Ένας αλγόριθμος Branch-and-Bound (κλάδου και ορίου) για συνήθεις αντικειμενικές συναρτήσεις επιλύει τα περισσότερα προβλήματα σε συστήματα παραγωγής open, job ή flow shop. Ο αλγόριθμός μας εφαρμόζει μία τεχνική εισαγωγής και παρεχόμενα από το χρήστη επάνω όριο και κάτω όρια. Το επάνω όριο μπορεί επίσης να προκύψει από προηγουμένως εφαρμοσμένες ευρετικές μεθόδους.

LISA- Shifting Bottleneck

Η διαδικασία Shifting Bottleneck Procedure (SBP), που δημιούργησε ο Adams το 1988 [Anant&Meeran,1998] εμπνεύστηκε από τη θεωρία των περιορισμών (Theory of constraints), είναι η επικρατέστερη τεχνική στη οποία βασίζονται οι ευρετικοί αλγόριθμοι. Η βασική ιδέα είναι η αναγωγή ενός προβλήματος m μηχανών σε επιμέρους m προβλήματα μιας μηχανής. Το κάθε υποπρόβλημα λύνεται μεμονωμένα και στη συνέχεια ανάλογα τη λύση οι αντίστοιχες μηχανές ιεραρχούνται σύμφωνα με την επίδοσή τους. Η μηχανή με την καλύτερη δυνατή λύση χαρακτηρίζεται ως «bottleneck» και έχει μεγαλύτερη προτεραιότητα από τις άλλες. Στη συνέχεια η μηχανή αυτή θεωρείται ως μηχανή αναφοράς και επομένως όλες οι προηγούμενες εργασίες επαναπροσδιορίζονται με τα νέα όμως δεδομένα. Η ίδια διαδικασία επαναλαμβάνεται και για τις υπόλοιπες μηχανές.



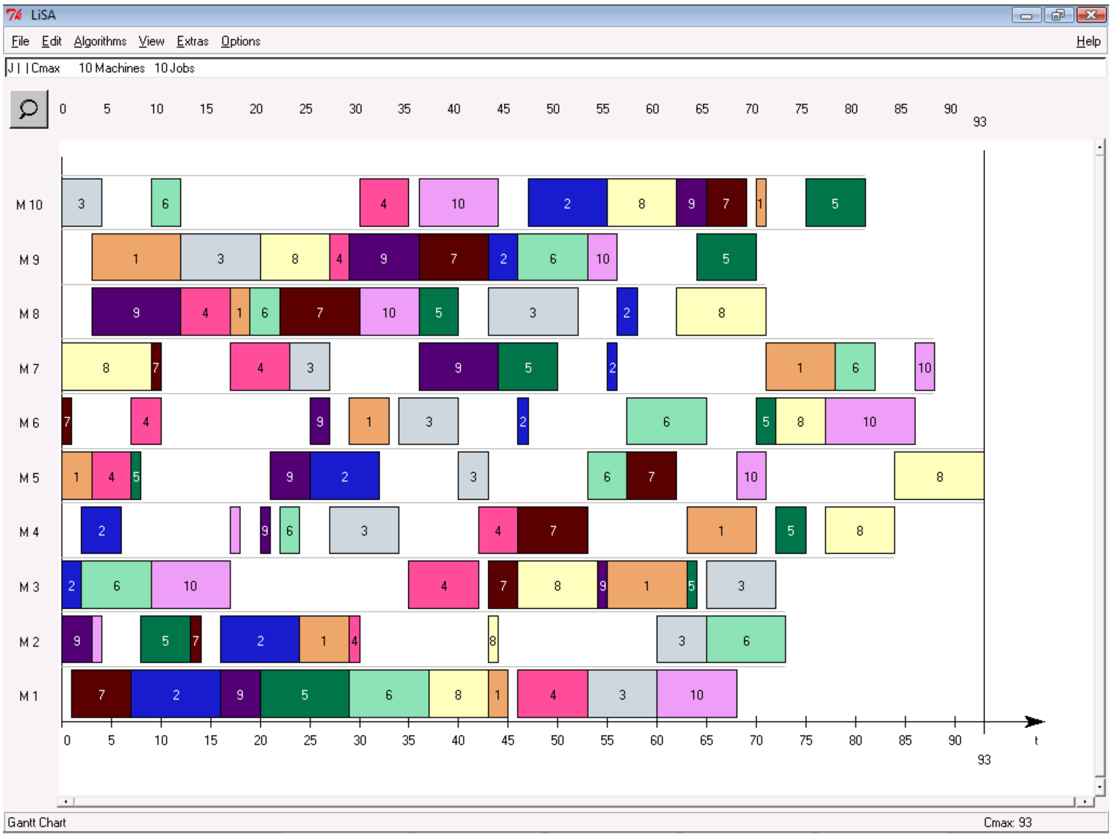
LISA- Dispatching Rules

Οι μέθοδοι προσέγγισης που αρχικά αναπτύχθηκαν με σκοπό να εφαρμοστούν σε τέτοιου είδους προβλήματα βασίστηκαν σε κανόνες διεκπεραίωσης προτεραιότητας, οι οποίοι λόγω της ευκολίας στην εφαρμογή τους και στον σημαντικά μειωμένο χρόνο υπολογισμού που απαιτούν, θεωρούνται ως μία από τις πιο δημοφιλείς μεθόδους *(Baker 1974 , French 1982 , Morton and Pentico 1993).*

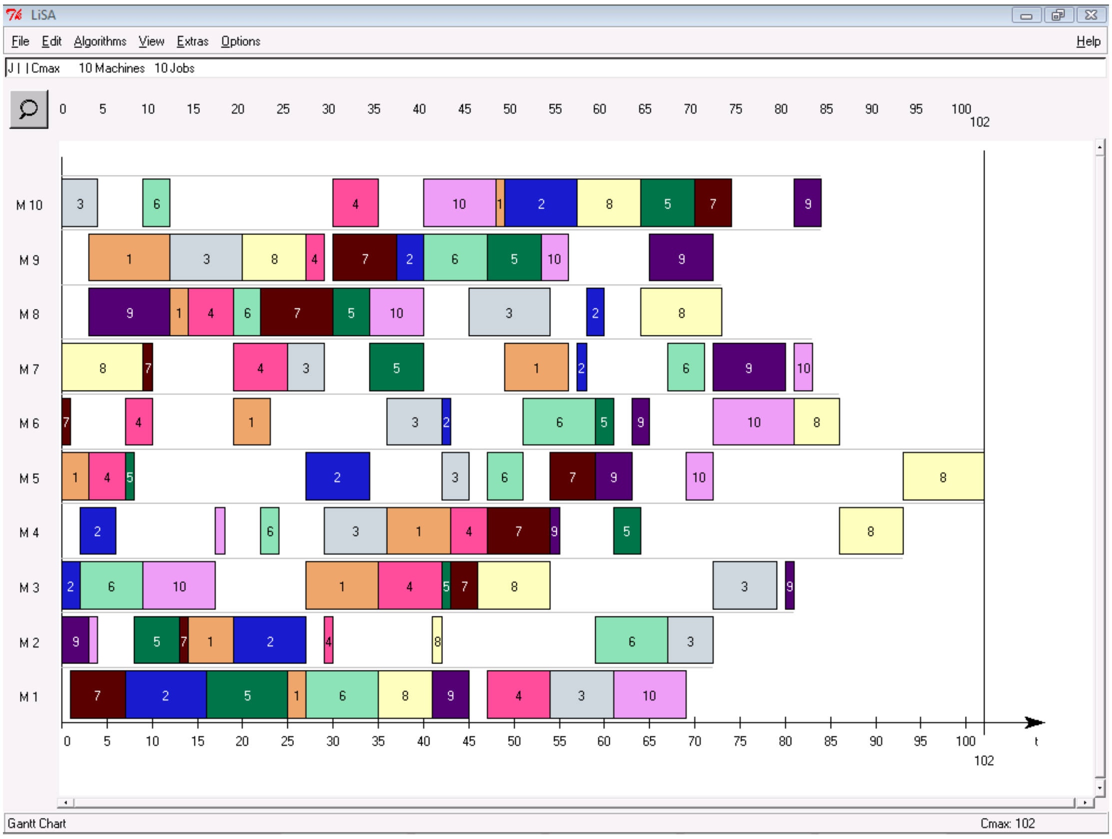
Σε κάθε διαδοχικό στάδιο σε όλες τις διεργασίες που είναι έτοιμες να υποστούν επεξεργασία ανατίθεται ένας βαθμός προτεραιότητας και η διεργασία που διαθέτει το μεγαλύτερο βαθμό προτεραιότητας διαλέγεται να υποστεί επεξεργασία, τη δεδομένη χρονική στιγμή. Συνήθως πολλές επαναλήψεις της εφαρμογής των κανόνων αυτών απαιτούνται προκειμένου να επιτευχθεί η λήψη σωστών και έγκυρων αποτελεσμάτων.

