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

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΑΙ ΕΠΙΛΥΣΗ ΣΥΝΘΕΤΩΝ ΠΡΟΒΛΗΜΑΤΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΕΡΓΑΣΙΩΝ ΜΕ ΧΡΗΣΗ ΧΡΟΝΙΣΜΕΝΩΝ ΑΥΤΟΜΑΤΩΝ – ΕΦΑΡΜΟΓΗ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ ΠΑΡΑΓΩΓΗΣ

Διδακτορική Διατριβή

Δημήτριος Π. Πανόπουλος

Επιβλέπων:

Ιωάννης Ψαρράς

Καθηγητής ΕΜΠ

Αθήνα, Ιανουάριος 2014

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

ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΚΑΙ ΕΠΙΛΥΣΗ ΣΥΝΘΕΤΩΝ ΠΡΟΒΛΗΜΑΤΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΕΡΓΑΣΙΩΝ ΜΕ ΧΡΗΣΗ ΧΡΟΝΙΣΜΕΝΩΝ ΑΥΤΟΜΑΤΩΝ – ΕΦΑΡΜΟΓΗ ΣΕ ΔΥΝΑΜΙΚΑ ΔΙΚΤΥΑ ΠΑΡΑΓΩΓΗΣ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Δημήτριος Π. Πανόπουλος

Συμβουλευτική Επιτροπή: Ιωάννης Ψαρράς, Καθηγητής ΕΜΠ
Ι.-Ε. Σαμουηλίδης, Ομοτ. Καθηγητής ΕΜΠ
Γρηγόριος Μέντζας, Καθηγητής ΕΜΠ

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή, στις 15 Ιανουαρίου 2014

……	……	……
Ιωάννης Ψαρράς	Ι.-Ε. Σαμουηλίδης	Γρηγόριος Μέντζας
Καθηγητής ΕΜΠ	Ομοτ. Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ
……	……	……
Βασίλειος Ασημακόπουλος	Δημήτριος Ασκούνης	Κωνσταντίνος Μεταξιώτης
Καθηγητής ΕΜΠ	Αναπλ. Καθηγητής ΕΜΠ	Επικ. Καθηγητής Παν. Πειραιά

……
Κωνσταντίνος Κούτρας
Αναπλ. Καθηγητής Παν. Πελοποννήσου

Αθήνα, Ιανουάριος 2014

.....
Δημήτριος Π. Πανόπουλος
Μηχανολόγος Μηχανικός
Διδάκτωρ Ε.Μ.Π.

ΠΡΟΛΟΓΟΣ

Η παρούσα διατριβή εκπονήθηκε στο Εργαστήριο Συστημάτων Αποφάσεων και Διοίκησης του Εθνικού Μετσοβίου Πολυτεχνείου υπό την επίβλεψη του Διευθυντή του Εργαστηρίου, Καθηγητή Ι. Ψαρρά και με την υποστήριξη των μελών της συμβουλευτικής μου επιτροπής, του Ομότ. Καθηγητή Ι.-Ε. Σαμουηλίδη και του Καθηγητή Γ. Μέντζα.

Νιώθοντας ιδιαίτερη ευγνωμοσύνη για την πολύτιμη συμβολή του στην ολοκλήρωσή της διατριβής μου, θα ήθελα να ευχαριστήσω πρωταρχικά τον Καθηγητή μου κ. Ι. Ψαρρά, τόσο για την καθοδήγησή του σε επιστημονικό επίπεδο όσο και για τη γενικότερη υποστήριξη που μου παρείχε και συνεχίζει να μου παρέχει στο πλαίσιο της παρουσίας μου στο Εργαστήριο Συστημάτων Αποφάσεων και Διοίκησης του οποίου νιώθω πλέον αναπόσπαστο μέλος.

Επιπρόσθετα, ιδιαίτερες ευχαριστίες θα ήθελα να απευθύνω στα μέλη της επταμελούς επιτροπής εξέτασης της διδακτορικής μου διατριβής, και συγκεκριμένα:

- ✓ Στον Ομοτ.Καθηγητή κ. Ι.-Ε. Σαμουηλίδη, η γνωριμία μου με τον οποίο οδήγησε στην ένταξη μου στο ΕΣΑΔ, για την πολυδιάστατη γνώση και τη φιλοσοφία αντιμετώπισης ζητημάτων την οποία μου μετέφερε στο πλαίσιο της συνεργασίας μας στις δραστηριότητες του εργαστηρίου.
- ✓ Στον Αναπλ. Καθηγητη κ. Δ. Ασκούνη, με τον οποίο είχα άριστη και στενή συνεργασία καθ' όλη τη διάρκεια της παρουσίας μου στο εργαστήριο στο πλαίσιο των ερευνητικών και ακαδημαϊκών του δραστηριοτήτων και του οποίου οι συμβουλές του για την εργασία μου υπήρξαν ιδιαίτερα πολύτιμες.
- ✓ Στον Καθηγητή κ. Γ. Μέντζα, μέλος της τριμελούς μου επιτροπής, για την πολύτιμη συμβολή του στην ολοκλήρωση της διατριβής μου και για τη μετάδοση πολύτιμων γνώσεων στο πεδίο της διοίκησης έργων τις οποίες αξιοποίησα στο πλαίσιο της διατριβής.
- ✓ Στον Αναπλ. Καθηγητή του Πανεπιστημίου Πελοποννήσου κ. Κ. Κούτρα με τον οποίο είχα την ευκαιρία να συνεργαστώ στενά σε ερευνητικό επίπεδο τα τελευταία χρόνια αποκομίζοντας ουσιαστικές γνώσεις σε προσεγγίσεις αντιμετώπισης σύνθετων προβλημάτων.
- ✓ Στον Επικ. Καθηγητή του Πανεπιστημίου Πειραιά κ. Κ. Μεταξιώτη με τον οποίο συνεργάστηκα στα πρώτα στάδια εκπόνησης της διατριβής μου και

στην κοινή ερευνητική μας προσπάθεια βασίστηκε ένα σημαντικό τμήμα της μεθοδολογίας που προτείνεται στην παρούσα εργασία.

- ✓ Στον Καθηγητή κ. Β. Ασημακόπουλο ο οποίος υπήρξε καθ' όλη τη διάρκεια της παρουσίας μου στο εργαστήριο ιδιαίτερα πρόθυμος να με υποστηρίξει σε ζητήματα που άπτονται των επιστημονικών πεδίων με τα οποία ασχολείται σε ακαδημαϊκό και ερευνητικό επίπεδο.

Θα ήθελα, πρόσθετα, να ευχαριστήσω τους φίλους και συνεργάτες μου στο Εργαστήριο με τους οποίους νιώθω ότι ανήκουμε πλέον σε μία μεγάλη οικογένεια: το Σωτήρη και τη Φεναρέτη με τους οποίους για μεγάλο διάστημα έχουμε καθημερινή επαφή, συνεργασία και στενή φιλία, τον Όθωνα και το Γιώργο με τους οποίους συνεργάστηκα για αρκετά χρόνια κατά την κοινή μας παρουσία στο εργαστήριο, την Ιωάννα, τον Παναγιώτη, τον Κώστα, τον Ιωσήφ, το Μιχάλη, τη Χριστίνα, το Χάρη, τη Ράνια και τους λοιπούς συναδέλφους μου, μεταπτυχιακούς φοιτητές και συνεργάτες του εργαστηρίου με τους οποίους συνεργαζόμαστε τα τελευταία χρόνια και έχουμε με όλους αναπτύξει σχέσεις φιλίας και αμοιβαίας εκτίμησης.

Σε προσωπικό επίπεδο θα ήθελα να ευχαριστήσω απεριόριστα τους γονείς μου Παναγιώτη και Ιωάννα και την αδερφή μου Κατερίνα για την αγάπη τους, τη συμπαράστασή τους και … την υπομονή, όσο και την υποστήριξή τους, καθ' όλη την πορεία της ζωής και των σπουδών μου.

Για το τέλος φύλαξα ένα πολύ μεγάλο ευχαριστώ για τη σύζυγό μου Κατερίνα, η συμβολή της οποίας τόσο μέσω της επιστημονικής της υποστήριξής όσο και μέσω της υπομονής, της επιμονής και της αγάπης της υπήρξε ανεκτίμητη για να καταφέρω να ολοκληρώσω τη διδακτορική μου διατριβή. Στην Κατερίνα οφείλω ένα ακόμα ευχαριστώ για τον ερχομό στον κόσμο της κόρης μου, η γέννηση της οποίας πριν λίγους μήνες αποτέλεσε ένα ισχυρό πρόσθετο κίνητρο για να εστιάσω στη διατριβή μου και να καταφέρω σήμερα να την ολοκληρώσω με επιτυχία.

ΠΕΡΙΛΗΨΗ ΔΙΑΤΡΙΒΗΣ

Η παρούσα διδακτορική διατριβή έχει ως αντικείμενο την ανάπτυξη μιας ολοκληρωμένης καινοτομικής προσέγγισης για την επίλυση προβλημάτων χρονικού προγραμματισμού παραγωγής με αξιοποίηση των υπολογιστικών δυνατοτήτων που παρέχουν τα αυτόματα πεπερασμένων καταστάσεων με χρόνους (χρονισμένα αυτόματα). Η διατριβή εδράζεται σε ένα επιστημονικό πεδίο το οποίο έχει αρχίσει να αναπτύσσεται την τελευταία μόλις δεκαετία και η έρευνα για την περαιτέρω αξιοποίησή του βρίσκεται σε πλήρη εξέλιξη. Οι δυνατότητες που παρέχουν τα χρονισμένα αυτόματα σε συνδυασμό με την ανάπτυξη εργαλείων που επιτρέπουν τη μοντελοποίηση σύνθετων συστημάτων με βάση τις αρχές της θεωρίας αυτομάτων, αποτέλεσαν την αφορμή για να ελεγχθεί κατά πόσο θα μπορούσε μια τέτοια σύγχρονη προσέγγιση να εφαρμοστεί αποτελεσματικά στην αντιμετώπιση προβλημάτων του άπτονται του χρονικού προγραμματισμού εργασιών.

Στο παραπάνω πλαίσιο, η παρούσα διατριβή προτείνει μια ολοκληρωμένη προσέγγιση για τη μοντελοποίηση προβλημάτων χρονικού προγραμματισμού εργασιών η οποία βασίζεται στην απεικόνιση των διακριτών καταστάσεων στις οποίες περιέρχεται ένα παραγωγικό σύστημα κατά την εκτέλεση των διεργασιών του. Ιδιαίτερη έμφαση δόθηκε στην μοντελοποίηση σύνθετων παραγωγικών συστημάτων που αποκλίνουν των κλασικών παραγωγικών δομών στις οποίες επικεντρώνεται η πλειοψηφία των προσεγγίσεων χρονοπρογραμματισμού εργασιών.

Πιο συγκεκριμένα, στο πλαίσιο της διατριβής, πραγματοποιήθηκε εκτενής μελέτη των εφαρμογών των χρονισμένων αυτομάτων σε προβλήματα της παραγωγής και αναπτύχθηκε μια νέα προσέγγιση για τη μοντελοποίηση παραγωγικών συστημάτων σύνθετης δομής και για την βέλτιστη επίλυση προβλημάτων χρονοπρογραμματισμού με αξιοποίηση αλγορίθμων προσπελασιμότητας. Σημαντική καινοτομία της διατριβής αποτελεί τόσο η δυνατότητα που παρέχει η προτεινόμενη προσέγγιση για την επίλυση σύνθετων προβλημάτων που δε μπορούν να προσεγγισθούν από κλασικές μεθόδους δρομολόγησης εργασιών σε κέντρα εργασίας όσο και η προσθήκη στα μοντέλα των αυτομάτων του στοιχείου του κόστους. Με τον τρόπο αυτό έγινε εφικτή η αντιμετώπιση και βέλτιστη επίλυση προβλημάτων στα οποία ο στόχος βελτιστοποίησης δεν είναι συνάρτηση αποκλειστικά του χρόνου αλλά και μιας σειράς διαφορετικών τύπων κόστους που

υπεισέρχονται στην παραγωγική διαδικασία, όπως τα καθαρά κόστη κατεργασίας, τα κόστη μεταφοράς και αποθήκευσης, τα κόστη καθυστέρησης παράδοσης αλλά και ειδικές περιπτώσεις όπως τα κόστη διακοπής και επανεκκίνησης μιας γραμμής παραγωγής.

Σε αυτό τον άξονα αναπτύχθηκαν μοντέλα χρονισμένων αυτομάτων με κόστη, τα οποία επιτρέπουν τη μοντελοποίηση και επίλυση προβλημάτων ανεξάρτητα της δομής του παραγωγικού συστήματος και των περιορισμών που τη διέπουν, όπως προβλημάτων με παράλληλες μηχανές / γραμμές παραγωγής, με εναλλακτικά φασεολόγια ή με εργασίες που μπορούν να διακόπτονται και να επανακάμπτουν. Τα δε κριτήρια βελτιστοποίησης τα οποία μπορούν να χρησιμοποιηθούν για την επίλυση των προβλημάτων με την προτεινόμενη προσέγγιση δεν περιορίζονται στα κριτήρια χρονικής διάστασης, όπως ο συνολικός χρόνος ολοκλήρωσης των εργασιών, αλλά αντίθετα εμπλέκουν ποικίλα στοιχεία κόστους, παρέχοντας τη δυνατότητα εξεύρεσης βέλτιστων χρονοπρογραμμάτων τα οποία περιορίζουν είτε το συνολικό παραγωγικό κόστος είτε επιλεγμένα στοιχεία κόστους, ανάλογα με το πρόβλημα.

Στη συνέχεια της διατριβής, η προτεινομένη προσέγγιση επεκτάθηκε και προσαρμόστηκε κατάλληλα για την αντιμετώπιση του προβλήματος του χρονικού προγραμματισμού σε δυναμικά δίκτυα παραγωγής. Η καινοτόμος πρόταση της διατριβής αντιμετωπίζει τα δεδομένα προβλήματα συντονισμού μεταξύ των παραγωγικών μονάδων των επιχειρήσεων που συμμετέχουν σε ένα δυναμικό δίκτυο παραγωγής και μοντελοποιεί τα στοιχεία κόστους που πρέπει να ληφθούν υπόψη προκειμένου να βελτιστοποιηθεί η λειτουργία του όλου δικτύου. Έτσι, η προτεινόμενη από τη διατριβή προσέγγιση παρέχει τη δυνατότητα αξιοποίησης των ισχυρών υπολογιστικών πλεονεκτημάτων των χρονισμένων αυτομάτων για την αποτελεσματική επίλυση προβλημάτων προγραμματισμού ανάθεσης εργασιών και εκτέλεσης παραγωγικών διεργασιών σε επίπεδο δικτύου συνεργαζόμενων επιχειρήσεων.

Η προτεινόμενη προσέγγιση εφαρμόστηκε με απόλυτη επιτυχία, στο πλαίσιο της διατριβής, σε μια πραγματική περίπτωση παραγωγικού δικτύου βάσει στοιχείων τα οποία διατέθηκαν από βιομηχανική επιχείρηση κατασκευής λεβήτων θέρμανσης η οποία δραστηριοποιείται στον ελληνικό χώρο και έχει δομήσει ένα συνεργατικό δίκτυο με τους βασικούς προμηθευτές της.

Εξετάζοντας στο σύνολό της την προσέγγιση που αναπτύσσεται στην παρούσα διατριβή, σε συνδυασμό με τις σύγχρονες τάσεις στον τομέα της παραγωγής που αξιοποιούν συστήματα αυτομάτων για την αντιμετώπιση διαφόρων ζητημάτων βελτιστοποίησης ή/και αυτομάτου ελέγχου, προκύπτει το συμπέρασμα ότι η πρόταση της διατριβής έχει σημαντική αξία όχι μόνο σε ερευνητικό αλλά και σε πρακτικό επίπεδο. Η συνεχής εξέλιξη των εργαλείων μοντελοποίησης αυτομάτων, σε συνδυασμό με το γεγονός ότι προσεγγίσεις, όπως η προτεινόμενη, συνδυάζουν την αποτελεσματικότητα και τη σχετική ευκολία μοντελοποίησης με την αυξημένη ευελιξία η οποία χαρακτηρίζει τις σύγχρονες παραγωγικές μονάδες, παρέχουν τα εχέγγυα για περαιτέρω αξιοποίηση των ερευνητικών αποτελεσμάτων και συμπερασμάτων της διατριβής όχι μόνο σε ερευνητικό αλλά και σε εμπορικό / επιχειρησιακό επίπεδο.

Λέξεις – κλειδιά:

Χρονοπρογραμματισμός Παραγωγής, Χρονισμένα Αυτόματα, Μοντελοποίηση Παραγωγικών Συστημάτων, Ανάθεση/Δρομολόγηση Εργασιών, Δυναμικά Δίκτυα Παραγωγής

ABSTRACT

The thesis aims at developing an integrated innovative approach for solving production scheduling problems by exploiting the computational capabilities of finite automata and more specifically of a new extension of them, called timed automata. The characteristics and capabilities of timed automata in conjunction with the development of tools which allow the accurate modelling of complex systems based on the principles of the automata theory were the main reasons for examining, in the framework of the thesis, whether such a modern approach could be applied effectively on problems related to job and task scheduling.

In the above context, the thesis proposes an integrated approach for modelling production scheduling problems based on the visualization of the several discrete states of a production system. Particular emphasis is given to the development of models of production systems with complex structures, which significantly differ from the usual production systems' structures on which the majority of scheduling approaches are focused.

More specifically, in the context of the thesis, a comprehensive study is presented based on existing applications of timed automata on production scheduling problems, while a new innovative approach is being proposed for modelling complex production systems in order to solve scheduling problems by utilising automata theory optimisation algorithms. A major innovation of the thesis is the addition of several types of costs to the timed automata models developed for solving job scheduling problems. This way it is possible to address and optimally solve problems that the objective function includes not only time but also cost elements involved in the production process, such as processing costs, transportation costs, storage costs, production switching costs etc.

In the above framework, several timed automata models have been developed which can be used as templates for building automata networks able to deal with complex cases like preemptive job scheduling problems and scheduling of jobs with alternative production routes. The optimization criteria that can be used to solve problems using the proposed approach are not limited to the usual time-related criteria but they involve a variety of cost elements, providing the possibility of finding optimal schedules that minimize either total production cost or specific types of costs, depending on the case or the problem.

The proposed approach is extended and adapted appropriately to address scheduling problems in Dynamic Manufacturing Networks (DMN) environments. The innovative approach, presented in the thesis, addresses the problem of the limited

coordination between producers belonging in a manufacturing network, taking into account several types of production and logistics costs, in order to optimize the operation and the production plans of the whole network.

The proposed approach has been applied successfully, in the context of the thesis, on a real manufacturing network case. More specifically, the manufacturing network examined is a network formed by a Greek Original Equipment Manufacturer (OEM), producing heating boilers, and by its main suppliers. The timed automata approach has been applied on real scheduling problems of this network and the results were very promising since the production plans produced were significantly better than the plans which were initially formed by the OEM.

By examining the approach proposed in the thesis, and by taking into account the existing trend of utilising automata networks in manufacturing environments for solving several types of problems, it can be concluded that the thesis has significant value not only in the research field but also in practice. The continuous development of automata modelling tools in conjunction with the fact that the proposed approach provides an easy-to-apply method for modelling complex production systems and solving scheduling problems guarantee that the outcomes of the thesis can be further exploited both in research and commercial level.

Keywords:

Production Scheduling, Timed Automata, Production Systems Modelling, Job Assignment, Dynamic Manufacturing Networks

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ

Κεφάλαιο 1:
Αντικείμενο, στόχος και συμβολή της διατριβής

1.1. Εισαγωγή

Στις μέρες μας οι οικονομικο-πολιτικές αλλαγές και η αστάθεια που παρατηρούνται σε παγκόσμια κλίμακα επηρεάζουν σε μεγάλο βαθμό τη λειτουργία τόσο των μικρών όσο και των μεγαλύτερων επιχειρηματικών οργανισμών με παραγωγική δραστηριότητα. Ειδικά τα τελευταία χρόνια έχουν λάβει χώρα, σε επίπεδο βιομηχανίας, πλήθος σημαντικών αλλαγών που διαφοροποίησαν τον τρόπο με τον οποίο ενεργούν οι επιχειρήσεις ώστε να διατηρήσουν την ανταγωνιστικότητά τους. Οι αλλαγές αυτές έχουν σχέση με την άσκηση της διοίκησης, τη χρήση της τεχνολογίας, τις προσδοκίες των πελατών, τη στάση των προμηθευτών, την ανταγωνιστική συμπεριφορά, τον ανασχεδιασμό των παραγωγικών διαδικασιών κλπ. Ως αποτέλεσμα αυτών, η ποιότητα, η αξιοπιστία και η ταχύτητα, σε συνδυασμό με το ελάχιστο κόστος, έχουν καταστεί πλέον βασικές απαιτήσεις για μια σύγχρονη επιχείρηση ώστε να επιτύχει στον χώρο της. Οι επιχειρήσεις που επιτυγχάνουν συνεχώς να εκσυγχρονίζονται και να προσαρμόζονται στις μεταβαλλόμενες συνθήκες που διαμορφώνονται στο διεθνές επιχειρηματικό και βιομηχανικό περιβάλλον, αποκτούν σημαντικά ανταγωνιστικά πλεονεκτήματα σε όποια αγορά και αν δραστηριοποιούνται. Για τις βιομηχανικές και

βιοτεχνικές επιχειρήσεις – ευρύτερα για τις επιχειρήσεις οι οποίες διαθέτουν ή διοικούν παραγωγικές μονάδες – η διαδικασία προσαρμογής στο μεταβαλλόμενο οικονομικό περιβάλλον περνάει κατά κόρον μέσα από την εξεύρεση και υιοθέτηση νέων τρόπων οργάνωσης και διοίκησης της παραγωγής αλλά και μέσα από το βελτίωση του συντονισμού των παραγωγικών τους συστημάτων με αυτά των συνεργατών και των προμηθευτών τους. Τα ολοένα και υψηλότερα επίπεδα εισχώρησης υπολογιστικών συστημάτων σε όλες τις επιχειρησιακές διαδικασίες και ειδικότερα στις διεργασίες της παραγωγής και της εφοδιαστικής αποτελούν ισχυρά όπλα για μια επιχείρηση προκειμένου να καταστεί περισσότερο αποδοτική και ευπροσάρμοστη σε νέα δεδομένα και κατ' επέκταση να αποκτήσει σημαντικά πλεονεκτήματα έναντι των ανταγωνιστών της.

Στο παραπάνω πλαίσιο, έχει ανοιχθεί ένας νέος ορίζοντας στον τομέα του προγραμματισμού της παραγωγής ο οποίος έχει δύο άξονες:

- Ο πρώτος άξονας αφορά στην ανάπτυξη νέων μεθόδων χρονοπρογραμματισμού εργασιών με έμφαση στο κόστος και αξιοποίηση των δυνατοτήτων που παρέχουν τα σύγχρονα πληροφοριακά εργαλεία. Με κατάλληλη αξιοποίηση της τεχνολογίας μέσω μοντέλων και μεθοδολογιών που αναπαριστούν τις πραγματικές συνθήκες είναι δυνατή η επίλυση πραγματικών προβλημάτων χωρίς να απαιτείται προηγούμενη προσαρμογή τους σε υφιστάμενα μοντέλα η οποία εκ των πραγμάτων υπόκειται σε περιορισμούς και μια σειρά παραδοχών, όχι πάντοτε ρεαλιστικών.
- Ο δεύτερος άξονας αφορά στην επέκταση του χρονοπρογραμματισμού σε επίπεδο δικτύου παραγωγής. Με δεδομένη την τάση των επιχειρήσεων να σχηματίζουν δίκτυα παραγωγής, με δυναμικό συχνά χαρακτήρα, στοχεύοντας τόσο σε οικονομίες κλίμακας όσο και στην επικέντρωση όλων των προσπαθειών κάθε επιχείρησης στις περιοχές όπου μπορεί να προσδώσει αξία στα προϊόντα, η έννοια του συντονισμού των επιμέρους παραγωγικών μονάδων αποκτά ιδιαίτερη αξία. Η έρευνα στον τομέα αυτό έχει πολλές διαστάσεις που ξεκινούν από το καθαρά προγραμματιστικό κομμάτι και καταλήγουν σε ζητήματα εμπιστευτικότητας και διαχείρισης του ανταγωνισμού.

1.2. Αντικείμενο και στόχος της διατριβής

Το αντικείμενο της παρούσας διατριβής σχετίζεται άμεσα με τους δύο άξονες ανάπτυξης της έρευνας στον τομέα του προγραμματισμού παραγωγής, όπως περιγράφηκαν ανωτέρω. Συγκεκριμένα η διατριβή βασίζεται σε μια προσέγγιση που έχει προταθεί κατά την τελευταία δεκαετία και αφορά στη χρήση αυτομάτων πεπερασμένων καταστάσεων για τη μοντελοποίηση συστημάτων παραγωγής και την επίλυση προβλημάτων που σχετίζονται με την παραγωγική διαδικασία με αξιοποίηση μεθόδων και αλγορίθμων ελέγχου προσπελασιμότητας. Τα αυτόματα πεπερασμένων καταστάσεων και συγκεκριμένα μια επέκταση αυτών που προσθέτει στις καταστάσεις χρόνους και βάρη και τα αποκαλούμε «χρονισμένα» αυτόματα διαθέτουν σημαντικά πλεονεκτήματα όσον αφορά στην ευελιξία που παρέχουν κατά τη μοντελοποίηση συστημάτων κάθε βαθμού πολυπλοκότητας χωρίς να απαιτούνται ιδιαίτερες παραδοχές και απλοποιήσεις. Η έρευνα στο πεδίο αυτό είναι σε πλήρη εξέλιξη και καθημερινά παρουσιάζονται όλο και πιο σύνθετες προσεγγίσεις που αξιοποιούν τα αυτόματα για τη μοντελοποίηση (αρχικά) και την επίλυση (στη συνέχεια) προβλημάτων χρονικού προγραμματισμού. Παρά το γεγονός, εν τούτοις, ότι μέχρι σήμερα έχουν προταθεί προσεγγίσεις αξιοποίησης σε προβλήματα προγραμματισμού παραγωγής των αυτομάτων πεπερασμένων καταστάσεων, υπάρχουν δύο κατευθύνσεις στις οποίες δεν έχει δοθεί μέχρι τώρα ιδιαίτερη έμφαση από τους ερευνητές που εργάζονται στον τομέα αυτό. Η πρώτη κατεύθυνση σχετίζεται με τη φύση των προβλημάτων χρονοπρογραμματισμού που αντιμετωπίζονται. Συγκεκριμένα, όλες οι εργασίες που έχουν μέχρι σήμερα παρουσιαστεί επικεντρώνονται στην ελαχιστοποίηση του χρόνου ολοκλήρωσης των εργασιών ή/και στην προσπάθεια ανάπτυξης χρονοπρογραμμάτων που να διασφαλίζουν την τήρηση προθεσμιών παράδοσης. Παρά το γεγονός, όμως, ότι η χρονική διάσταση στον προγραμματισμό της παραγωγής έχει εκ των ουκ άνευ μεγάλη σημασία, στην πράξη οι σύγχρονες επιχειρήσεις δίνουν τη μεγαλύτερη έμφαση στο κομμάτι της μείωσης του κόστους η οποία μπορεί με διάφορους τρόπους να επέλθει μέσα από ένα «έξυπνο» προγραμματισμό της παραγωγής τους. Προγραμματισμό που θα λαμβάνει υπόψη όχι μόνο χρονικούς παράγοντες αλλά και παράγοντες κόστους που σχετίζονται και με το άμεσο κόστος παραγωγής όπως και με άλλα κόστη, λ.χ. με ρήτρες καθυστέρησης ή/και επιβαρύνσεις διατήρησης αποθεμάτων ή με κόστη προετοιμασίας των γραμμών παραγωγής.

Η δεύτερη κατεύθυνση στην οποία δεν έχουν παρουσιαστεί σημαντικά βήματα προόδου και που εκ των πραγμάτων υπάρχει σχετικό πεδίο ανάπτυξης, αφορά στην

επέκταση των μοντέλων για το χρονικό προγραμματισμό παραγωγής με χρήση αυτομάτων σε δίκτυα συνεργαζόμενων επιχειρήσεων. Η ιδέα αφορά στη δυνατότητα κεντρικού προγραμματισμού της παραγωγής επιχειρήσεων που ανήκουν σε επιχειρηματικά δίκτυα και έχουν συνάψει συνεργασίες σε επίπεδο παραγωγής προκειμένου να επωφεληθούν των πλεονεκτημάτων εξοικονόμησης χρόνου και κόστους που προσφέρει η οργάνωση αυτή.

Η προτεινόμενη διατριβή επικεντρώνεται ακριβώς στις δύο κατευθύνσεις που αναλύθηκαν παραπάνω στοχεύοντας στην ανάπτυξη μοντέλων χρονισμένων αυτομάτων πεπερασμένων καταστάσεων τα οποία θα:

- έχουν τη δυνατότητα να αναπαραστήσουν παραγωγικά συστήματα κάθε δομής, χωρίς να περιορίζονται στις κλασικές δομές (jobshop, flowshop κλπ) στις οποίες επικεντρώνεται η πλειοψηφία των εργασιών στον τομέα αυτό,
- αναπαριστούν όλα τα κέντρα στα οποία υπεισέρχονται κόστη στην παραγωγική διαδικασία και θα επιτρέπουν την παραγωγή βελτιστοποιημένων προγραμμάτων με αντικειμενικό στόχο τη μείωση του κόστους είτε συνολικά είτε επικεντρωμένα (πχ ελαχιστοποίηση του κόστους αποθήκευσης),
- παρέχουν την δυνατότητα συνδυαστικής μοντελοποίησης των ανεξάρτητων παραγωγικών μονάδων που συμμετέχουν σε ένα παραγωγικό δίκτυο και τη βελτιστοποίηση ως προς διαφορετικές παραμέτρους του προγράμματος παραγωγής όλου του δικτύου, λαμβάνοντας παράλληλα υπ’ οψη τις επιμέρους ανάγκες, απαιτήσεις και περιορισμούς που προέρχονται από κάθε συνεργαζόμενο μέρος.