Στα παρακάτω παραδείγματα έχουμε ομαδοποιήσει κάποιους από τους αλγορίθμους προτεραιοτήτων διότι , πράγμα που είναι απόλυτα λογικό, οδηγούσαν στις ίδιες ακριβώς λύσεις των προβλημάτων μας.

LISA- Dispatching Rules (FCFS-εξυπηρέτηση με βάση τη σειρά άφιξης στο σύστημα)

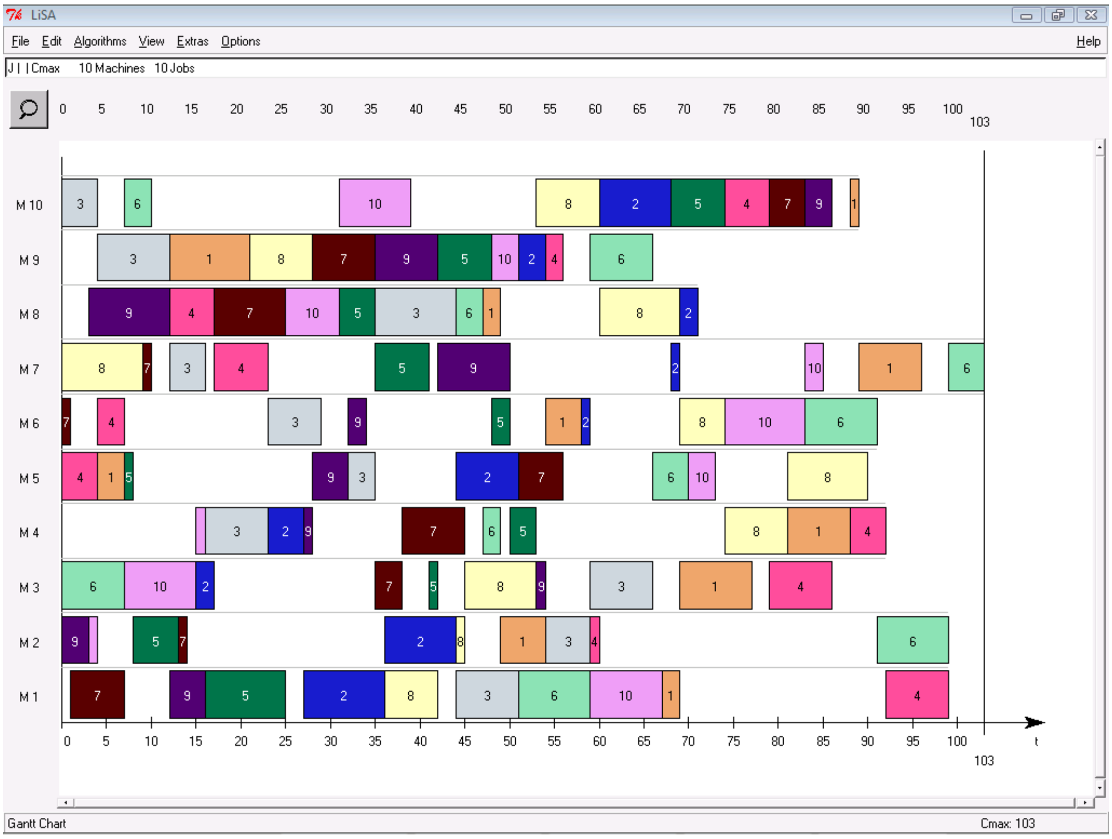


LISA- Dispatching Rules (EDD-νωρίτερη ημερομηνίας περάτωσης πρώτα, WI- μεγαλύτερο βάρος πρώτα)

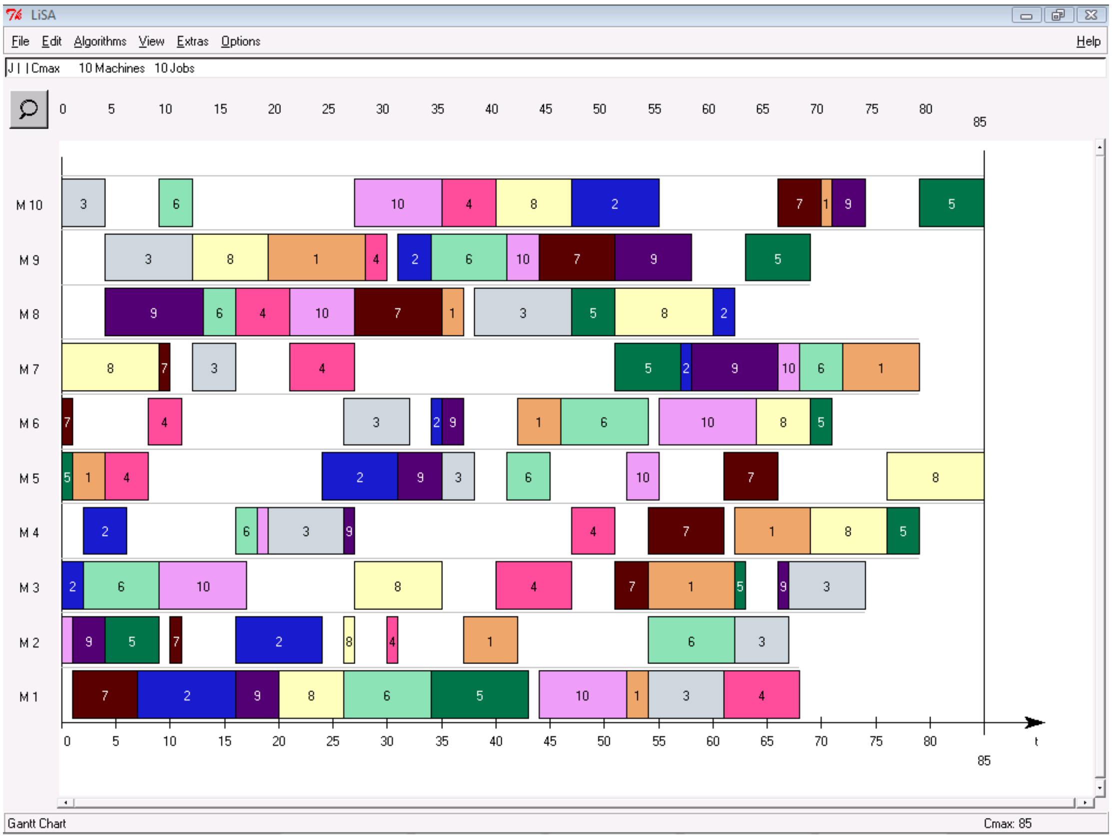


LISA- Dispatching Rules (LQUE-με μικρότερη διαφορά ημερομηνίας περάτωσης και χρόνου(επεξεργασίας + ουράς) ,