1.3. Συμβολή της διατριβής

Η συμβολή της διατριβής στην ερευνητική κοινότητα ουσιαστικά πηγάζει από τις ερευνητικές κατευθύνσεις και τα κενά που έρχεται να καλύψει στο επιστημονικό πεδίο της αξιοποίησης αυτομάτων πεπερασμένων καταστάσεων για τον βέλτιστο προγραμματισμό της παραγωγής τόσο αυτόνομων μονάδων όσο και δυναμικών δικτύων παραγωγής. Συγκεντρωτικά, η παρούσα διατριβή συμβάλλει ουσιαστικά:

- Στην ανάπτυξη πρότυπων μοντέλων που επιτρέπουν την ακριβή αναπαράσταση σύνθετων παραγωγικών δομών με χρήση αυτομάτων και μπορούν να ενσωματώσουν όλα τα κέντρα κόστους μιας παραγωγικής μονάδας.
- Στην ανάπτυξη τεχνικών για την εξεύρεση βέλτιστων προγραμμάτων παραγωγής ως προς σύνθετες αντικειμενικές συναρτήσεις που εμπλέκουν τόσο τα διάφορα κόστη παραγωγής όσο και τη χρονική διάσταση. Σημειώνεται ότι η ανάπτυξη σύνθετων μοντέλων που αναπαριστούν πλήρως ένα πραγματικό παραγωγικό σύστημα αυξάνει αντίστοιχα τη δυσκολία επίλυσης. Στο πλαίσιο αυτό προτείνονται προσεγγίσεις περιορισμού του χώρου λύσεων προκειμένου να είναι εφικτή η επίλυση προβλημάτων στα σύνθετα παραγωγικά μοντέλα.
- Στην ανάπτυξη μιας μεθοδολογίας για τον κεντρικό προγραμματισμό της παραγωγής σε δυναμικά δίκτυα παραγωγής, λαμβάνοντας υπόψη τόσο τα κόστη του κάθε μέλους του δικτύου όσο και τους περιορισμούς που ενδεχόμενα θέτει η παράλληλα αυτόνομη λειτουργία των επιχειρήσεων και εκτός των παραγωγικών δικτύων. Η διατριβή προτείνει ένα μοντέλο δύο επιπέδων προγραμματισμού - σε επίπεδο δικτύου και σε επίπεδο παραγωγικής μονάδας - το οποίο αναπαριστάται με αντίστοιχα δίκτυα αυτομάτων πεπερασμένων καταστάσεων που συγχρονίζονται μεταξύ τους. Η συγκεκριμένη προσέγγιση έρχεται να συνεισφέρει σε ένα ακόμα διαπιστωμένο κενό σε ότι αφορά στα αυτόματα, καθώς η αξιοποίησή τους, με βάση τη βιβλιογραφική επισκόπηση που πραγματοποιήθηκε, περιορίζεται μέχρι σήμερα αποκλειστικά σε επίπεδο αυτόνομης παραγωγικής μονάδας χωρίς να μοντελοποιείται η πολυπλοκότητα που πηγάζει από τη συμμετοχή σε ένα παραγωγικό δίκτυο.
- Στην ευρύτερη ενίσχυση της έρευνας στην κατεύθυνση της περαιτέρω ανάπτυξης του επιστημονικού πεδίου του χρονοπρογραμματισμού της παραγωγής με αξιοποίηση των τεράστιων υπολογιστικών δυνατοτήτων που παρέχουν τα αυτόματα πεπερασμένων καταστάσεων και τα εργαλεία που τα υποστηρίζουν. Πρόκειται για ένα πεδίο το οποίο έχει την τελευταία δεκαετία ουσιαστικά αναπτυχθεί καθώς, παρά το γεγονός ότι σε θεωρητικό επίπεδο οι βασικές του αρχές είχαν αποτυπωθεί σε προηγούμενο χρόνο, δεν υπήρχαν μέχρι πρόσφατα πληροφοριακά εργαλεία που να μπορούσαν να υποστηρίξουν τη μοντελοποίηση και επίλυση σύνθετων συστημάτων. Η διατριβή αναμένεται να συμβάλλει ουσιαστικά στην περαιτέρω ανάπτυξη του συγκεκριμένου πεδίου και των προοπτικών του αλλά και να καταδείξει ότι πρόκειται για μια προσέγγιση

που μπορεί να βρει άμεσα βιομηχανική εφαρμογή και να μην περιορίζεται σε ερευνητικό επίπεδο – κάτι που ήδη ισχύει στη βιομηχανία σε ότι αφορά στην αξιοποίηση αυτομάτων πεπερασμένων καταστάσεων για τον αυτόματο έλεγχο βιομηχανικών συστημάτων.

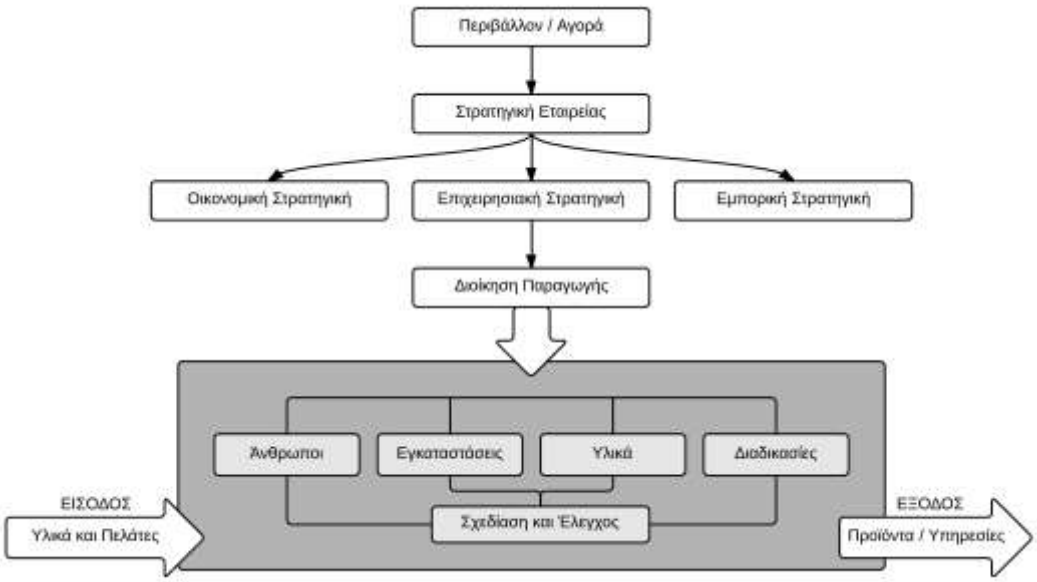
Κεφάλαιο 2:
Βασικές έννοιες προγραμματισμού παραγωγής

2.1. Διοίκηση παραγωγικών συστημάτων

Η επιστημονική περιοχή της διοίκησης παραγωγικών συστημάτων, στην οποία εντάσσεται η παρούσα διατριβή, έχει ως αντικείμενο την υποστήριξη σε επιστημονικό, στρατηγικό και ερευνητικό επίπεδο της προσπάθειας των επιχειρήσεων να βελτιστοποιήσουν τη λειτουργία των μονάδων παραγωγής τους. Η διοίκηση ενός παραγωγικού συστήματος επικεντρώνεται στην εξασφάλιση της παραγωγής των προϊόντων ή υπηρεσιών στις ποσότητες που απαιτούνται, σύμφωνα με ορισμένες ποιοτικές προδιαγραφές, σε συγκεκριμένες προθεσμίες και με το μικρότερο δυνατό κόστος, λαμβάνοντας υπόψη κάθε είδους περιορισμούς που προέρχονται από το εσωτερικό και εξωτερικό περιβάλλον. Οι αποφάσεις που λαμβάνονται στα πλαίσια της οργάνωσης ενός παραγωγικού συστήματος διακρίνονται σε στρατηγικές τακτικές και λειτουργικές. Οι πρώτες αφορούν θέματα σχεδιασμού παραγωγικών συστημάτων και έχουν μακροπρόθεσμες επιπτώσεις, ενώ οι τακτικές και λειτουργικές αφορούν στην οργάνωση και τον έλεγχο της λειτουργίας της παραγωγικής μονάδας, με τις τακτικές να έχουν βραχυπρόθεσμες επιπτώσεις και τις λειτουργικές άμεσες, σε επίπεδο καθημερινής λειτουργίας. Απαραίτητη προϋπόθεση για ένα επιτυχημένο σύστημα

διοίκησης είναι η ικανοποίηση των γενικών στόχων της επιχείρησης/οργανισμού και των ειδικών στόχων του παραγωγικού συστήματος. Οι στόχοι αυτοί σχετίζονται με την βέλτιστη χρήση των παραγωγικών πόρων, γεγονός που σημαίνει ελαχιστοποίηση του κόστους και βελτιστοποίηση της ποιότητας των προϊόντων με ταυτόχρονη τήρηση των προθεσμιών παράδοσής τους [170].

Στο σχήμα που ακολουθεί παρουσιάζεται το κλασικό μοντέλο ενός παραγωγικού συστήματος και οι βασικές αλληλεπιδράσεις του με το περιβάλλον. Φυσικά, τα παραγωγικά συστήματα των σύγχρονων επιχειρήσεων μπορεί να έχουν διαφορετικές δομές, συχνά ιδιαίτερα πολύπλοκες. Οι συνηθέστερες εξ αυτών, οι οποίες συναντώνται στις μέρες μας στην πλειονότητα των επιχειρήσεων παρουσιάζονται στην επόμενη παράγραφό. Εν τούτοις, ανεξάρτητα από την εσωτερική οργάνωση του παραγωγικού συστήματος, σε κάθε περίπτωση οι βασικές διεπαφές του τόσο με το εσωτερικό όσο και με το εξωτερικό περιβάλλον της επιχείρησης κατά βάση δε διαφέρουν από τις αναφερόμενες στο Σχήμα 1.



Σχήμα 1: Κλασικό μοντέλο παραγωγικού συστήματος [40]

2.2. Κατηγοριοποίηση παραγωγικών συστημάτων

Τα παραγωγικά συστήματα αποτελούνται συνήθως από κάποια κύρια υποσυστήματα, τα οποία εκτελούν τις κύριες παραγωγικές λειτουργίες και από κάποια δευτερεύοντα υποσυστήματα που υποστηρίζουν τα πρώτα. Για παράδειγμα, ένα

εργοστάσιο περιλαμβάνει τα παραγωγικά τμήματα της παραγωγής των τελικών προϊόντων - εκείνα δηλαδή στα οποία επιτελείται η κύρια λειτουργία του συστήματος "εργοστάσιο". Η λειτουργία αυτή υποστηρίζεται από τα υποσυστήματα προμηθειών, μεταφορών, επικοινωνιών, λογιστικής, δικτύων ενέργειας, νερού, κλπ. Τα δευτερεύοντα υποσυστήματα αποτελούν παραγωγικά συστήματα που με τη λειτουργία τους (λειτουργία υποστήριξης) υποστηρίζουν τις κύριες λειτουργίες (λειτουργίες γραμμής) ενός συστήματος [135].

Τα συστήματα παραγωγής σε ένα βιομηχανικό περιβάλλον μπορούν να χαρακτηρισθούν από παράγοντες όπως ο αριθμός των μηχανών, η σύνθεση ή η διάταξη των τμημάτων που τις απαρτίζουν, τα χαρακτηριστικά τους, το επίπεδο αυτοματοποίησης, το σύστημα διαχείρισης των μηχανών κλπ. Οι διαφοροποιήσεις σε αυτά τα χαρακτηριστικά είναι υπεύθυνες για την ύπαρξη διαφόρων τύπων συστημάτων παραγωγής και μεθόδων οργάνωσης αυτών. Δεδομένου ότι το αντικείμενο της παρούσας διατριβής επικεντρώνεται στο ζήτημα του χρονικού προγραμματισμού παραγωγικών συστημάτων, στη συνέχεια παρουσιάζεται μια κατηγοριοποίηση αυτών με βάση τη δομή των παραγωγικών πόρων τους (μηχανών και ανθρώπων) και του τρόπου με τον οποίο λειτουργούν κατά την παραγωγή προϊόντων. Με βάση αυτό το σκεπτικό τα συστήματα παραγωγής κατηγοριοποιούνται σε μια σειρά βασικών κατηγοριών, η κάθε μία εκ των οποίων απαιτεί διαφορετική αντιμετώπιση του προβλήματος του χρονικού προγραμματισμού των εργασιών της. Στη συνέχεια του κεφαλαίου περιγράφονται οι πιο συνηθισμένες κατηγορίες παραγωγικών συστημάτων, ενώ γίνεται ειδική αναφορά στο σύστημα προγραμματισμού σε παρτίδες το οποίο και αντιπροσωπεύει ένα ιδιαίτερα μεγάλο τμήμα των παραγωγικών μονάδων διεθνώς και την πλειοψηφία των μικρομεσαίων παραγωγικών μονάδων που λειτουργούν στην ελληνική επικράτεια.

2.2.1. Σύστημα μίας μηχανής / θέσης παραγωγής

Η περίπτωση των συστημάτων που διαθέτουν μία σταθερή θέση παραγωγής (συνήθως μια μοναδική μηχανή) είναι η απλούστερη και αποτελεί ειδική περίπτωση των κατά βάση πιο πολύπλοκων περιπτώσεων που συναντώνται στην πράξη. Σε επίπεδο χρονικού προγραμματισμού, αυτός εστιάζεται στο πρόβλημα της δρομολόγησης της κατεργασίας ενός αριθμού *ν* εργασιών σε μια μηχανή. Το παραπάνω πρόβλημα έχει αναλυθεί με ιδιαίτερη λεπτομέρεια κάτω από όλες τις ειδικές συνθήκες και περιορισμούς καθώς και για πλήθος διαφορετικών αντικειμενικών συναρτήσεων. Το αποτέλεσμα είναι

μια σειρά από κανόνες οι οποίοι, παρόλο που είναι ενίοτε απαιτούν πληροφοριακή υποστήριξη για να εφαρμοστούν, ως επί το πλειστον παρέχουν ικανοποιητικές λύσεις σε περιβάλλοντα μίας μοναδικής μηχανής. Για παράδειγμα, ο κανόνας νωρίτερης ημερομηνίας παράδοσης (Earliest Due Date - EDD) που διατάσσει τις διεργασίες σύμφωνα με αύξουσα σειρά χρόνων προθεσμίας, ελαχιστοποιεί την μέγιστη καθυστέρηση μεταξύ των εργασιών, ενώ ο κανόνας ελάχιστου χρόνου εκτέλεσης (Shortest Processing Time - SPT) ελαχιστοποιεί τον μέσο αριθμό των εργασιών που αναμένουν για επεξεργασία.

2.2.2. Σύστημα παράλληλων μηχανών

Πρόκειται για ένα παραγωγικό σύστημα με μηχανές συνδεδεμένες παράλληλα και το οποίο έχει ιδιαίτερο ενδιαφέρον από θεωρητική και πρακτική σκοπιά. Από θεωρητικής απόψεως είναι η γενίκευση του συστήματος με μια απλή μηχανή και μια ειδική περίπτωση του ευέλικτου συστήματος ροϊκής παραγωγής. Από πρακτικής απόψεως είναι σημαντικό γιατί πολλά βιομηχανικά περιβάλλοντα αποτελούνται από ένα αριθμό διαδοχικών σταδίων παραγωγής κάθε ένα από τα οποία αποτελείται από μηχανές συνδεδεμένες παράλληλα. Το μοντέλο των παράλληλων μηχανών έχει ιδιαίτερη σημασία στην ακόλουθη περίπτωση: Αν σε ένα συγκεκριμένο στάδιο παραγωγής υπάρχει πρόβλημα συμφόρησης (bottleneck) τότε η επίδοση αυτού του σταδίου παραγωγής θα επηρεάσει την επίδοση του συνολικού συστήματος. Η συμφόρηση αυτή μπορεί συχνά να μοντελοποιηθεί σαν μια ομάδα από παράλληλες μηχανές και μπορεί να αναλυθεί διακριτά. Οι βασικές διαφοροποιήσεις που υπάρχουν στα συστήματα αυτά είναι οι ακόλουθες:

- **Ίδιες μηχανές, συνδεδεμένες παράλληλα.** Στη συγκεκριμένη περίπτωση όταν απαιτείται η επεξεργασία μιας εργασίας, αυτή μπορεί να γίνει σε όποια μηχανή είναι διαθέσιμη. Το πρόβλημα σε αυτήν τη περίπτωση μπορεί να περιγραφεί ως εξής: Υπάρχουν n εργασίες J_i (i = 1, . . ., n) με χρόνους επεξεργασίας p_i (i = 1, . . ., n) που πρέπει να υποστούν επεξεργασία σε m ίδιες μηχανές M_1, . . ., M_m που είναι συνδεδεμένες παράλληλα.
- **Μηχανές με διαφορετική ταχύτητα, συνδεδεμένες παράλληλα (ομοιόμορφες μηχανές)**. Σε αυτή τη περίπτωση οι μηχανές που συνδέονται παράλληλα έχουν διαφορετικές ταχύτητες. Αυτό μπορεί, ενδεικτικά, να οφείλεται στο ότι κάποια μηχανή είναι παλαιότερης τεχνολογίας (και συνεπώς βραδύτερη)

από μια άλλη. Αν οι μηχανές έχουν τις ίδιες ταχύτητες τότε αυτό το περιβάλλον είναι όμοιο με το προηγούμενο. Το πρόβλημα του χρονοπρογραμματισμού διαμορφώνεται ως εξής: Θεωρούμε n εργασίες, J_i (i = 1, . . ., n) που πρέπει να υποστούν επεξεργασία σε m παράλληλες μηχανές M_j (j = 1, . . ., m). Οι μηχανές έχουν διαφορετικές ταχύτητες s_j (j = 1, . . ., m). Κάθε εργασία J_i έχει μια συγκεκριμένη απαίτηση επεξεργασίας p_i (i = 1, . . ., n). Η εκτέλεση μιας εργασίας J_i σε μια μηχανή M_j απαιτεί p_i / s_j μονάδες χρόνου. Αν τεθεί $s_j = 1$ για j = 1, . . ., m προκύπτουν m παράλληλες όμοιες μηχανές.

- **Ασυσχέτιστες μηχανές, συνδεδεμένες παράλληλα**. Το περιβάλλον αυτό είναι μια γενίκευση του προηγούμενου. Υπάρχουν διαφορετικές μηχανές συνδεδεμένες παράλληλα. Μια μηχανή μπορεί να επεξεργαστεί μια εργασία με μια συγκεκριμένη ταχύτητα. Αν οι ταχύτητες των μηχανών είναι ανεξάρτητες των εργασιών, τότε αυτό το περιβάλλον είναι όμοιο με το προηγούμενο. Το προς εξέταση πρόβλημα ορίζεται ακολούθως: Έστω n ανεξάρτητες εργασίες i = 1, . . ., n που πρέπει να υποστούν επεξεργασία σε m μηχανές. Ο χρόνος επεξεργασίας κάθε εργασίας i στην μηχανή M_j είναι p_{ij} (i = 1, . . ., n; j = 1, . . ., m). Αυτό το μοντέλο είναι μια γενίκευση του μοντέλου των ομοιόμορφων μηχανών, αν τεθεί $p_{ij} = p_i / s_j$. Και σε αυτή τη περίπτωση στόχος είναι η ελαχιστοποίηση μιας αντικειμενικής συνάρτησης όπως του συνολικού χρόνου ολοκλήρωσης του συνόλου των εργασιών.

2.2.3. Σύστημα ροϊκής παραγωγής (flow shop)

Στα συστήματα του τύπου flow shop, η παραγωγή εξειδικεύεται σε ένα περιορισμένο αριθμό τυποποιημένων προϊόντων που παράγονται σε αντίστοιχες γραμμές παραγωγής. Προϊόντα που προορίζονται για ευρεία κατανάλωση κατασκευάζονται ως επί το πλείστον σε μονάδες που διαθέτουν ροϊκή παραγωγή και με βασική επιδίωξη την ελαχιστοποίηση του κόστους παραγωγής αλλά και τη σταθεροποίηση του ποιότητας των προϊόντων. Οι εργασίες περνούν από συγκεκριμένα στάδια επεξεργασίας. Τα στάδια αυτά μπορεί να είναι διακριτά (π.χ. σε μια αυτοκινητοβιομηχανία) ή συνεχή, οπότε και είναι αδύνατο να διαχωριστούν πλήρως μεταξύ τους (π.χ. στα διυλιστήρια). Όλες οι εργασίες υπόκεινται σε πολλαπλή επεξεργασία σε ένα αριθμό διαφορετικών μηχανών. Έχουν το ίδιο δρομολόγιο μέσα στο σύστημα παραγωγής: πρέπει όλες να επεξεργαστούν πρώτα από τη μηχανή 1, μετά από τη μηχανή 2 και ούτω καθεξής. Οι μηχανές είναι εγκατεστημένες εν σειρά και όταν

μια εργασία ολοκληρώνεται σε μια μηχανή προωθείται στην επόμενη. Η ακολουθία των εργασιών μπορεί να ποικίλει από μηχανή σε μηχανή, εφόσον οι εργασίες μπορεί να αναδιαταχθούν μεταξύ των μηχανών. Εντούτοις, η ίδια ακολουθία των εργασιών διατηρείται διαμέσου όλου του συστήματος εφόσον ένα σύστημα χειρισμού των υλικών μεταφέρει τις εργασίες από τη μια μηχανή στην άλλη.

Το αντίστοιχο πρόβλημα χρονοπρογραμματισμού για τα παραπάνω συστήματα συνίσταται στο να προγραμματισθούν οι εργασίες με τέτοιο τρόπο ώστε να ελαχιστοποιείται ο χρόνος στον οποίο ολοκληρώνεται η επεξεργασία τους. Μαθηματικά, δύναται να περιγραφεί ως εξής:

Έστω m μηχανές $\{M_1, \ldots, M_m\}$ και n εργασίες $\{J_1, \ldots, J_n\}$. Κάθε μια από τις n εργασίες πρέπει να επεξεργασθεί στις m μηχανές $M_1, \ldots, M_m$ με αυτή τη σειρά. Μια εργασία J_j, j = 1, . . ., n αποτελείται από μια σειρά m επιμέρους διεργασιών $O_{1j}, \ldots, O_{mj}$, όπου η O_{ij} πρέπει να υποστεί επεξεργασία στη μηχανή M_i για ένα δεδομένο χρόνο p_{ij}. Κάθε μηχανή M_i, i = 1, . . ., m μπορεί να επεξεργαστεί το πολύ μια εργασία κάθε στιγμή και κάθε εργασία J_j , j = 1, . . ., n δύναται να υπόκειται σε επεξεργασία από το πολύ μια μηχανή κάθε στιγμή. Έστω C_{ij} ο χρόνος ολοκλήρωσης της επιμέρους διεργασίας O_{ij}. Ο στόχος είναι να βρεθεί ένα χρονοπρόγραμμα που θα ελαχιστοποιεί μια αντικειμενική συνάρτηση η οποία μπορεί να σχετίζεται με τον παραπάνω χρόνο ολοκλήρωσης ή με άλλα κριτήρια (π.χ. μέση καθυστέρηση ολοκλήρωσης εργασιών).

Το παραπάνω πρόβλημα είναι δυνατό να διαφοροποιηθεί ως προς ορισμένες παραμέτρους, δημιουργώντας υποπεριπτώσεις του κλασικού συστήματος flow shop:

- **Συστήματα προσαρμοζόμενης ροϊκής παραγωγής (Non-permutation flow shop).** Σε ορισμένα συστήματα του τύπου flow shop, αν μια εργασία δεν απαιτεί την επεξεργασία από μια συγκεκριμένη μηχανή, μπορεί να την παρακάμψει και να βρεθεί μπροστά από άλλες εργασίες που υπόκεινται σε επεξεργασία ή αναμένουν για επεξεργασία εκεί. Τα συστήματα αυτά είναι θεωρούνται πιο αντιπροσωπευτικά των περιπτώσεων συστημάτων flow shop γιατί αποτελούν μια καλύτερη αναπαράσταση των πρακτικών προβλημάτων.
- **Συστήματα αυστηρά ροϊκής παραγωγής (Permutation flow shop).** Υπάρχουν συστήματα flow shop που δεν επιτρέπουν την προαναφερόμενη παράκαμψη. Σε αυτήν την περίπτωση λειτουργούν αυστηρά με βάση την πειθαρχία τύπου *first in first out (FIFO)*.

Μια γενίκευση των συστημάτων ροϊκής παραγωγής είναι αυτή της ευέλικτης ροϊκής παραγωγής (flexible flow shop), τα οποία περιλαμβάνουν ένα αριθμό σταδίων σε σειρά με ένα αριθμό μηχανών συνδεδεμένων παράλληλα σε κάθε στάδιο (συνδυασμός μηχανών συνδεδεμένων παράλληλα και flow shop). Οι εργασίες υφίστανται επεξεργασία σε κάθε στάδιο σε οποιαδήποτε από τις παράλληλες μηχανές. Οι ουρές μεταξύ των διάφορων σταδίων συνήθως λειτουργούν με βάση διάταξη τύπου FIFO. Ο παραπάνω τύπος συστήματος απαντάται μεταξύ άλλων σε βιομηχανίες καλλυντικών, τροφίμων και ενδυμάτων [40].

2.2.4. Σύστημα παραγωγής κατά παραγγελία (job shop)

Στα συστήματα job shop, κυρίαρχη έννοια είναι η παραγγελία η οποία πρέπει να διεκπεραιωθεί. Αυτή η παραγγελία αφορά τις περισσότερες φορές την παραγωγή ενός προϊόντος με συγκεκριμένες προδιαγραφές που τίθενται από τον πελάτη. Η παραγωγή γίνεται σε παρτίδες. Εφόσον οι προδιαγραφές είναι διαφορετικές ανά παραγγελία, η ροή της παραγωγής είναι επίσης διαφορετική για κάθε μια, ανεξάρτητα από το αν θα χρησιμοποιηθεί ένα συγκεκριμένο πλήθος παραγωγικών μονάδων. Τα συστήματα job shop αποτελούν γενίκευση των συστημάτων flow shop (ένα σύστημα flow shop είναι ουσιαστικά ένα σύστημα job shop στο οποίο κάθε διεργασία έχει το ίδιο δρομολόγιο).

Το πρόβλημα του χρονοπρογραμματισμού στην περίπτωση ενός συστήματος παραγωγής αυτού του τύπου, περιγράφεται ως εξής: Έστω **n** εργασίες i = 1, . . ., n και **m** μηχανές M_1, . . ., M_m. Η εργασία **i** συνίσταται από μια σειρά $\mathbf{n_i}$ επιμέρους διεργασιών O_{i1}, O_{i2}, . . ., O_{in_i} οι οποίες πρέπει να υποστούν επεξεργασία σε αυτή τη σειρά, δηλαδή υπάρχουν περιορισμοί προτεραιότητας της μορφής $\mathbf{O_{ij}} \rightarrow \mathbf{O_{i,j+1}}$ (j = 1, . . ., n_i – 1). Υπάρχει μια μηχανή $\boldsymbol{\mu_{ij}} \in \{M_1, . . ., M_m\}$ και ένας χρόνος επεξεργασίας $\mathbf{p_{ij}}$ που σχετίζεται με κάθε επιμέρους διεργασία $\mathbf{O_{ij}}$**.** Οι επιμέρους διεργασίες $\mathbf{O_{ij}}$ πρέπει να υποστούν επεξεργασία για $\mathbf{p_{ij}}$ μονάδες χρόνου στη μηχανή $\boldsymbol{\mu_{ij}}$. Το πρόβλημα είναι να βρεθεί ένας εφικτός χρονοπρογραμματισμός που ελαχιστοποιεί κάποια αντικειμενική συνάρτηση που εξαρτάται από τους χρόνους ολοκλήρωσης C_i των τελευταίων επιμέρους εργασιών O_{i,n_i}, των διεργασιών. Στη γενική περίπτωση υποτίθεται ότι $\mu_{ij} \neq \mu_{i,j+1}$ για i = 1,2, . . . n_i – 1.

Αν οι εργασίες που πρέπει να προγραμματιστούν είναι διαθέσιμες από την αρχή του προβλήματος, τότε το πρόβλημα χρονοπρογραμματισμού καλείται *στατικό*. Αν το σετ των εργασιών αλλάζει συνεχώς, τότε το πρόβλημα καλείται *δυναμικό*. Επιπλέον, αν

όλες οι παράμετροι είναι γνωστές με ακρίβεια το πρόβλημα χαρακτηρίζεται ως *αιτιοκρατικό* ή *ντετερμινιστικό*, ενώ αν τουλάχιστον μια παράμετρος καθορίζεται με πιθανότητες (όπως για παράδειγμα οι χρόνοι ολοκλήρωσης των εργασιών) τότε καλείται *στοχαστικό*. Σημειώνεται ότι οι παραπάνω διαχωρισμοί ισχύουν όχι μόνο για το jobshop αλλά και για την πλειοψηφία των προβλημάτων χρονοπρογραμματισμού [168].

Σε αυτό το σημείο, είναι σκόπιμο να αναφερθούν οι σημαντικότερες διαφοροποιήσεις του συστήματος παραγωγής job shop [18]:

- **Κλασικό (classic) job shop**: Σε ένα τυπικό σύστημα παραγωγής του τύπου classic job shop κάθε παραγγελία είναι μοναδική και έχει ένα μοναδικό δρομολόγιο. Οι διάφορες εργασίες εκτελούνται με τη σειρά σαν μια μεγάλη παρτίδα από διάφορα τμήματα που ταξιδεύουν μαζί διαμέσου του συστήματος παραγωγής. Η διαδικασία του χρονοπρογραμματισμού είναι αρκετά περίπλοκη και δεν επαναλαμβάνεται με κανένα τρόπο. Το σύστημα classic job shop αναφέρεται επίσης και ως *closed shop* γιατί οι παραγγελίες είναι διακριτές και οι εργασίες που βρίσκονται σε εξέλιξη (work–in–process ή WIP) δεν μπορούν να χρησιμοποιηθούν για άλλο σκοπό (no multi–use parts).
- **Ανοικτό (open) job shop**: Ένα σύστημα παραγωγής που παράγει προϊόντα τα οποία μπορούν να αποθηκευτούν ώστε να εξυπηρετήσουν μια μελλοντική ζήτηση και δεν απευθύνονται αποκλειστικά σε συγκεκριμένες παραγγελίες, είναι γνωστό ως open job shop. Μπορεί δηλαδή να υπάρχουν πολλοί πελάτες που ζητούν τα ίδια (ή σχεδόν τα ίδια) προϊόντα και έτσι έχει νόημα η αποθήκευση τελικών προϊόντων (ή προϊόντων των οποίων η επεξεργασία έχει σχεδόν τελειώσει) ή η εκτροπή δραστηριοτήτων που ήταν προορισμένες για ένα πελάτη, προς χάρη ενός άλλου πελάτη μεγαλύτερης προτεραιότητας (multi–use parts). Η διαδικασία του χρονοπρογραμματισμού είναι παρόμοια με αυτή των συστημάτων τύπου classic job shop με τη διαφορά ότι η κατηγοριοποίηση των εργασιών (είτε η επεξεργασία τους έχει ολοκληρωθεί είτε όχι) σύμφωνα με τον πελάτη ή με τις προθεσμίες παράδοσης, μετατρέπεται σε μια πιο δυναμική και περίπλοκη διαδικασία. Ένα σύστημα κατασκευής έτοιμων σπιτιών (επιλογή ανάμεσα σε συγκεκριμένους τύπους κατοικιών) είναι ένα πολύ καλό παράδειγμα ενός open job shop.
- **Ευέλικτο (flexible) job shop:** Αποτελεί γενίκευση του κλασικού job shop. Στη συγκεκριμένη περίπτωση, κάθε παραγωγική μονάδα αντικαθίσταται από ένα

αριθμό μηχανών συνδεδεμένων παράλληλα. Όταν μια εργασία, στη διάρκεια του δρομολογίου της, φθάσει σε αυτήν την παραγωγική μονάδα με τις παράλληλες μηχανές μπορεί να υποστεί επεξεργασία σε οποιαδήποτε από τις μηχανές. Το παραπάνω περιβάλλον είναι πολύ συνηθισμένο, για παράδειγμα, στη βιομηχανία ημιαγωγών.

2.2.5. Σύστημα παραγωγής σε παρτίδες (batch shop)

Ένα σύστημα παραγωγής του τύπου batch shop είναι ένα σύστημα στο οποίο η παραγωγή πανομοιότυπων τελικών ή ενδιάμεσων προϊόντων απαιτείται να είναι τόσο μεγάλη ώστε να προτιμάται η παραγωγή κάποιων τμημάτων σε συγκεκριμένες παρτίδες, με αποτέλεσμα να επιτυγχάνονται μεγάλες οικονομίες κλίμακος. Η ροή των εργασιών στο συγκεκριμένο σύστημα παραγωγής δεν είναι τελείως γραμμική άλλα συνήθως είναι λιγότερο πολύπλοκη από ότι στα συστήματα open και closed job shop. Ένα παράδειγμα ενός διακριτού συστήματος batch shop είναι μια βιοτεχνία ρούχων.

2.3. Χρονοπρογραμματισμός παραγωγής

Ο χρονοπρογραμματισμός της παραγωγής άπτεται μια ευρείας γκάμας οικονομικών δραστηριοτήτων. Περιλαμβάνει την εκπλήρωση παραγγελιών οι οποίες δεσμεύουν 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 την απόδοση 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 εργασίας και τον έλεγχο της παραγωγικής διαδικασίας (ανάλυση της κατάστασης εκτελούμενων εργασιών, επίσπευση καθυστερημένων και κρίσιμων εργασιών).