LPT-μεγαλύτερος χρόνος επεξεργασίας πρώτα)



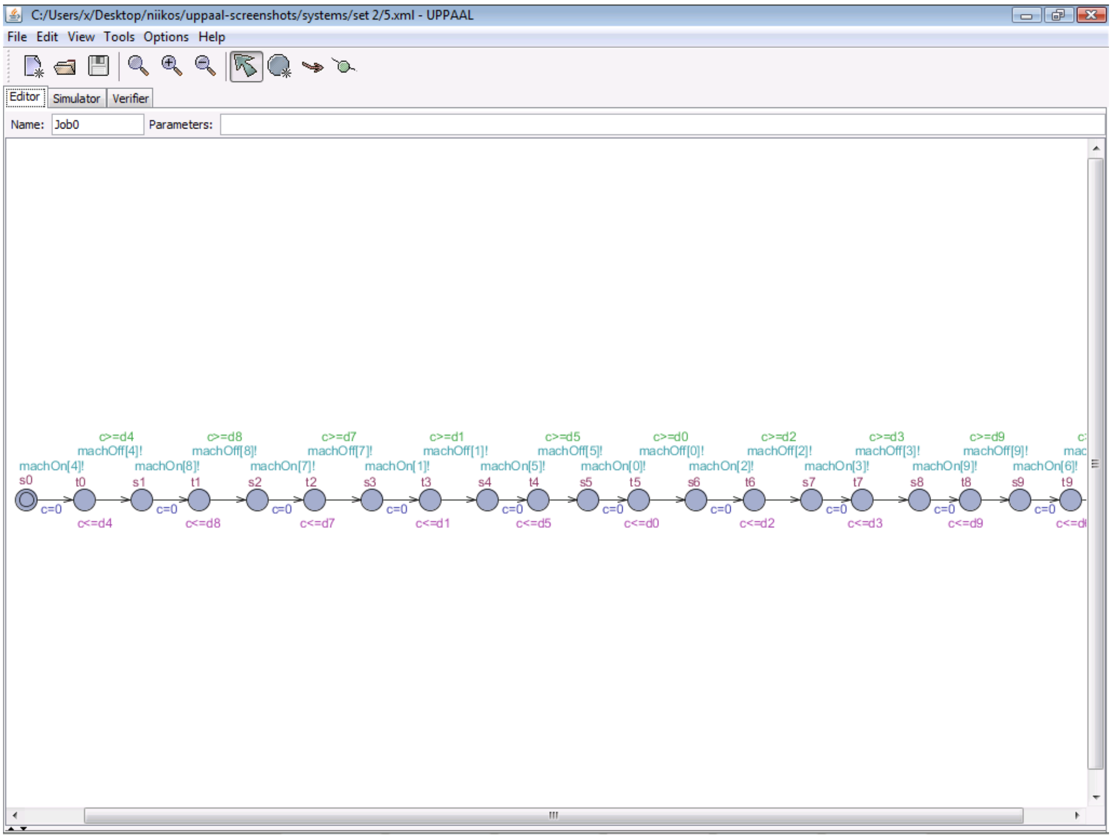
LISA- Dispatching Rules (SPT-συντομότερος χρόνος επεξεργασίας πρώτα, WSPT-συντομότερος σταθμισμένος χρόνος επεξεργασίας πρώτα, ECT-"εφικτός" χρόνος νωρίτερης ολοκλήρωσης πρώτα)