2.3.1. Η σημασία του χρονοπρογραμματισμού παραγωγής

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. Παλαιότερα, ο αυτόματος χρονοπρογραμματισμός για σύνθετα συστήματα παραγωγής ήταν μια καθαρά ακαδημαϊκή υπόθεση [171]. Θεωρούνταν λογικό και αυτονόητο να τηρούνται σημαντικά 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 όπως:

- η συνεχώς αυξανόμενη πολυπλοκότητα και η γρήγορη παλαίωση των προϊόντων,
- ο περιορισμός των διάφορων ελαττωμάτων που παρουσιάζονται στα προϊόντα κατά την παραγωγική διαδικασία,
- το γεγονός ότι η αγορά ζητάει συντομότερους χρόνους παράδοσης και δυνατότητες αναθεώρησης της παραγγελίας κατά την εξέλιξή της,
- η απαίτηση για ευέλικτη και άμεση αντίδραση στις επιθυμητές αλλαγές της παραγωγής καθώς και σε επείγοντα προβλήματα και
- η ανάγκη για 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 βέλτιστος χρονοπρογραμματισμός της παραγωγής τους έχει τεράστια σημασία για την επιβίωση των επιχειρήσεων στις απαιτητικές συνθήκες της αγοράς [168].

2.3.2. Παράγοντες χρονοπρογραμματισμού παραγωγής

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* 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 [171].

Εφόσον τέτοιου είδους περιορισμοί δύνανται να καθιστούν ιδιαίτερα δύσκολη την εύρεση λύσεων με ικανοποιητική ακρίβεια, είναι αναμενόμενο να οδηγείται η έρευνα στην κατεύθυνση της επίλυσης 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. Αυτό συμβαίνει διότι σημαντικοί “στόχοι” - όπως π.χ. η ικανοποίηση του πελάτη - είναι δύσκολο να ποσοτικοποιηθούν ορθά. 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 παραγωγής και εφοδιαστικής.

Μεταξύ των πιθανών προσεγγίσεων συγκαταλέγονται:

- Η επίλυση προβλημάτων με ένα συγκεκριμένο στόχο κάθε φορά.
- Η επίλυση συνδυασμών των διαφόρων στόχων μέσω καμπυλών.
- Ο 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 του χρονοπρογραμματισμού. 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, να εκμισθωθούν επιπλέον πόροι ή να μπει προσωρινά στην παραγωγή μια πεπαλαιωμένη (ανενεργή) 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. Αυτοί οι τύποι αποφάσεων εντάσσονται στην έννοια της «*διαμόρφωσης 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 [170].

2.3.3. Χρονοπρογραμματισμός και πληροφοριακά συστήματα

Έως τώρα έγινε αναφορά στις σημαντικές αποφάσεις που αφορούν στο 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 *διαχωρισμός [167]*.

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 *κατανεμημένα*. 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.

2.3.4. Το γενικευμένο πρόβλημα χρονοπρογραμματισμού

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 ιδιαίτερα μεγάλα. Είναι επόμενο να έχει δημιουργηθεί ένας συγκεκριμένος συμβολισμός που να αποδίδει με περιεκτικό τρόπο τη δομή των μοντέλων που έχουν αναπτυχθεί στη βιβλιογραφία [171].

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 μια πλειάδα εισόδων με πιο συνηθισμένες αυτές που παρουσιάζονται στον ακόλουθο πίνακα [29]:

Πεδίο α	***Κατηγορία Παραγωγικού Συστήματος***
$\mathbf{P_m}$	***Parallel machine shop #1:*** Πρόκειται για m όμοιες μηχανές που λειτουργούν παράλληλα. Μια εργασία j μπορεί να υποστεί επεξεργασία σε οποιαδήποτε μηχανή.
$\mathbf{Q_m}$	***Parallel machine shop #2:*** Πρόκειται για m όμοιες μηχανές που λειτουργούν παράλληλα, αλλά σε αυτήν την περίπτωση έχουν διαφορετικές ταχύτητες.
$\mathbf{R_m}$	***Parallel machine shop #3:*** Πρόκειται για γενίκευση του προηγουμένου με m διαφορετικές μηχανές που λειτουργούν παράλληλα.
$\mathbf{F_m}$	***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.
$\mathbf{FF_s}$	***Flexible flow shop:*** Πρόκειται για γενίκευση του προηγουμένου. Αποτελείται από έναν αριθμό βαθμίδων εν σειρά με έναν αριθμό παραλλήλων μηχανών σε κάθε βαθμίδα.
$\mathbf{J_m}$	***Job shop:*** Πρόκειται για m μηχανές. Στην προκειμένη περίπτωση κάθε εργασία ακολουθεί τη δική της ανεξάρτητη διαδρομή.
$\mathbf{O_m}$	***Open shop:*** Κάθε εργασία πρέπει να υποστεί επεξεργασία από κάθε μηχανή m. Ωστόσο, κάποιοι από τους χρόνους επεξεργασίας μπορεί να είναι μηδενικοί. Διαφορετικές εργασίες μπορεί να ακολουθούν διαφορετικές διαδρομές.

Στο *πεδίο β* ορίζονται περιορισμοί που μπορεί να περιλαμβάνουν πολλαπλές εισόδους. Μερικές από αυτές είναι οι ακόλουθες [17], [18]:

- **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 αποθηκευτικού 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.

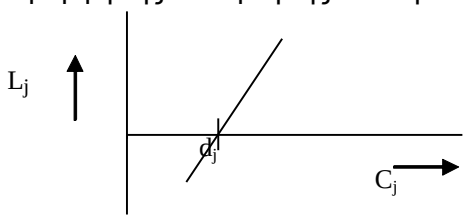
- **Machine eligibility constraints (M_j)**. Η είσοδος M_j (περιορισμοί καταλληλότητας των μηχανών) μπορεί να εμφανίζεται στο πεδίο β όταν το περιβάλλον που έχουμε είναι m μηχανές συνδεδεμένες παράλληλα (Pm). Όταν ο συμβολισμός M_j είναι παρόν, δεν είναι όλες οι m μηχανές ικανές για να επεξεργαστούν 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:

$$L_j = C_j - d_j$$

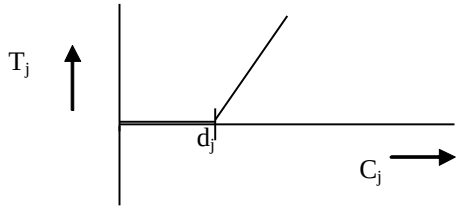
το οποίο είναι θετικό όταν η εργασία j έχει ολοκληρωθεί αργότερα και αρνητικό όταν έχει ολοκληρωθεί νωρίτερα. Η μορφή της συνάρτησης είναι η ακόλουθη:



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.

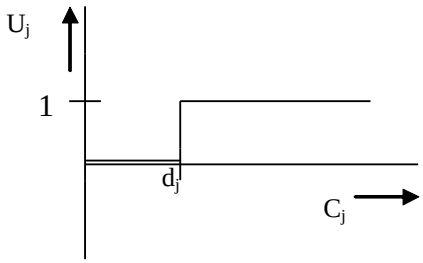
Αν μια εργασία ολοκληρωθεί πριν το χρόνο προθεσμίας της, τότε έχουμε την περίπτωση της *νωρίτερης ολοκλήρωσης* (*earliness*) που δίνεται από τη σχέση:

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

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 νωρίτερη ολοκλήρωση) 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 $\max(C_1, \ldots, C_n)$, 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 $\max(L_1, \ldots, L_n)$. 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 – $\Sigma wjUj$).** 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 παραγωγικών συστημάτων δε 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* [92].

Κεφάλαιο 3:
Επισκόπηση μεθόδων χρονικού προγραμματισμού παραγωγικών συστημάτων

3.1. Εισαγωγή

Οι σύγχρονες τάσεις της αγοράς απαιτούν μεγάλη ποικιλία προϊόντων, ταχύτερη εισαγωγή νέων και μείωση του κόστους και των τιμών. Όλα αυτά έχουν οδηγήσει με τη σειρά τους στην ανάγκη για εφαρμογή συστημάτων με μηδενικά αποθέματα. Από την άλλη μεριά βέβαια μια επιχείρηση για να πετύχει και να διατηρήσει τα μερίδιά της χρειάζεται να ανταποκρίνεται άμεσα στις μεταβολές της ζήτησης κάτι που οδηγεί στην ανάγκη για διατήρηση υψηλού επιπέδου αποθεμάτων. Αυτού του είδους οι αντιθέσεις καθιστούν επιτακτική την ανάγκη για έναν ακριβή και πιο αποτελεσματικό χρονοπρογραμματισμό παραγωγής με την εφαρμογή σύγχρονων και αποδοτικών τεχνικών. Ο χρονοπρογραμματισμός σχετίζεται με την επίλυση του Προβλήματος Βελτιστοποίησης με Περιορισμούς (Constraint Optimization Problem) από τη μεριά της παραγωγής. Έχει να κάνει με την εύρεση του τρόπου κατανομής πόρων έτσι ώστε να βελτιστοποιείται μια συγκεκριμένη αντικειμενική συνάρτηση. Το πρόβλημα του αιτιοκρατικού (ντετερμινιστικού) χρονοπρογραμματισμού job shop συστημάτων (Π_j) αποτελεί την πιο χαρακτηριστική περίπτωση των προβλημάτων χρονικού

προγραμματισμού. Παρά το γεγονός ότι η διατριβή δεν περιορίζεται στο συγκεκριμένο πρόβλημα, το job shop αποτελεί μια ισχυρή βάση πάνω στην οποία μπορούν να δομηθούν πιο σύνθετα προβλήματα προγραμματισμού. Για το λόγο αυτό στη συνέχεια αναλύεται διεξοδικότερα το συγκεκριμένο μοντέλο προβλήματος, ενώ στη συνέχεια της εργασίας θα αναφερθούν σχετικές επεκτάσεις και διαφοροποιήσεις του προβλήματος στο πλαίσιο της κάλυψης πιο σύνθετων περιπτώσεων που στην πράξη απαντώνται στη βιομηχανία.

Το πρόβλημα job shop (Π_j) περιλαμβάνει ένα πεπερασμένο σύνολο *J* που αποτελείται από *n* εργασίες $\{J_i\}_{i=1}^{n}$ που πρέπει να επεξεργαστούν σε ένα πεπερασμένο σύνολο *M* από *m* μηχανές $\{M_k\}_{k=1}^{m}$. Κάθε εργασία πρέπει να επεξεργαστεί σε κάθε μηχανή και συνεπώς προκύπτει ένα σύνολο από m_i διεργασίες (λειτουργίες) $O_{i1}, O_{i2,} \ldots, O_{im_i}$ το οποίο απεικονίζεται με τη μορφή πίνακα ως:

$$\begin{bmatrix} O_{11} & O_{12} & \cdots & O_{1m_i} \\ O_{21} & O_{22} & \cdots & O_{2m_i} \\ \vdots & \vdots & \ddots & \vdots \\ O_{i1} & O_{i2} & \cdots & O_{im_i} \end{bmatrix}$$

Όλες αυτές οι διεργασίες πρέπει να υλοποιηθούν σε προκαθορισμένη σειρά (περιορισμός προτεραιότητας). Το σύνολο των εργασιών που πρέπει να ολοκληρωθούν αναλύεται σε μια σειρά διεργασιών προς επεξεργασία στις διάφορες μηχανές. Κάθε διεργασία είναι ένα ζεύγος της μορφής (m, d), με $m \in M$ και $d \in \aleph$ που δείχνει ότι κατά την εκτέλεση της δεδομένης λειτουργίας απασχολείται η *m* μηχανή και η διάρκεια εκτέλεσης ή ολοκλήρωσης της διεργασίας είναι *d*. Συνεπώς προκύπτει ένα σύνολο *N* βημάτων που είναι $N = \sum_{i=1}^{n} m_i$. Το O_{ik} είναι η διεργασία της εργασίας J_i η επεξεργασία της οποίας πρέπει να πραγματοποιηθεί στη μηχανή M_k για ένα συνεχές χρονικό διάστημα τ_{ik} ενώ τα βήματα δεν μπορούν να προεκτοπιστούν. Αυτό σημαίνει ότι από τη στιγμή που θα ξεκινήσει η επεξεργασία της εργασίας στη συγκεκριμένη μηχανή δεν μπορεί να σταματήσει αν δεν έχει προηγουμένως ολοκληρωθεί. Το γεγονός αυτό αποτελεί μια από τις βασικές υποθέσεις του προβλήματος job shop. Στη συνέχεια της διατριβής θα παρουσιαστεί μεταξύ άλλων και πως μπορεί να προσεγγιστεί ο χρονοπρογραμματισμός με χρήση αυτομάτων όταν δεν ισχύει η υπόθεση αυτή, δηλαδή το προεκτοπιστικό μοντέλο όπου η επεξεργασία των εργασιών στις μηχανές μπορεί να διακόπτεται προσωρινά. Επίσης κάθε εργασία μπορεί να επεξεργαστεί σε μια μόνο μηχανή και αντίστροφα μια μηχανή μπορεί να απασχοληθεί από μια εργασία σε

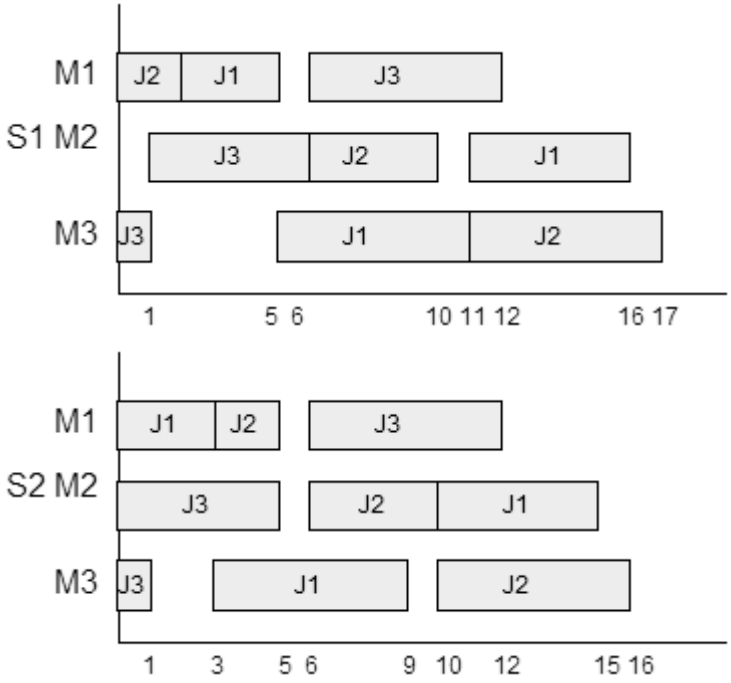
δεδομένο χρόνο (περιορισμός χωρητικότητας). Στον χρονοπρογραμματισμό, ο συνολικός χρόνος ολοκλήρωσης όλων των εργασιών ονομάζεται makespan ($C_{\max}$). Στόχος του κλασικού προβλήματος job shop είναι να είναι να προσδιοριστεί ο χρόνος έναρξης κάθε διεργασίας $t_{ik} \geq 0$ έτσι ώστε να ελαχιστοποιηθεί το $C_{\max}$ ικανοποιώντας όμως τους προηγούμενους δύο περιορισμούς (προτεραιότητας και χωρητικότητας). Δηλαδή:

$$C_{\max} = \min(C_{\max}) = (\max(t_{ik} + \tau_{ik}) : \forall J_i \in J, M_k \in M)$$

Παράδειγμα: Έστω ότι έχουμε ένα σύνολο τριών μηχανών $M = \{m_1, m_2, m_3\}$ και ένα σύνολο τριών εργασιών $J = \{J_1, J_2, J_3\}$ που πρέπει να επεξεργαστούν στις τρεις μηχανές. Η J_1 περιλαμβάνει τρία βήματα και συγκεκριμένα το πρώτο προβλέπει την επεξεργασία της J_1 στη μηχανή m_1 με χρονική διάρκεια 3, το δεύτερο την επεξεργασία στη m_2 με διάρκεια 5 και το τρίτο στη μηχανή m_3 με διάρκεια 6. Αντίστοιχα για τις εργασίες J_2 και J_3 προβλέπονται τρία βήματα για την κάθε μια.

$$J_1 = (m_1,3),(m_3,6),(m_2,5)$$
$$J_2 = (m_1,2),(m_1,4),(m_3,6)$$
$$J_3 = (m_3,1),(m_2,5),(m_1,6)$$

Με βάση τα παραπάνω δεδομένα μπορούμε να εξάγουμε δύο χρονοπρογράμματα S_1 και S_2 και τα οποία απεικονίζονται μέσω διαγραμμάτων Gantt στο ακόλουθο Σχήμα. Τα διαγράμματα δείχνουν την εξέλιξη των τριών εργασιών στις τρεις μηχανές. Στο S_1 η συνολική διάρκεια διεκπεραίωσης των εργασιών στις μηχανές είναι 17 ενώ στο S_2 είναι κατά μία μονάδα συντομότερη.