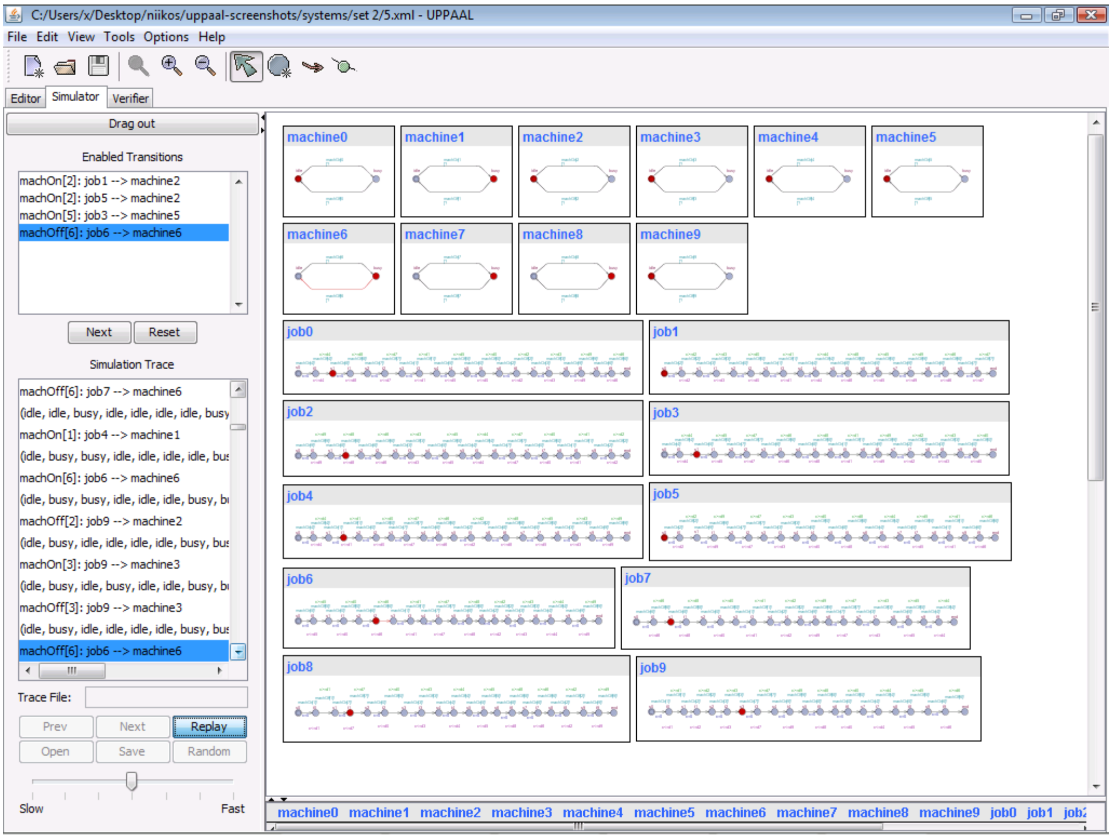
UPPAAL

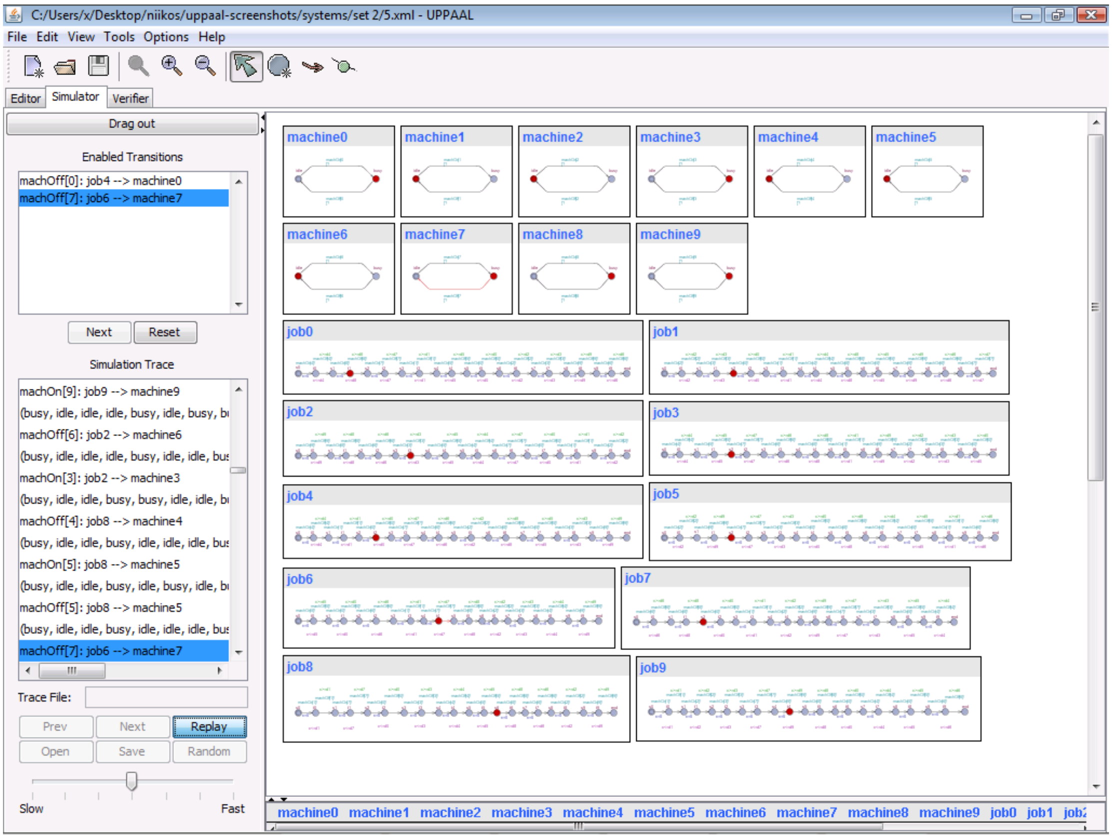
Σε αντίθεση με την προηγούμενη προσέγγιση, στην επίλυση με τα χρονισμένα αυτόματα και επειδή η λύση προκύπτει από μία σειρά χρονισμένων βημάτων θα δείξουμε σταδιακά τον τρόπο με τον οποίο φτάνουμε στη μορφή της λύσης.

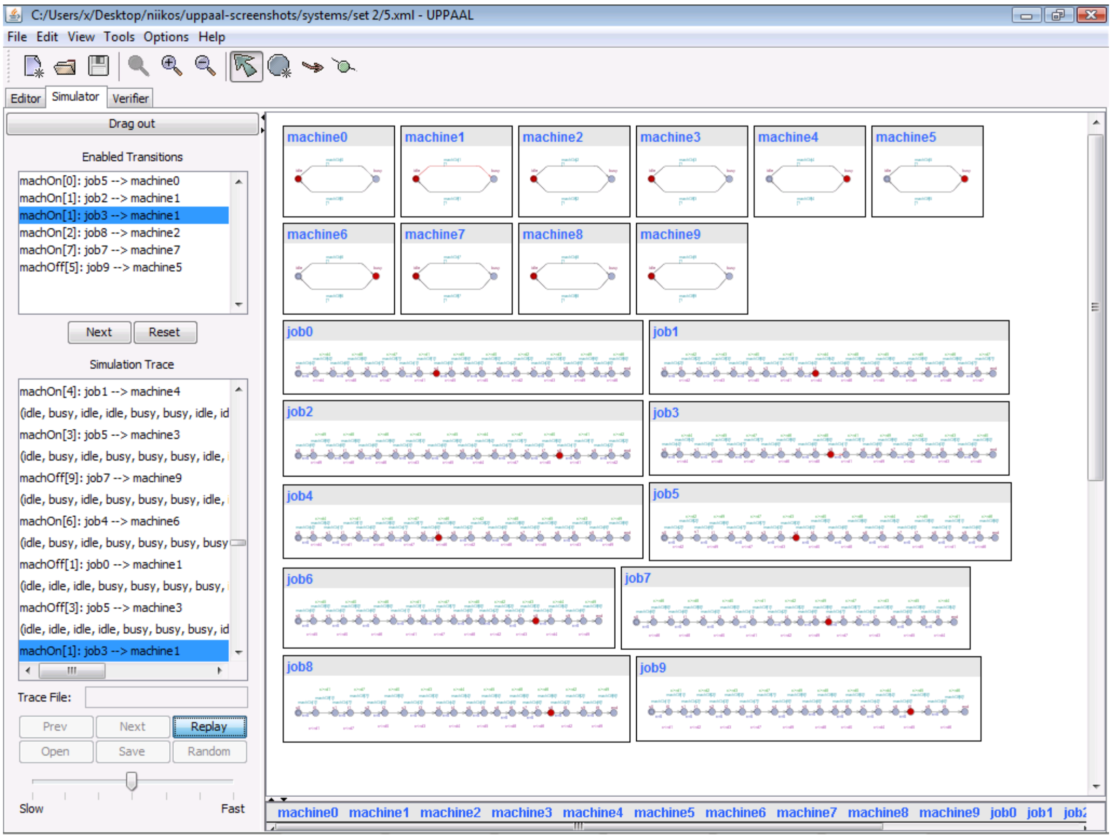
Στην πρώτη εικόνα φαίνεται ο τρόπος που έχουμε προγραμματίσει το UPPAAL για να φτάσουμε στην ολοκλήρωση της Job 0 του παραδείγματός μας. Με παρόμοιο τρόπο αλλά διαφορετική τιμή μεταβλητών έχουν προγραμματιστεί και οι υπόλοιπες εργασίες.

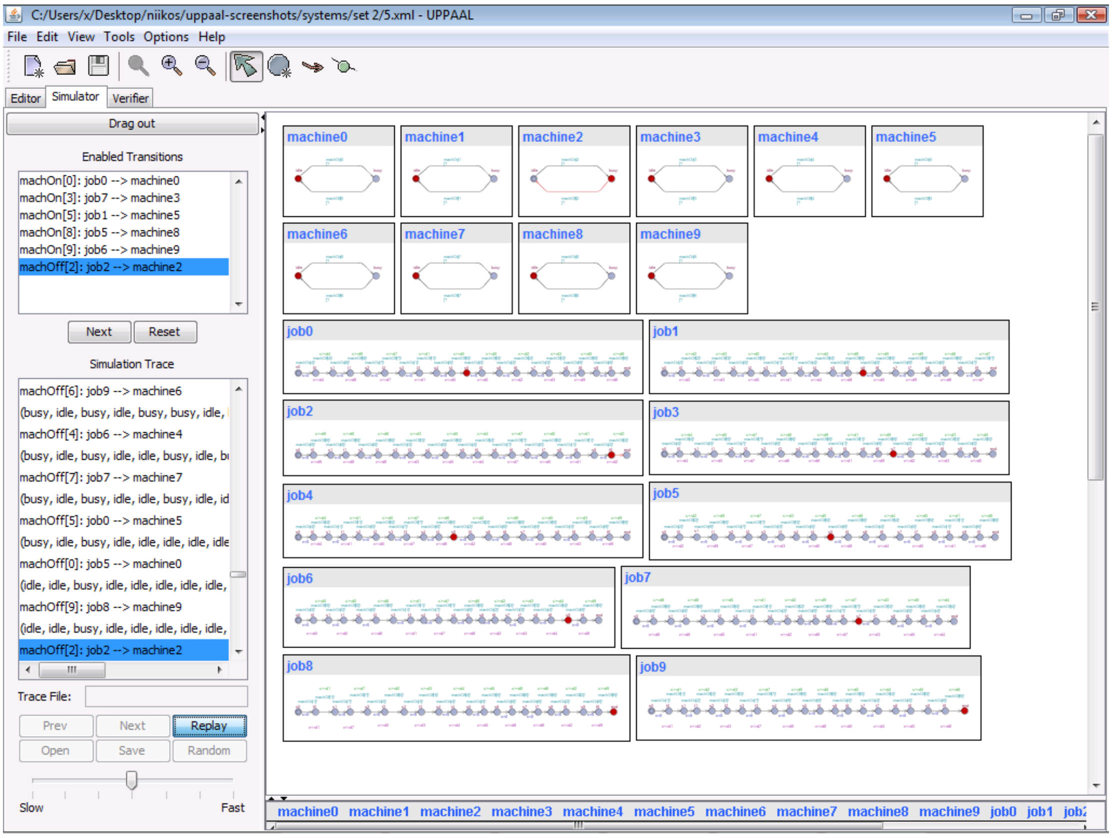


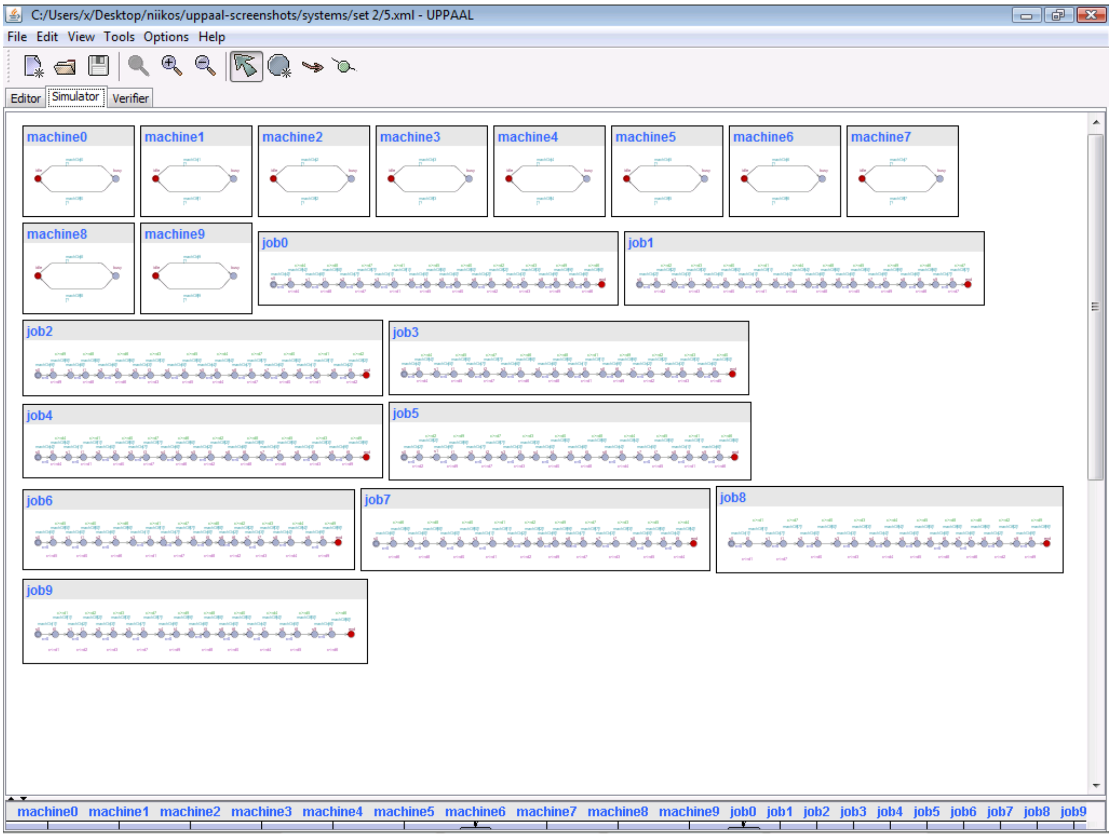
Στις παρακάτω εικόνες δείχνουμε ενδεικτικά 5 διαφορετικές φάσεις - βήματα της διαδικασίας επίλυσης. Η τελευταία εικόνα αποτελεί την ολοκλήρωση - λύση του προβλήματος όπου όπως γίνεται αντιληπτό όλες οι εργασίες έχουν φτάσει στο τέλος τους.


C:/Users/x/Desktop/niikos/uppaal-screenshots/systems/set 2/5.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
Drag out
Enabled Transitions
machOn[2]: job1 --> machine2
machOn[2]: job5 --> machine2
machOn[5]: job3 --> machine5
machOff[6]: job6 --> machine6
Next Reset
Simulation Trace
machOff[6]: job7 --> machine6
(idle, idle, busy, idle, idle, idle, idle, busy
machOn[1]: job4 --> machine1
(idle, busy, busy, idle, idle, idle, idle, bus
machOn[6]: job6 --> machine6
(idle, busy, busy, idle, idle, idle, busy, bu
machOff[2]: job9 --> machine2
(idle, busy, idle, idle, idle, idle, busy, bus
machOn[3]: job9 --> machine3
(idle, busy, idle, busy, idle, idle, busy, bu
machOff[3]: job9 --> machine3
(idle, busy, idle, idle, idle, idle, busy, bus
machOff[6]: job6 --> machine6
Trace File:
Prev Next Replay
Open Save Random
Slow Fast
machine0 machine1 machine2 machine3 machine4 machine5 machine6 machine7 machine8 machine9
job0 job1 job2 job3 job4 job5 job6 job7 job8 job9


C:/Users/x/Desktop/niikos/uppaal-screenshots/systems/set 2/5.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
Drag out
Enabled Transitions
machOff[0]: job4 --> machine0
machOff[7]: job6 --> machine7
Next Reset
Simulation Trace
machOn[9]: job9 --> machine9
(busy, idle, idle, idle, busy, idle, busy, bu
machOff[6]: job2 --> machine6
(busy, idle, idle, idle, busy, idle, idle, bus
machOn[3]: job2 --> machine3
(busy, idle, idle, busy, busy, idle, idle, bu
machOff[4]: job8 --> machine4
(busy, idle, idle, busy, idle, idle, idle, bus
machOn[5]: job8 --> machine5
(busy, idle, idle, busy, idle, busy, idle, bu
machOff[5]: job8 --> machine5
(busy, idle, idle, busy, idle, idle, idle, bus
machOff[7]: job6 --> machine7
Trace File:
Prev Next Replay
Open Save Random
Slow Fast
machine0 machine1 machine2 machine3 machine4 machine5 machine6 machine7 machine8 machine9
job0 job1 job2 job3 job4 job5 job6 job7 job8 job9