Σχήμα 2: Απεικόνιση των χρονοπρογραμμάτων S1 και S2 μέσω διαγραμμάτων Gantt

Η διάσταση του Π_j προβλήματος ορίζεται ως $n \times m$. Από το γινόμενο αυτό θεωρούμε συνήθως ότι προκύπτει και το πλήθος των βημάτων *N* με την προϋπόθεση όμως ότι $m_i = m$ για κάθε εργασία $J_i \in J$ και ότι κάθε εργασία χρησιμοποιεί ακριβώς μια φορά κάθε μηχανή. Βέβαια θα μπορούσε μια εργασία να επεξεργαστεί σε λιγότερες μηχανές ή και το ανάποδο δηλαδή να επεξεργαστεί περισσότερες από μια φορές κάποιες άλλες. Το γεγονός αυτό θα οδηγούσε σε ένα m_i που θα μπορούσε να είναι μικρότερο ή και μεγαλύτερο από το m.

Η ελαχιστοποίηση του makespan ως κριτήριο βελτιστοποίησης ενδεχομένως να μην αποτελεί την καλύτερη επιλογή για την αντιμετώπιση του προβλήματος, αλλά είναι κάτι που έχει καθιερωθεί και έχει ιστορικές προεκτάσεις αφού είναι το πρώτο είδος κριτηρίου που υιοθετήθηκε από τους ερευνητές στης αρχές της δεκαετίας του ’50. Επίσης είναι εκείνο το κριτήριο που παρουσιάζει με τον πλέον προφανή τρόπο τη θεμελιώδη υπολογιστική δυσκολία που ενυπάρχει στην προσπάθεια για τον προσδιορισμό του βέλτιστου χρονοπρογράμματος.

Η περίπτωση του Π_j με την εύρεση του ελάχιστου makespan έχει αρκετές ομοιότητες με την περίπτωση του ***Προβλήματος του Περιοδεύοντος Πωλητή (Traveling Salesman Problem)***. Η προσπάθεια για επίλυση του Π_j και η εστίαση που

έχει δοθεί από βιβλιογραφικής πλευράς στο πρόβλημα δίνει τη δυνατότητα να προσεγγιστούν άλλα περισσότερο δύσκολα και πολύπλοκα προβλήματα. Η ανάπτυξη και εφαρμογή διαφόρων αλγορίθμων και τεχνικών για την επίλυσή του έχει συμβάλλει στη δημιουργία νέων αλγοριθμικών ιδεών για την προσέγγιση άλλων περισσότερο πρακτικών προβλημάτων σχετικών όμως με το Π_j.

Όπως είναι προφανές η πολυπλοκότητα ενός τέτοιου προβλήματος αυξάνεται καθώς αυξάνεται ο αριθμός των εργασιών και των μηχανών. Έτσι ένα Π_j μεγέθους $n \times m$ έχει ένα άνω εύρος (upper bound) πιθανών λύσεων $(n!)^m$. Αυτό σημαίνει πχ πως για ένα πρόβλημα 20 εργασιών και 10 μηχανών θα υπάρχουν $7{,}2651 \times 10^{183}$ πιθανές λύσεις κάτι που καθιστά πολύ δύσκολη έως πρακτικά αδύνατη την απαρίθμηση όλων αυτών των πιθανών λύσεων για την εύρεση της βέλτιστης. Το γεγονός αυτό έχει εντάξει το Π_j στην κατηγορία των ***NP-complete προβλημάτων*** , δηλαδή αυτών που δεν μπορούν να επιλυθούν μέχρι στιγμής από οποιονδήποτε πολυωνυμικό αλγόριθμο. Πρακτικά σημαίνει ότι αν *x* είναι η εισροή και *y* μια σταθερά τότε ο χρόνος που απαιτείται για την λύση του είναι $O(x^y)$ και το πρόβλημα δηλώνεται με τον όρο ***P*** . Αντίθετα αν η πολυπλοκότητα του προβλήματος είναι της τάξης $O(y^x)$, τότε θεωρείται ως ***NP–hard*** και ένας αλγόριθμος για την βελτιστοποίησή του θα απαιτούσε ένα πλήθος υπολογιστικών βημάτων που θα αύξανε εκθετικά με την εισροή (είσοδο). Γενικά για τα *NP-complete* προβλήματα ισχύουν οι εξής δύο ιδιότητες:

- Δεν έχει βρεθεί μέχρι στιγμής κανένας πολυωνυμικός αλγόριθμος που να είναι σε θέση να τα επιλύσει.
- Αν κάποια στιγμή βρεθεί πολυωνυμικός αλγόριθμος που να μπορεί να επιλύσει κάποιο *NP-hard* πρόβλημα, τότε θα υπάρχουν πολυωνυμικοί αλγόριθμοι για όλα τα *NP-hard* προβλήματα.

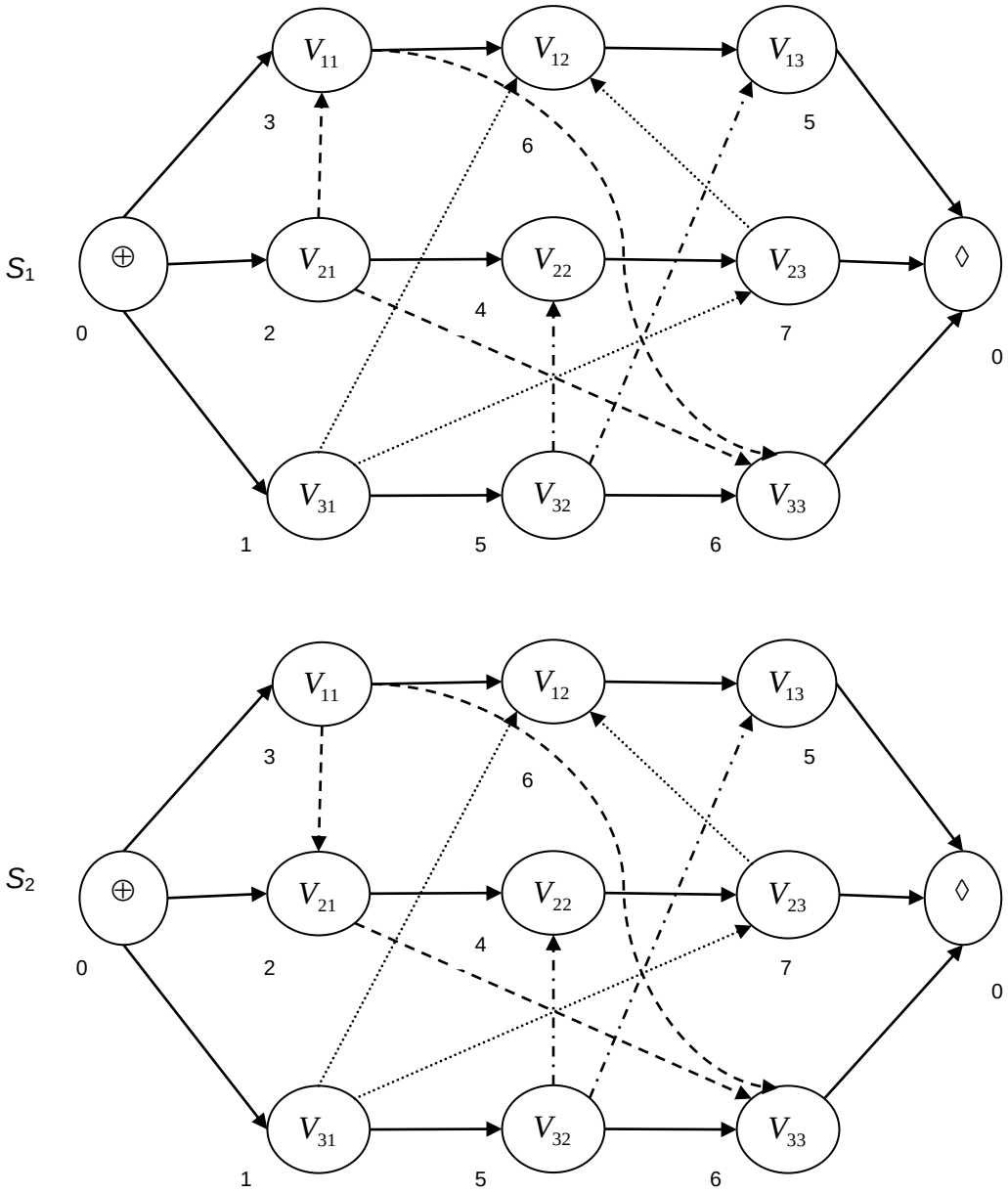
Αν και πολλοί έχουν εκφράσει την άποψη ότι δεν υπάρχουν τέτοιοι πολυωνυμικοί αλγόριθμοι εντούτοις κάτι τέτοιο δεν έχει αποδειχθεί μέχρι στιγμής μαθηματικά. Ο βασικός λόγος για τη δυσκολία επίλυσής τέτοιου τύπου προβλημάτων είναι ότι διαθέτουν μια «ενδογενή» πολυπλοκότητα επίλυσης από υπολογιστικής πλευράς. Αν υπήρχε αλγόριθμος που να επιλύει ένα *NP-complete* πρόβλημα τότε θα απαιτούσε εκθετικό χρόνο και δεν θα ήταν εύκολα εφαρμόσιμος [166].

Στη συνέχεια θα παρουσιάσουμε μερικές από τις κυριότερες τεχνικές και μεθόδους που έχουν εφαρμοστεί για την επίλυση του Π_j. Τέτοιες μέθοδοι έχουν να κάνουν με μαθηματικές προσεγγίσεις, τεχνικές διακλάδωσης και οριοθέτησης (branch

and bound techniques), bottleneck based heuristics, τεχνητή νοημοσύνη (artificial intelligence), τοπική αναζήτηση (local search methods) κλπ.

Πριν προχωρήσουμε αναλυτικότερα σε κάθε μέθοδο θα αναφερθούμε στο γραφικό ***διαζευκτικό μοντέλο (disjunctive graph model)*** των Roy και Sussmann (1964) αφού χρησιμοποιείται από τις περισσότερες μεθόδους [137].

Ένας διαζευκτικός γράφος που σχετίζεται με ένα Job –shop πρόβλημα είναι ένας γράφος $G=(V,W,A,E)$ όπου *V* είναι ένα σύνολο από κόμβους που αντιστοιχούν στα βήματα των εργασιών που πρέπει να επεξεργαστούν σε ένα σύνολο μηχανών *M*. Στο σύνολο *V* έχουν προστεθεί δύο «ειδικοί» κόμβοι $\oplus$ και $\lozenge$ που αντιστοιχούν στο αρχικό και στο τελικό βήμα και ονομάζονται ως «πηγή» (source) και «καταβόθρα» (sink). Έτσι το σύνολο *V* είναι $V=\{\oplus,1,2,3,\ldots,\lozenge\}$. Το θετικό βάρος που σχετίζεται με κάθε κόμβο είναι ίσο με τη διάρκεια εκτέλεσης κάθε βήματος όπου $\tau_{\oplus}=\tau_{\lozenge}=0$. Κάθε βήμα (εκτός των βημάτων $\oplus$ και $\lozenge$) έχει δύο άμεσα προηγούμενα και δύο άμεσα επόμενα (διάδοχα) βήματα. Δηλαδή έχει την προηγούμενη εργασία και μηχανή και την ακόλουθη εργασία και μηχανή. Ο περιορισμός προτεραιότητας που αναφέρθηκε πιο πάνω μεταξύ των βημάτων αναπαρίσταται από ένα σύνολο ***κατευθυνόμενων ακμών (directed edges)*** *A* τέτοιο ώστε $(v,v')\in A$ και δείχνει ότι το βήμα *v* είναι το αμέσως προηγούμενο του βήματος v'. Η διαμάχη μεταξύ των πόρων (μηχανών) αναπαριστάται από ένα σύνολο ***μη κατευθυνόμενων ακμών (un-directed edges)*** τέτοιων ώστε οποιαδήποτε στιγμή τα βήματα *v* και v' χρησιμοποιούν την ίδια μηχανή θα ισχύει και για τα δύο $(v,v')\in E$ και $(v',v)\in E$. Το *E* είναι μια σχέση ισότητας και θεωρούμε ως E_m το υποσύνολο του *E* που αντιστοιχεί στη μηχανή *m*. Ένας ***προσανατολισμός (orientation)*** του *E* είναι ένα υποσύνολο $E'\subseteq E$ τέτοιο ώστε για κάθε μηχανή, E'_m είναι μια γραμμική σειρά. Κάθε προσανατολισμός μπορεί να θεωρηθεί ότι καθορίζεται από μια σχέση ***προτεραιότητας (priority relation)*** μεταξύ των βημάτων που απαιτούν την ίδια μηχανή. Στο παρακάτω Σχήμα οι γεμάτες γραμμές αναπαριστούν τις ακμές του συνόλου *V* ενώ οι διακεκομμένες τις ακμές του συνόλου *E*. Φαίνεται ότι η εργασία J_2 έχει προτεραιότητα της εργασίας J_1 για τη μηχανή m_1 στο χρονοπρόγραμμα S_1 ενώ το αντίθετο συμβαίνει για το χρονοπρόγραμμα S_2 (η J_1 έχει προτεραιότητα της J_2 για τη μηχανή m_1).



Σχήμα 3: Οι δύο προσανατολισμοί που αντιστοιχούν στα χρονοπρογράμματα S1 και S2 όπου Vij είναι ο κόμβος που αντιστοιχεί στο βήμα j της εργασίας i.

Για κάθε δεδομένο προσανατολισμό, το μήκος του μακρύτερου μονοπατιού από την αρχή (⊕) μέχρι το τέλος (◊) είναι ίσο με το μήκος του μικρότερου χρονοπρογράμματος μεταξύ όλων αυτών που ικανοποιούν τις σχέσεις προτεραιότητας. Έτσι το πρόβλημα του να βρούμε το βέλτιστο χρονοπρόγραμμα περιορίζεται στο να βρούμε τη σχέση προτεραιότητας της οποίας ο προσανατολισμός οδηγεί στο ελάχιστο μικρότερο μονοπάτι. Το πρόβλημα αυτό μπορεί να επιλυθεί από τη στιγμή που υπάρχει πεπερασμένος αριθμός σχέσεων προτεραιότητας, αλλά ο αριθμός τους είναι εκθετικός

γεγονός που κάνει το πρόβλημα δύσκολο. Γενικά κάθε χρονοπρόγραμμα είναι μια εφικτή λύση που περιγράφεται από τις ακόλουθες σχέσεις:

min $t_{\lozenge}$	$\lozenge \in \aleph$
δεδομένου ότι:	
$t_v - t_{v'} \geq \tau_{v'}$	(περιορισμός σύζευξης – conjunctive constraint) $\forall v, v' \in \aleph,\ (v, v') \in A$
$t_{v'} \geq 0$	(περιορισμός νωρίτερου χρόνου έναρξης) $\forall v \in \aleph$
$t_v - t_{v'} \geq \tau_{v'} \vee t_{v'} - t_v \geq \tau_v$	(διαζευκτικός περιορισμός) $v, v' \in \aleph, (v, v') \in E_k, k \in M$

3.2. Μέθοδοι βελτιστοποίησης

3.2.1. Αποτελεσματικές μέθοδοι

Με τις μεθόδους που ακολουθούνται στους αποτελεσματικούς αλγορίθμους προκύπτει μια βέλτιστη λύση από τα δεδομένα του προβλήματος ακολουθώντας ένα σύνολο απλών κανόνων οι οποίοι προσδιορίζουν τη σειρά επεξεργασίας των εργασιών στις μηχανές. Οι απαιτήσεις σε χρόνο που υπάρχουν για την εύρεση της βέλτιστης λύσης αυξάνονται πολυωνυμικά με βάση το μέγεθος της εισόδου του προβλήματος. Η πρώτη προσπάθεια που έγινε για την επίλυση του προβλήματος και πιθανότατα αποτελεί και την πρώτη εργασία πάνω στο συγκεκριμένο θέμα πραγματοποιήθηκε από τον Johnson το 1954 , ο οποίος ανέπτυξε έναν αλγόριθμο για ένα σύστημα flow –shop δύο μηχανών με σκοπό την ελαχιστοποίηση του μέγιστου χρόνου ροής [95]. Το πρόβλημα αυτό περιγράφηκε καλύτερα ως $n/2/F/F_{\max}$ [43] και σήμαινε ότι *n* εργασίες πρέπει να επεξεργαστούν σε 2 μηχανές, σε ένα flow –shop σύστημα και το ζητούμενο είναι ο μέγιστος χρόνος ροής $F_{\max}$. Η συνεισφορά της πρώτης αυτής εργασίας του Johnson ήταν σημαντική διότι τέθηκε για πρώτη φορά ως κριτήριο για τη βελτιστοποίηση το ελάχιστο makespan. Η εφαρμογή του αλγορίθμου αυτού μπορεί να επεκταθεί επιλύοντας προβλήματα του τύπου $n/2/G/F_{\max}$ και $n/3/F/F_{\max}$. Άλλες αποτελεσματικές μέθοδοι που αναπτύχθηκαν περιλαμβάνουν αυτές των Akers (1956) [10] και Jackson (1956) [91] για τις περιπτώσεις $2 \times m$ και $n \times 2$ αντίστοιχα όπου όμως δεν υπάρχουν περισσότερα των δύο βημάτων για κάθε μηχανή. Αρκετά χρόνια αργότερα, το 1982, οι Hefetz και Adiri [82] ανέπτυξαν μια αποτελεσματική προσέγγιση για το πρόβλημα $n \times 2$ όπου όλα τα βήματα είναι μοναδιαίου χρόνου επεξεργασίες ενώ ο Williamson [162] απέδειξε ότι αν ο συνολικός χρόνος που απαιτείται από όλα τα

βήματα σε κάθε μηχανή δεν είναι μεγαλύτερος του τρία τότε μπορεί να υπάρξει ένα χρονοπρόγραμμα με makespan τρία σε πολυωνυμικό χρόνο.

Άλλες μέθοδοι που έχουν εφαρμοστεί για την αντιμετώπιση του προβλήματος είναι οι ***αλγόριθμοι ρ–προσέγγισης (ρ–approximation algorithms)*** οι οποίες προσπαθούν να εξασφαλίσουν μια λύση που να είναι ένα ποσοστό της βέλτιστης. Κάποιες από αυτές τις μεθόδους θα δούμε στη συνέχεια λίγο πιο αναλυτικά. Το 1994 ο Shmoys *et al*. πρότεινε διάφορες πολύ-λογαριθμικές προσεγγίσεις του βέλτιστου χρονοπρογράμματος που αξιολογήθηκαν από τη μεριά του σχετικού λάθους της χειρότερης περίπτωσης [140]. Επίσης ο Fizzano *et al*. (1997) παρουσίασε μια σειρά από προσεγγιστικούς αλγορίθμους οι οποίοι μορφοποιούσαν τις μηχανές σε μορφή δακτυλίου [62]. Παρόλα αυτά όμως μια από τις πιο πρόσφατες μελέτες που έχουν πραγματοποιηθεί στον τομέα αυτό από τον Williamson *et al*. αποκαλύπτει ότι τα προβλήματα χρονοπρογραμματισμού είναι δύσκολο να επιλυθούν ακόμα και προσεγγιστικά [162]. Συγκεκριμένα απέδειξε πως για κάθε $\rho \le 5/4$ δεν υπάρχει ρ – προσεγγιστικός αλγόριθμος πολυωνυμικού χρόνου για το Π_j αν δεν ισχύει *P=NP*.

Συνοψίζοντας, αν και έχει σημειωθεί σημαντική πρόοδος, ιδίως τα τελευταία χρόνια, δεν έχουν βρεθεί αποτελεσματικές μέθοδοι για εκείνες τις περιπτώσεις του Π_j όπου $m \ge 3$και $n \ge 3$. Με ακόμα πιο απαισιόδοξο τρόπο το 1982 ο French προβλέπει ότι δεν θα αναπτυχθούν ποτέ αποτελεσματικοί αλγόριθμοι για την πλειοψηφία των προβλημάτων χρονοπρογραμματισμού [66]. Το γεγονός αυτό έχει ως αποτέλεσμα μεγάλο μέρος της έρευνας να έχει μετατοπιστεί σε ***απαριθμητικές μεθόδους (enumerative methods)***. Η βασική φιλοσοφία των απαριθμητικών μεθόδων είναι η δημιουργία χρονοπρογραμμάτων «ένα προς ένα» χρησιμοποιώντας έξυπνες μεθόδους περιορισμού. Σκοπός τους είναι να επαληθεύσουν αν ένα μη βέλτιστο χρονοπρόγραμμα σημαίνει και μη βελτιστοποίηση για πολλά άλλα που δεν έχουν ακόμα δημιουργηθεί και διερευνηθεί έτσι ώστε με τον τρόπο αυτό να αποτραπεί η ανάγκη για αναζήτηση σε ολόκληρο το χώρο των εφικτών λύσεων.

3.2.2. Μαθηματικές διατυπώσεις

Έχει διαπιστωθεί ότι το πρόβλημα του χρονοπρογραμματισμού είναι δυνατόν αν επιλυθεί με βέλτιστο τρόπο χρησιμοποιώντας ***τεχνικές μαθηματικού προγραμματισμού (mathematical programming techniques)***. Μια από τις πιο

γνωστές και χαρακτηριστικές μορφές μαθηματικής διατύπωσης για το Π_j είναι αυτή του ***μικτού γραμμικού προγραμματισμού ακεραίου (Mixed Integer Linear Programming - MILP)*** του Manne (1960) που συνοψίζεται παρακάτω [108]. Στη περίπτωση του MILP έχουμε ένα σύνολο γραμμικών περιορισμών, μια γραμμική αντικειμενική συνάρτηση, με τον επιπρόσθετο περιορισμό όμως ότι κάποιες από τις μεταβλητές απόφασης (y_{ipk}) είναι ακέραιοι. Εδώ οι ακέραιες μεταβλητές είναι δυαδικές (binary) και χρησιμοποιούνται για να εκφράσουν τους διαζευκτικούς περιορισμούς (disjunctive constraints). Το *K* είναι ένας μεγάλος αριθμός που σύμφωνα με τον Van Hulle πρέπει να είναι μεγαλύτερος από το άθροισμα όλων των μικρότερων χρόνων επεξεργασίας [157].

Ελαχιστοποίηση του $C_{\max}$ δεδομένου ότι:		
Χρόνοι έναρξης	$t_{ik} \geq 0$	$\{i, p\} \in J \quad \{k, h\} \in M$
Περιορισμός προτεραιότητας	$t_{ik} - t_{ih} \geq \tau_{ih}$	Αν το βήμα O_{ih} προηγείται του O_{ik}
Διαζευκτικός περιορισμός	$t_{pk} - t_{ik} + K(1 - y_{ipk}) \geq \tau_{ik}$	$y_{ipk} = 1$, αν το O_{ik} προηγείται του O_{pk}
	$t_{ik} - t_{pk} + K(y_{ipk}) \geq \tau_{pk}$	$y_{ipk} = 0$, διαφορετικά
		Όπου $K > \left(\sum_{i=1}^{n} \sum_{k=1}^{m} \tau_{ik} - \min(\tau_{ik}) \right)$

Το χαρακτηριστικό αυτής της μεθόδου είναι ότι οι μεταβλητές ακεραίων κλιμακώνονται εκθετικά [29], ενώ αν και κατά καιρούς έχουν χρησιμοποιηθεί καλύτερες διατυπώσεις εντούτοις έχουν έναν μεγάλο αριθμό περιορισμών [108]. Μια σειρά από εργασίες που έχουν δημοσιευθεί τονίζουν τα μειονεκτήματα αυτών των μεθόδων και διαπιστώνουν ότι αυτού του είδους οι τεχνικές είναι εφαρμόσιμες μόνο για πολύ απλοποιημένες περιπτώσεις του Π_j, έχοντας όμως ταυτόχρονα μεγάλες απαιτήσεις σε χρόνο. Συγκεκριμένα, οι Giffler και Thompson (1960) αναφέρουν ότι ο προγραμματισμός ακεραίων δεν έχει οδηγήσει σε πρακτικές μεθόδους επίλυσης [70], ενώ ο French (1982) υποστηρίζει ότι η διατύπωση προγραμματισμού ακεραίων για το Π_j είναι υπολογιστικά ανέφικτη [66]. Τέλος οι Nemhauser και Wolsey (1988) [122] και Blazewicz *et al*. (1991) [27] έχουν εστιάσει στις δυσκολίες εφαρμογής και έχουν δείξει

ότι τα μοντέλα γραμμικού προγραμματισμού δεν έχουν ακόμα πετύχει να αντιμετωπίσουν όπως θα έπρεπε προβλήματα χρονοπρογραμματισμού.

Όποια πρόοδος έχει επιτευχθεί με τη χρήση μαθηματικών διατυπώσεων ανήκει σε μια ειδική κατηγορία αυτών που είναι οι ***προσεγγίσεις Lagrangian Relaxation (LR)*** [60], [61], [156], [49], [84] καθώς και στις ***μεθόδους αποσύνθεσης (decomposition methods)*** [17], [14], [42], [100]. Στις μεθόδους LR οι περιορισμοί προτεραιότητας και χωρητικότητας αποδίδονται χρησιμοποιώντας μη αρνητικούς πολλαπλασιαστές Lagrange (non negative Lagrangian Multipliers). Αντίστοιχα στις μεθόδους αποσύνθεσης το συνολικό πρόβλημα «σπάει» σε μια σειρά από υποπροβλήματα και τα οποία λόγω του ότι είναι πιο εύκολα στη διαχείρισή τους επιλύονται με βέλτιστο τρόπο.

Γενικά οι μαθηματικές μέθοδοι δεν αντιμετωπίζουν επαρκώς το πρόβλημα του χρονοπρογραμματισμού Job –shop συστημάτων και συνεπώς το βάρος έχει πέσει στης τεχνικές διακλάδωσης και οριοθέτησης που θα δούμε στη συνέχεια.

3.2.3. Τεχνικές διακλάδωσης και οριοθέτησης

Οι αλγόριθμοι διακλάδωσης και οριοθέτησης βασίζονται στην κατασκευή δέντρων μέσω των οποίων μπορεί να γίνει η αναπαράσταση του χώρου των λύσεων του προβλήματος. Η αναζήτηση ξεκινάει από τον υψηλότερο κόμβο (ρίζα) του δέντρου και ολοκληρώνεται όταν πλέον έχει διερευνηθεί ο κόμβος του χαμηλότερου επιπέδου (φύλλο). Κάθε κόμβος που βρίσκεται σε ένα δεδομένο επίπεδο ρ του δέντρου αναπαριστά μια επιμέρους (ακολουθία ρ βημάτων. Όπως προδίδει και το όνομα των μεθόδων αυτών για την επιτυχή αναζήτηση χρησιμοποιούνται τεχνικές τόσο διακλάδωσης όσο και οριοθέτησης. Συγκεκριμένα από έναν μη επιλεγμένο κόμβο η τεχνική της διακλάδωσης έχει σκοπό να προσδιορίσει το επόμενο σύνολο των πιθανών κόμβων στους οποίους θα συνεχιστεί η αναζήτηση. Οι δύο πιο γνωστές στρατηγικές διακλάδωσης που έχουν προταθεί είναι η Generating Active Schedules (GAS) και η Settling Essential Conflicts (SEC) [101], [21]. Στη GAS [70] κάθε κόμβος περιλαμβάνει ένα επιμέρους χρονοπρόγραμμα και οι μηχανισμοί διακλάδωσης καθορίζουν το σύνολο των βημάτων που ακολουθηθούν στη συνέχεια. Στη SEC η τεχνική της διακλάδωσης προσδιορίζει το πότε η O_i πρέπει να τοποθετηθεί πριν τη O_j και αντίστροφα. Πάντως η SEC θεωρείται πιο ευέλικτη ως στρατηγική και ανώτερη της GAS [21]. Από την άλλη πλευρά η μέθοδος της οριοθέτησης βοηθά στο να ληφθεί η απόφαση σχετικά με ποιο βήμα θα συνεχιστεί η διαδικασία της αναζήτησης. Βασίζεται στο εκτιμώμενο κάτω εύρος

(LB) και στο καλύτερο άνω εύρος που έχει μέχρι τώρα επιτευχθεί. Αν σε κάποιο κόμβο το εκτιμώμενο κάτω εύρος βρεθεί ότι είναι μεγαλύτερο από το τρέχων καλύτερο άνω εύρος, τότε δεν χρειάζεται να συνεχιστεί περαιτέρω η αναζήτηση σε αυτή την επιμέρους επιλογή αφού δεν θα βελτιωθεί το υπάρχον UB. Συνεπώς διαγράφονται τόσο η επιμέρους αυτή επιλογή όσο και οι ακόλουθες αυτής. Σε αυτή την περίπτωση η αναζήτηση επιστρέφει πίσω στον υψηλότερο μη εμβαθυνόμενο κόμβο του δέντρου. Η αναζήτηση τερματίζει όταν όλοι οι κόμβοι έχουν είτε με τον ένα είτε με τον άλλο τρόπο εξερευνηθεί. Γενικά στη βιβλιογραφία έχουν προταθεί διάφοροι τρόποι οριοθέτησης που αποτρέπουν την αναζήτηση σε διαφόρους τομείς του δέντρου μειώνοντας το χώρο των λύσεων [10], [32], [33]. Ο κυριότερος τρόπος που επιτυγχάνεται αυτό είναι αποσυνθέτοντας (σπάζοντας) το πρόβλημα σε επιμέρους υποπροβλήματα.

Οι τεχνικές BB μελετήθηκαν αρχικά από τους Brooks και White το 1965 [30], Ignall και Schrange (1965) [89] και Lomnicki (1965) [107], ενώ στις πρώτες εργασίες εντάσσονται και αυτές των Brown και Lomnicki το 1966 και Greenberg το 1968 [31], [79]. Το 1969 πραγματοποιήθηκε η πρώτη χρήση μεθόδων BB στο Π_j από τον Balas, που εφάρμοσε το διαζευκτικό γραφικό μοντέλο εξετάζοντας όμως μόνο τα κρίσιμα βήματα [19]. Αρκετά χρόνια αργότερα οι McMahon και Florian (1975) παρουσίασαν την πρώτη επιτυχημένη εφαρμογή στο πρόβλημα της αποσύνθεσης του προβλήματος σε πρόβλημα μιας μηχανής [116].

Το 1989 οι Carlier και Prinson [36] υπολόγισαν το κάτω εύρος χρησιμοποιώντας το Προεκτοπιστικό Χρονοπρόγραμμα του Jackson (JPS) (1955) [90] βασιζόμενοι σε μια από τις εργασίες του. Αργότερα και συγκεκριμένα το1990 οι δύο αυτοί ερευνητές βελτίωσαν την εργασία τους προτείνοντας νέους κανόνες για καλύτερες τεχνικές διακλάδωσης.

Ένα χρόνο αργότερα (1991) οι Applegate και Cook λαμβάνοντας ως αφετηρία την εργασία των Carlier και Prinson προσδιορίζουν αν ένα βήμα *i* πρέπει να πραγματοποιηθεί πριν ή μετά από ένα σύνολο βημάτων *g*, χρησιμοποιώντας στρατηγικές διακλάδωσης και οριοθέτησης ταυτόχρονα [14]. Η συγκεκριμένη στρατηγική που είναι γνωστή ως Edge-Finding βασίζεται στο συνδυασμό χαμηλών εύρων που προκύπτουν, αποσυνθέτοντας το πρόβλημα σε επιμέρους προβλήματα χρονοπρογραμματισμού μιας μηχανής.

Γενικά στην εργασία των Carlier και Prinson βασίστηκαν αρκετοί άλλοι ερευνητές με σκοπό μια πιο αποτελεσματική αναζήτηση του χώρου των λύσεων [134]. Οι προτεινόμενες μέθοδοι έχουν να κάνουν με στρατηγικές παράλληλης αναζήτησης.

Τέλος ενώ όλες οι προηγούμενες μέθοδοι στηρίζονται στο διαζευκτικό γραφικό μοντέλο ο Martin (1996) υιοθέτησε μια μεθοδολογία που βασίζεται σε μια αναπαράσταση προσανατολισμένη στο χρόνο (time oriented representation) για τη λήψη αποφάσεων στο Π_j [114].