C:/Users/x/Desktop/niikos/uppaal-screenshots/systems/set 2/5.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
Drag out
Enabled Transitions
machOn[0]: job5 --> machine0
machOn[1]: job2 --> machine1
machOn[1]: job3 --> machine1
machOn[2]: job8 --> machine2
machOn[7]: job7 --> machine7
machOff[5]: job9 --> machine5
Next Reset
Simulation Trace
machOn[4]: job1 --> machine4
machOn[3]: job5 --> machine3
machOff[9]: job7 --> machine9
machOn[6]: job4 --> machine6
machOff[1]: job0 --> machine1
machOff[3]: job5 --> machine3
machOn[1]: job3 --> machine1
Trace File:
Prev Next Replay
Open Save Random
Slow Fast
machine0 machine1 machine2 machine3 machine4 machine5 machine6 machine7 machine8 machine9
job0 job1 job2 job3 job4 job5 job6 job7 job8 job9


C:/Users/x/Desktop/niikos/uppaal-screenshots/systems/set 2/5.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
Drag out
Enabled Transitions
machOn[0]: job0 --> machine0
machOn[3]: job7 --> machine3
machOn[5]: job1 --> machine5
machOn[8]: job5 --> machine8
machOn[9]: job6 --> machine9
machOff[2]: job2 --> machine2
Next Reset
Simulation Trace
machOff[6]: job9 --> machine6
machOff[4]: job6 --> machine4
machOff[7]: job7 --> machine7
machOff[5]: job0 --> machine5
machOff[0]: job5 --> machine0
machOff[9]: job8 --> machine9
machOff[2]: job2 --> machine2
Trace File:
Prev Next Replay
Open Save Random
Slow Fast
machine0 machine1 machine2 machine3 machine4 machine5 machine6 machine7 machine8 machine9
job0 job1 job2 job3 job4 job5 job6 job7 job8 job9


C:/Users/x/Desktop/niikos/uppaal-screenshots/systems/set 2/5.xml - UPPAAL
File Edit View Tools Options Help
Editor Simulator Verifier
machine0 machine1 machine2 machine3 machine4 machine5 machine6 machine7
machine8 machine9 job0 job1
job2 job3
job4 job5
job6 job7 job8
job9
machine0 machine1 machine2 machine3 machine4 machine5 machine6 machine7 machine8 machine9 job0 job1 job2 job3 job4 job5 job6 job7 job8 job9

ΚΕΦΑΛΑΙΟ 7

Αποτελέσματα-Συμπεράσματα

7. ΑΠΟΤΕΛΕΣΜΑΤΑ- ΣΥΜΠΕΡΑΣΜΑΤΑ

7.1 ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο θα παρουσιαστούν τα συγκεντρωτικά αποτελέσματα, σε πίνακες, που θα μας οδηγήσουν σε ασφαλή συμπεράσματα όσον αφορά την ακριβέστερη και πιο προσεγγιστική στο επιθυμητό αποτέλεσμα λύση του προβλήματός μας.

7.2 ΣΥΓΚΕΝΤΡΩΤΙΚΟΙ ΠΙΝΑΚΕΣ

Στην ενότητα αυτή παρατίθενται οι συγκεντρωτικοί πίνακες με τα χρονικά αποτελέσματα των διαφορετικών μεθόδων που χρησιμοποιήσαμε. Σε πρώτη φάση έχουμε διαχωρίσει τους πίνακες ώστε να συγκρίνουμε τα αποτελέσματα ανά σετ προβλημάτων αλλά και μέθοδο, ενώ στον τελευταίο συγκεντρωτικό πίνακα έχουμε υπολογίσει τους χρονικούς μέσους όρους των διαφορετικών μεθόδων επίλυσης.

SET 1							
ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑ ΜΕΘΟΔΟ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ							
Α/Α	ΒΙΒΛΙΟΘΗΚΗ LISA						ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ
	BRUCKER	DISP EDD-WI	DISP FCFS	DISP LQUE-LPT	DISP SPT-WSPT-ECT	SHIFTING BOTTLENECK	UPPAAL
1	81	111	105	107	102	105	130
2	87	114	111	137	102	104	157
3	80	116	113	98	86	92	126
4	91	106	111	126	97	114	126
5	81	105	103	107	98	105	123
6	86	114	103	110	99	93	142
7	79	140	100	108	136	99	126
8	81	113	97	112	112	116	125
9	89	115	107	104	121	114	144
10	73	103	90	102	85	94	116

11	93	122	106	111	111	117	135
12	88	112	97	126	108	117	148
13	86	143	106	127	101	102	144
14	84	121	95	114	95	105	135
15	81	116	100	111	86	104	136
16	90	131	109	111	118	110	145
17	82	114	100	106	98	118	145
18	73	103	90	102	85	94	119
19	93	122	106	111	111	117	139
20	88	112	97	126	108	117	132
ΣΥΝΟΛΟ	**1686**	**2333**	**2046**	**2256**	**2059**	**2137**	**2693**

SET 2							
ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑ ΜΕΘΟΔΟ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ							
Α/Α	**ΒΙΒΛΙΟΘΗΚΗ LISA**						**ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ**
	BRUCKER	**DISP EDD-WI**	**DISP FCFS**	**DISP LQUE-LPT**	**DISP SPT-WSPT-ECT**	**SHIFTING BOTTLENECK**	**UPPAAL**
1	78	88	100	99	100	92	137
2	73	92	94	100	93	96	128
3	86	102	106	115	99	106	140
4	78	88	100	99	100	92	127
5	79	102	93	103	85	98	127
6	83	96	101	110	102	117	146
7	73	92	94	100	93	96	133
8	86	102	106	115	99	106	144
9	78	88	100	99	100	92	121
10	79	102	93	103	85	98	131
11	73	92	94	100	93	96	105
12	86	102	106	115	99	106	138
13	78	88	100	99	100	92	134
14	79	102	93	103	85	98	121
15	73	92	94	100	93	96	125
16	86	102	106	115	99	106	129
17	78	88	100	99	100	92	141
18	79	102	93	103	85	98	137
19	83	96	101	110	102	117	127
20	75	101	95	104	102	108	138
ΣΥΝΟΛΟ	**1583**	**1917**	**1969**	**2091**	**1914**	**2002**	**2629**

SET 3							
ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑ ΜΕΘΟΔΟ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ							
Α/Α	**ΒΙΒΛΙΟΘΗΚΗ LISA**						**ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ**
	BRUCKER	**DISP EDD-WI**	**DISP FCFS**	**DISP LQUE-LPT**	**DISP SPT-WSPT-ECT**	**SHIFTING BOTTLENECK**	**UPPAAL**
1	84	117	99	138	101	122	139
2	73	111	91	96	100	91	130
3	86	100	97	119	103	102	128
4	77	113	112	112	108	108	123
5	83	107	90	120	114	98	155
6	86	117	117	113	115	107	152
7	84	117	99	138	101	122	153
8	84	100	101	101	114	101	140
9	80	94	94	100	104	113	126
10	79	95	95	112	95	99	126
11	80	106	97	107	118	97	133
12	82	105	96	103	108	102	133
13	76	109	99	115	98	98	133
14	84	113	118	116	105	119	151
15	73	97	84	98	88	94	142
16	82	112	97	106	108	98	140
17	78	113	106	112	90	99	149
18	79	101	106	112	94	97	129
19	87	123	122	115	112	115	145
20	79	113	93	100	101	104	129
ΣΥΝΟΛΟ	**1616**	**2163**	**2013**	**2233**	**2077**	**2086**	**2756**

SET 4							
ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑ ΜΕΘΟΔΟ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ							
Α/Α	**ΒΙΒΛΙΟΘΗΚΗ LISA**						**ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ**
	BRUCKER	**DISP EDD-WI**	**DISP FCFS**	**DISP LQUE-LPT**	**DISP SPT-WSPT-ECT**	**SHIFTING BOTTLENECK**	**UPPAAL**
1	77	89	91	94	83	90	129
2	79	118	97	127	84	110	142
3	78	113	97	117	90	103	116
4	75	102	104	93	90	94	121
5	81	103	99	110	111	105	132
6	85	106	102	109	102	114	126
7	79	117	92	102	96	105	149
8	84	133	107	101	104	102	137
9	72	107	97	94	91	83	120
10	91	116	100	111	126	115	126
11	87	99	102	104	103	109	152
12	84	117	100	115	102	102	139
13	83	110	99	117	117	109	136
14	79	117	92	102	96	105	140
15	71	90	80	83	85	99	149
16	84	133	107	101	104	102	121
17	72	107	97	94	91	83	126
18	83	96	98	98	98	95	123
19	80	122	86	116	89	106	117
20	84	117	100	115	102	102	133
ΣΥΝΟΛΟ	**1608**	**2212**	**1947**	**2103**	**1864**	**2033**	**2634**

SET 5							
ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑ ΜΕΘΟΔΟ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ							
Α/Α	**ΒΙΒΛΙΟΘΗΚΗ LISA**						**ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ**
	BRUCKER	**DISP EDD-WI**	**DISP FCFS**	**DISP LQUE-LPT**	**DISP SPT-WSPT-ECT**	**SHIFTING BOTTLENECK**	**UPPAAL**
1	78	113	97	108	85	99	145
2	75	106	88	107	80	106	126
3	82	103	95	107	97	126	125
4	81	100	101	105	103	111	136
5	82	92	91	103	87	133	132
6	81	100	101	105	103	111	147
7	77	109	93	112	89	103	114
8	85	102	109	95	98	110	135
9	81	112	98	102	95	105	130
10	82	103	95	107	97	126	132
11	82	92	91	103	87	133	141
12	79	90	90	98	92	96	130
13	78	101	87	115	95	93	131
14	85	106	103	115	115	119	134
15	83	111	110	119	104	97	137
16	74	117	87	90	96	82	119
17	76	113	91	98	98	96	137
18	74	105	104	110	98	97	113
19	82	105	101	113	92	95	140
20	81	107	101	105	103	111	130
ΣΥΝΟΛΟ	**1598**	**2087**	**1933**	**2117**	**1914**	**2149**	**2634**

ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑ ΜΕΘΟΔΟ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ							
	ΒΙΒΛΙΟΘΗΚΗ LISA						**ΧΡΟΝΙΣΜΕΝΑ ΑΥΤΟΜΑΤΑ**
ΧΡΟΝΙΚΟΣ ΜΟ	**BRUCKER**	**DISP EDD-WI**	**DISP FCFS**	**DISP LQUE-LPT**	**DISP SPT-WSPT-ECT**	**SHIFTING BOTTLENECK**	**UPPAAL**
SET 1	84,3	116,7	102,3	112,8	103,0	106,9	134,7
SET 2	79,2	95,9	98,5	104,6	95,7	100,1	131,5
SET 3	80,8	108,2	100,7	111,7	103,9	104,3	137,8
SET 4	80,4	110,6	97,4	105,2	93,2	101,7	131,7
SET 5	79,9	104,4	96,7	105,9	95,7	107,5	131,7
ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΙΚΟΣ ΜΟ	**80,9**	**107,1**	**99,1**	**108,0**	**98,3**	**104,1**	**133,5**

7.3 ΣΥΜΠΕΡΑΣΜΑΤΑ

Στη διάρκεια εκπόνησης της παρούσης εργασίας αποδόθηκαν λύσεις στο πρόβλημα του Χρονοπρογραμματισμού παραγωγής τόσο με τη χρήση αλγορίθμων που βασίζονται στα χρονισμένα αυτόματα όσο και με τη χρήση αλγορίθμων από την βιβλιοθήκη LISA η οποία εμπεριέχει ευρετικούς, εποικοδομητικούς και άλλων ειδών αλγορίθμους. Σκοπός μας ήταν εξαρχής η σύγκριση των διαφορετικών μεθόδων επίλυσης ως προς τα χρονικά αποτελέσματά τους, αλλά και η εξαγωγή χρήσιμων συμπερασμάτων που θα μας βοηθήσει στη περεταίρω αξιολόγησή τους και σε μια μελλοντική πιθανή αξιοποίησή τους.

Όπως φαίνεται λοιπόν και από τους συγκεντρωτικούς πίνακες αποτελεσμάτων τα χρονισμένα αυτόματα οδηγούν σε μια λύση το πρόβλημα, αλλά όχι τόσο αποτελεσματική όσον αφορά την ελαχιστοποίηση του συνολικού χρόνου διεκπεραίωσης της παραγωγής. Το σημαντικό τους πλεονέκτημα είναι η μεγάλη ευελιξία που έχουν όσον αφορά ειδικής δομής προβλήματα. Με τις δυνατότητες μοντελοποίησης τους, μας δίνουν την άνεση να επιλύσουμε προβλήματα με αρκετούς περιορισμούς ή ειδικές συνθήκες οι οποίες θα ήταν αδύνατο να προσαρτηθούν σε κάποιο από τα άλλα μοντέλα επίλυσης προβλημάτων τέτοιου τύπου. Ο βαθμός τυχαιότητας όμως που υφίσταται σε έναν αλγόριθμο χρονισμένων αυτομάτων και αφορά κυρίως το κομμάτι της αναζήτησης του δένδρου του γραφήματος για να βρεθεί η βέλτιστη λύση είναι αρκετά μεγάλος, ώστε τελικά η λύση του προβλήματος να είναι και η λιγότερο αποδοτική.

Η μέθοδος Brucker B&B (Branch and Bound) όπως φαίνεται στο συγκεντρωτικό πίνακα είναι η μέθοδος που μας οδήγησε πιο κοντά από όλες στη βέλτιστη λύση του προβλήματος χρονοπρογραμματισμού. Αυτό είναι κάτι που περιμέναμε διότι λόγω φύσης και κατασκευής του αλγορίθμου θα μας οδηγούσε στο καλύτερο αποτέλεσμα. Άλλωστε η χρήση ευρετικών αλγορίθμων που περιορίζουν το εύρος των επιθυμητών αποτελεσμάτων (άνω όριο) είναι το στοιχείο που βοηθάει τον αλγόριθμο Brucker να αναζητά λύσεις στο δέντρο του γραφήματος κάτω από το ελάχιστο άνω όριο. Το μειονέκτημα του αλγορίθμου Brucker καθώς και των υπολοίπων αλγορίθμων της βιβλιοθήκης LISA είναι ότι η μοντελοποίηση περιορίζεται αποκλειστικά σε τέτοιου τύπου προβλήματα (open, job-shop, flow-shop) και αδυνατεί να οδηγήσει στην επίλυση προβλήματα ειδικών κατηγοριών όπως π.χ.

προβλήματα με περιορισμούς στις μηχανές, με πολλαπλή χρήση των ιδίων μηχανών, με ταυτόχρονη χρήση μηχανών για την ίδια εργασία κ.τ.λ.

Αναφορικά με τις άλλες μεθόδους της βιβλιοθήκης LISA που χρησιμοποιήσαμε συμπεραίνουμε ότι μας οδηγούν επιτυχημένα στη λύση του προβλήματος αλλά όχι τόσο αποδοτικά όσο η μέθοδος αλγορίθμου Brucker. Οι πιο κοντινοί στην βέλτιστη λύση, στο συγκεκριμένο τουλάχιστον παράδειγμα, είναι οι αλγόριθμοι με προτεραιότητες: SPT (συντομότερος χρόνος επεξεργασίας πρώτα), WSPT (συντομότερος σταθμισμένος χρόνος επεξεργασίας πρώτα), ECT ("εφικτός" χρόνος νωρίτερης ολοκλήρωσης πρώτα), πράγμα το οποίο ήταν αναμενόμενο διότι αν και δεν διαθέτουν την πολυπλοκότητα αναζήτησης που έχει ο αλγόριθμος Brucker ωστόσο οι προτεραιότητές τους είναι τέτοιες που έχουν εξ' ορισμού στόχο να ελαχιστοποιήσουν τον συνολικό χρόνο ολοκλήρωσης των εργασιών.

Ολοκληρώνοντας την εργασία, καταλήγουμε ότι ο συνδυασμός των χρονισμένων αυτομάτων με ευρετικούς και branch and bound αλγόριθμους θα μας προσδώσουν ακρίβεια περιορίζοντας το εύρος λύσεων αλλά και ευελιξία στο να μοντελοποιούμε ειδικής δομής προβλήματα. Ουσιαστικά θα πρέπει να εκμεταλλευτούμε τα πλεονεκτήματα της κάθε μίας από τις μεθόδους που εξετάσαμε. Έτσι, βασισμένοι στη μοντελοποίηση των χρονισμένων αυτομάτων θα μπορούμε να διαμορφώσουμε πρόβλημα οιασδήποτε μορφής, δομής και συνθηκών, το οποίο εκ των προτέρων θα έχει ελεγχθεί με κάποιον ευρετικό ή branch and bound αλγόριθμο με σκοπό την ελαχιστοποίηση των δένδρων αναζήτησης του γραφήματος που είναι πιθανά να μας οδηγήσουν στη βέλτιστη λύση. Αξίζει να σημειωθεί ότι στην πρόσφατη βιβλιογραφία έχουν ήδη γίνει αναφορές σε μεθόδους που χρησιμοποιούν συνδυαστικές προσεγγίσεις στο πλαίσιο αυτών που αναφέρουμε παραπάνω και έχουν οδηγήσει στην επίλυση προβλημάτων με πολύ ενθαρρυντικά αποτελέσματα (S.Subbiah et al,2011, S.Subbiah&S.Engell,2010).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Παππής. Κ., 'Προγραμματισμός Παραγωγής', Εκδ. Α.Σταμούλης, Αθήνα – Πειραιάς (2001)

[2] Εργαζάκης, Κ.Α., 'Ανάπτυξη Έμπειρου Συστήματος Ευφυούς Διαχείρισης Αλγορίθμων Προγραμματισμού στη Βιομηχανική Παραγωγή', Διπλωματική Εργασία, Εργαστήριο Συστημάτων Αποφάσεων και Διοίκησης, Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Ε.Μ.Π. (2001)

[3] Jain, A.S., Meeran, S., 'A state-of-the-art review of job-shop scheduling techniques', Technical Report, Department of Applied Physics, Electronic and Mechanical Engineering, University of Dundee, Dundee, Scotland (1998)

[4] Roy, B., Sussmann, B., 'Les probl`*emes d'ordonnancement avec contraintes disjonctives*.', Note DS 9 bis, SEMA, Paris (1964)

[5] Brucker, P., '*Scheduling Algorithms*', 3rd Edition, Springer – Verlag, Berlin (2001)

[6] Kaskavelis, C., Caramanis, M., 'Efficient Langrangian relaxation algorithms for industry size job-shop scheduling problems', *IIE Transactions*, 30, pp. 1085-1097 (1998)

[7] Moursli O., Pochet, Y., 'A branch-and-bound algorithm for the hybrid flowshop', *Int. J. Production Economics*, 64, pp. 113-125 (2000)

[8] Glover, F., Greenberg, H.J., 'New approaches for heuristic search: bilateral linkage with operations research', *European Journal of Operations Research*, 39, pp. 119-130 (1989)

[9] Jayamohan, M.S., Rajendran, C., 'New dispatching rules for shop scheduling: a step forward', *Int J. Prod. Res.*, 38, (3), pp. 563-586 (2000)

[10] Rajendran C., Ziegler, H., 'Heuristics for scheduling in flowshops and flowline-based manufacturing cells to minimize the sum of weighted flowtime and weighted tardiness of jobs', *Computers & Industrial Engineering*, 37, pp. 671-690 (1999)

[11] McMullen, P., Frazier, G., 'A simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line', *IEE Transactions*, 32, pp. 679-686 (2000)

[12] Armentano, V., Scrich, C., 'Tabu search for minimizing total tardiness in a job shop', *Int. J. Production Economics*, 63, pp. 131-140 (2000)

[13] Dornan, B., 'A Status Report: Artificial Intelligence', *Production*, pp.46-50 (1987)

[14] Γιαννοπούλου, Ε.Α., 'Χρονοπρογραμματισμός Εργασιών με χρήση Νευρωνικών Δικτύων', Διπλωματική Εργασία, Εργαστήριο Συστημάτων Αποφάσεων και Διοίκησης, Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Ε.Μ.Π. (2003)

[15] Fisher, H., Thompson, G.L., 'Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules', Prentice-Hall, Englewood Cliffs, NJ (1963)

[16] Adams, J., Balas, E., Zawack, D., 'The shifting bottleneck procedure for job shop scheduling', *Management Science*, 34, pp. 391-401 (1988)

[17] Demirkol, E., Mehta, S., Uzsoy, R., 'A Computational Study of Shifting Bottleneck Procedures for Shop Scheduling Problems', *Journal of Heuristics*, 3 (2), 111-137 (1997)

[18] Carlier, J., 'The one-machine sequencing problem', *European Journal of Operational Research*, 11, pp. 42-47 (1982)

[19] Lambrecht, M., Ivens, P., 'Extending the Shifting Bottleneck Procedure to real-Life Applications', *European Journal of Operational Research*, 90, pp. 252-268 (1996)

[20] Balas, E., Lancia, G., Serafini, P., Vazacopoulos, A., 'Job shop scheduling with dead-lines', *Journal of Combinatorial Optimization*, 1, pp. 329-353 (1998)

[21] Peter Niebert, Stavros Tripakis, Sergio Yovine : ''Minimum-time reachability for timed automata '', June 2000.

[22] C.L. Liu: ''Στοιχεία διακριτών μαθηματικών'', Πανεπιστημιακές εκδόσεις Κρήτης.

[23] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine: "Symbolic model checking for real-time systems", Information and Computation 111, 1994, pp. 193-244.

[24] L. Halkjaer, K. Haervi, A. Ingolfsdottir: "Verification of the legOS scheduler using Uppaal", Theoritical computer science 39 No. 3 (2000).

[25] Luca Aseto, Augusto Burgueno, kim Larsen: "Model checking via reachability testing for timed automata", November 1997.

[26] http://lisa.math.uni-magdeburg.de/index.php

[27] Subanatarajan Subbiah, Christian Schoppmeyer, Sebastian Engell, 'Efficient Scheduling of Batch Plants Using Reachability Tree Search for Timed Automata with Lower Bound Computations', 21st European Symposium on Computer Aided Process Engineering, 2011

[28] Subanatarajan Subbiah and Sebastian Engell, 'Short-Term Scheduling of Multi-Product Batch Plants with Sequence-Dependent Changeovers Using Timed Automata Models', 20th European Symposium on Computer Aided Process Engineering, 2010