Αν και από υπολογιστικής πλευράς έχουν γίνει μεγάλες βελτιώσεις για την αντιμετώπιση του Π_j χρησιμοποιώντας τεχνικές BB εντούτοις αυτό οφείλεται περισσότερο στη διαθέσιμη τεχνολογία παρά στις ίδιες τις μεθόδους. Δεν μπορούν να εφαρμοστούν σε μεγάλα προβλήματα ενώ και η εκτέλεσή τους απαιτεί καλή κατανόηση του Π_j. Αυτό συμβαίνει διότι απαιτούνται καλοί κανόνες και μέθοδοι επιλογής για να εμβαθύνει κανείς σε κόμβους που βρίσκονται σε υψηλό επίπεδο στο δέντρο χωρίς εκτενή αναζήτηση. Το γεγονός αυτό έχει οδηγήσει πολλούς ερευνητές να στρέψουν την προσοχή τους σε προσεγγιστικές μεθόδους.

3.3. Προσεγγιστικές μέθοδοι

Αν και οι προσεγγιστικές μέθοδοι δεν οδηγούν σε βέλτιστες λύσεις εντούτοις έχουν το πλεονέκτημα ότι πετυχαίνουν λύσεις κοντά στο βέλτιστο μέσα σε αποδεκτούς χρόνους και συνεπώς είναι κατάλληλες για μεγάλα προβλήματα. Οι πρώτοι οι οποίοι τόνισαν τη σημασία των προσεγγιστικών μεθόδων για την επίλυση των προβλημάτων ήταν οι Glover και Greenberg (1989), οι οποίοι εξέφρασαν την άποψη ότι η μεθοδολογία για την αναζήτηση λύσεων μέσω κατευθυνόμενων δέντρων είναι μη ικανοποιητική ιδίως όταν χρησιμοποιείται για προβλήματα δύσκολα υπολογιστικά [75]. Επίσης υποστήριξαν ότι ευρετικές μέθοδοι που θα βασίζονταν στα φυσικά φαινόμενα και στην επίλυση έξυπνων προβλημάτων θα ήταν περισσότερο κατάλληλες και συνεπώς θα μπορούσε να υπάρξει σύνδεση μεταξύ επιχειρησιακής έρευνας και τεχνητής νοημοσύνης. Παρακάτω θα εξεταστούν τέσσερις κύριες κατηγορίες προσεγγιστικών μεθόδων και συγκεκριμένα τους Κανόνες Διεκπεραίωσης με Προτεραιότητες (Priority Dispatch Rules), τη μέθοδο Bottleneck Based Heuristics, τη Τεχνητή Νοημοσύνη και τις Μεθόδους Τοπικής Αναζήτησης (Local Search Methods).

3.3.1. Κανόνες διεκπεραίωσης με προτεραιότητες

Μια από τις πρώτες κατηγορίες προσεγγιστικών μεθόδων που εφαρμόστηκαν στο Π_j ήταν οι Κανόνες Διεκπεραίωσης με Προτεραιότητες και οι οποίοι έγιναν ιδιαίτερα δημοφιλής λόγω της ευκολίας εφαρμογής τους και των σχετικά περιορισμένων απαιτήσεων που είχαν. [18], [66], [119]. Σε γενικές γραμμές η λειτουργία τους ήταν η εξής: σε κάθε δεδομένη στιγμή όλα τα βήματα που είναι διαθέσιμα για προγραμματισμό λαμβάνουν μια προτεραιότητα και το βήμα εκείνο με την υψηλότερη προτεραιότητα επιλέγεται για δρομολόγηση. Επακόλουθο της μεθόδου είναι η δημιουργία πολλαπλών εκτελέσεων που σκοπό έχουν την επίτευξη των επιθυμητών αποτελεσμάτων.

Οι πρώτες χρονολογικά εργασίες που πραγματοποιήθηκαν αναφορικά με τους Κανόνες Διεκπεραίωσης με Προτεραιότητες ανήκουν στους Jackson [90], [92], Smith [141] , Rowe και Jackson [136], Giffler και Thompson [70] και Gere [69]. Από όλες, αυτή που ξεχωρίζει περισσότερο είναι αυτή των Giffler και Thompson (1960) και η οποία αποτελεί τη βάση για όλους τους κανόνες διεκπεραίωσης με προτεραιότητες. Στη συγκεκριμένη μέθοδο επιλέγουμε εκείνο το βήμα O_j με το μικρότερο χρόνο ολοκλήρωσης από το σύνολο των βημάτων που δεν έχουν δρομολογηθεί ακόμα [70]. Στη συνέχει βρίσκουμε όλα τα άλλα βήματα τα οποία χρησιμοποιούν την ίδια μηχανή M_k και τα οποία ξεκινούν νωρίτερα από το χρόνο ολοκλήρωσης του O_j. Αυτά τα βήματα ορίζουν ένα σύνολο συγκρουόμενων βημάτων. Ακολούθως από αυτό το σύνολο επιλέγεται ένα βήμα και τοποθετείται να ξεκινήσει το συντομότερο δυνατόν, ενώ η ίδια διαδικασία επαναλαμβάνεται μέχρι να τοποθετηθούν σε σειρά όλα τα βήματα. Οι Κανόνες Διεκπεραίωσης με Προτεραιότητες ουσιαστικά έχουν να κάνουν με την επιλογή των βημάτων από το σύνολο των συγκρουόμενων βημάτων.

Κατά καιρούς έχουν πραγματοποιηθεί διάφορες έρευνες σχετικά με τις μεθόδους που διαχρονικά έχουν εφαρμοστεί για το *Πj* στηριζόμενες στους κανόνες διεκπεραίωσης με προτεραιότητες και οι οποίες έχουν ταξινομηθεί ανάλογα με τα αποτελέσματα που κάθε φορά πετυχαίνουν [131], [81], [26]. Η πιο πρόσφατη τέτοιου είδους συγκριτική ανάλυση που πραγματοποιήθηκε σύγκρινε 42 κανόνες διεκπεραίωσης με προτεραιότητες χρησιμοποιώντας το μοντέλο γραμμικού προγραμματισμού [38]. Τα αποτελέσματα της έρευνας έδειξαν ότι οι κανόνες που βασίζονται στον συντομότερο χρόνο επεξεργασίας (shortest processing time –SPT) αποδίδουν ενώ οι κανόνες που βασίζονται στον μεγαλύτερο χρόνο επεξεργασίας (longest processing time –LPT) δεν αποδίδουν καλά. Από τη στιγμή που ιδιαίτεροι κανόνες αποδίδουν άσχημα αναφορικά με το κριτήριο του makespan έχουν εφαρμοστεί διάφορες ευρετικές μέθοδοι για την

αντιμετώπιση αυτών των προβλημάτων. Συγκεκριμένα ο αλγόριθμος του Viviers ενσωματώνει τρία επίπεδα κλάσεων προτεραιοτήτων μέσα στις ευρετικές συντομότερου χρόνου επεξεργασίας [159]. Η πιο κοινή μέθοδος για τη βελτίωση της απόδοσης των λύσεων είναι η πραγματοποίηση πιθανοτικού συνδυασμού ανεξαρτήτων κανόνων διεκπεραίωσης προτεραιοτήτων. Παραδείγματα τέτοιων στρατηγικών έχουν εφαρμοστεί το 1963 από τους Crowston *et al*. [44], Fisher και Thompson [59]. Ο Lawrence (1984) συνέκρινε την απόδοση δέκα τέτοιων κανόνων διεκπεραίωσης με προτεραιότητες εφαρμόζοντας συνδυασμό των κανόνων με τυχαίο τρόπο και έδειξε ότι με αύτη την τακτική επιτυγχάνονται πολύ καλύτερα αποτελέσματα αλλά απαιτείται παράλληλα μεγαλύτερος υπολογιστικός χρόνος [105]. Άλλες πιο προχωρημένες μέθοδοι χρησιμοποιούν γενετικούς αλγορίθμους και fuzzy logic για τον έλεγχο και την εύρεση του κανόνα που θα εφαρμοστεί.

Είναι γεγονός ότι οι κανόνες διεκπεραίωσης με προτεραιότητες επιλέγουν μια λειτουργία και την προσθέτουν σε μια επιμέρους τρέχουσα ακολουθία λειτουργιών ενώ ταυτόχρονα τεχνικές οριοθέτησης και διακλάδωσης εξετάζουν όλες τις υπόλοιπες λειτουργίες. Η τεχνική beam search [119] δίνει τη δυνατότητα για ισορροπία μεταξύ αυτών των προσεγγίσεων εξετάζοντας τον αριθμό των καλύτερων λύσεων σε κάθε δεδομένο σημείο απόφασης.

Άλλες τεχνικές οι οποίες χρησιμοποιούν την τεχνική beam search είναι ο job insertion αλγόριθμος των Werner και Wrinkler (1995) ο οποίος αποτελείται από δύο φάσεις [160]. Στη πρώτη φάση, αυτή της εφαρμογής της παρεμβολής (insertion), κατασκευάζεται ένα χρονοπρόγραμμα και επεκτείνεται ο αλγόριθμος του Nawaz *et al.* (1983) [121]. Τοποθετείται μια λειτουργία σε ένα μερικό χρονοπρόγραμμα έτσι ώστε να ελαχιστοποιείται η διάρκεια της μακρύτερης διαδρομής που περνάει από αυτό. Στη δεύτερη φάση εφαρμόζεται η στρατηγική της εκ νέου παρεμβολής (reinsertion) που σκοπό έχει με επαναληπτικό τρόπο να βελτιώσει την αρχική λύση. Και στις δύο φάσεις χρησιμοποιείται η beam search για τη βελτίωση της αναζήτησης.

Οι κανόνες διεκπεραίωσης με προτεραιότητες είναι περισσότερο κατάλληλες για αρχικές λύσεις παρά για την αντιμετώπιση ενός ολοκληρωμένου Π_j προβλήματος. και η beam search θα πρέπει να χρησιμοποιείται σε συνδυασμό με άλλες μεθόδους.

3.3.2. Ευρετικές μέθοδοι

Για πάρα πολλά χρόνια οι κανόνες διεκπεραίωσης με προτεραιότητες κυριαρχούσαν αναφορικά με το σύνολο των προσεγγιστικών μεθόδων. Τα τελευταία όμως χρόνια με τη δυναμική ανάπτυξη των υπολογιστών, όπως επίσης και τον προσεχτικό σχεδιασμό, ανάλυση και εφαρμογή διαφόρων τεχνικών αναπτύχθηκαν διάφορες νέες μέθοδοι όπως η Shifting Bottleneck Procedure (SBP).

Η SBP χαρακτηρίζεται από μια σειρά από λειτουργίες που πρέπει να γίνουν όπως η αναγνώριση υποπροβλημάτων, η επιλογή bottleneck, η λύση του υποπροβλήματος και η εκ νέου βελτιστοποίηση του χρονοπρογράμματος. Η βασική στρατηγική που ακολουθείται στο Π_j πρόβλημα έχει να κάνει με τη διάσπαση του συνολικού προβλήματος σε *m* υποπροβλήματα μιας μηχανής και την επίλυση του κάθε υποπροβλήματος με επαναληπτικό τρόπο χρησιμοποιώντας την προσέγγιση του Carlier (1982) [35]. Κάθε λύση του προβλήματος μιας μηχανής συγκρίνεται με τις άλλες λύσεις και στη συνέχεια οι μηχανές ταξινομούνται με βάση τις λύσεις. Η μηχανή εκείνη που δεν έχει δρομολογηθεί και έχει αναφορικά με τη λύση τη μεγαλύτερη τιμή χαρακτηρίζεται ως μηχανή bottleneck. Η SBP αυτό που κάνει είναι να τοποθετεί τη bottleneck μηχανή με βάση τις μηχανές που είναι ήδη χρονοπρογραμματισμένες ενώ οι υπόλοιπες μηχανές που δεν έχουν ακόμα δρομολογηθεί αγνοούνται. Η επιλογή της bottleneck μηχανής γίνεται με βάση την υπόθεση ότι ο χρονοπρογραμματισμός της σε αργότερο στάδιο θα χειροτέρευε το makespan περισσότερο.

Κάθε φορά που μια μηχανή αναγνωρίζεται ως μηχανή bottleneck γίνεται ο προγραμματισμός της και στη συνέχεια όλες οι υπόλοιπες ήδη προγραμματισμένες μηχανές βελτιστοποιούνται εκ νέου επιλύοντας το πρόβλημα μιας μηχανής ξανά. Η βασική συνεισφορά αυτής της προσέγγισης έχει να κάνει με τη λήψη αποφάσεων σχετικά με τη σειρά που οι μηχανές προγραμματίζονται. Ο Adams *et al.* (1988) αναφέρει τη συγκεκριμένη τεχνική ως SB(I). Η τεχνική SB(I) έχει εφαρμογή και σε κόμβους σε επιμέρους τμήματα δέντρων και ονομάζεται SB(II) έτσι ώστε να μπορούν να εξεταστούν διαφορετικοί προγραμματισμοί μηχανών [9].

Οι Applegate και Cook (1991) βασιζόμενοι στη μέθοδο SB(II) έχουν κατασκευάσει μια μεθοδολογία, την “Bottle-k” (όπου το k= 4,5 και 6), όπου το k εκφράζει τις k τελευταίες μη προγραμματισμένες μηχανές [14]. Επίσης κατά καιρούς έχουν προταθεί διάφορες μέθοδοι που αποτελούν συνδυασμό μεθόδων όπως της μεθόδου “Bottle-k” , της μεθόδου Edge-finder και του αλγορίθμου “Shuffle”. Στη

περίπτωση αυτή από ένα αρχικό χρονοπρόγραμμα που είναι κατασκευασμένο με τη μέθοδο “Bottle-k” ο αλγόριθμος “Shuffle” φτιάχνει τη σειρά επεξεργασίας μιας ή ενός μικρού αριθμού επιλεγομένων με ευρετικό τρόπο μηχανών, ενώ οι υπόλοιπες μηχανές βελτιστοποιούνται μέσω της μεθόδου Edge-finder.

Οι παραπάνω μέθοδοι δέχτηκαν την κριτική των Dauzere-Peres και Lasserre [46] και Balas *et al.* (1995) [20] οι οποίοι ανέφεραν μια σειρά από μειονεκτήματα σχετικά με τις στρατηγικές που είχαν προταθεί από τον Adams *et al.* (1988), ενώ η κριτική αφορούσε και τη μεταβλητή της μεθόδου που πρότειναν οι Applegate και Cook (1991). Τα μειονεκτήματα αυτά είχαν να κάνουν μεταξύ άλλων με τη μέθοδο SB(I) και συγκεκριμένα όταν δημιουργείται μια ακολουθία σε μια μηχανή αυτό μπορεί να οδηγήσει περιορισμούς σε ζεύγη εργασιών σε μη προγραμματισμένες μηχανές. Οι περιορισμοί αυτοί ονομάζονται delay precedence constraints (DPCs) και προκύπτουν εξαιτίας του ότι ο προγραμματισμός μιας δεδομένης μηχανής μπορεί να θέτει συγκεκριμένες συνθήκες στον προγραμματισμό κάποιας άλλης μηχανής. Για παράδειγμα η εργασία *i* πρέπει να προηγείται της εργασίας *j* αφού πρώτα έχει περάσει ένας προκαθορισμένο χρονικό διάστημα. Έτσι λοιπόν όταν αυτή η εξάρτηση έχει να κάνει με ένα στιγμιότυπο του προβλήματος που είναι αυτό του χρονοπρογραμματισμού μιας μηχανής υπάρχουν λιγότεροι περιορισμοί με αποτέλεσμα να μη επιλέγεται η bottleneck μηχανή, η σωστή ακολουθία (δρομολόγηση) δεν πραγματοποιείται, η εκ νέου βελτιστοποίηση δεν εξασφαλίζει τη μείωση του makespan και συνεπώς να μην είναι εφικτή η λύση με τη μέθοδο SB(I). Το γεγονός αυτό οδήγησε σε μια σειρά από προτάσεις για ευρετικές στρατηγικές που έχουν να κάνουν με τους DPCs και τη μέθοδο SBP [46], [45], [20].

Πάντως παρά τα οποιαδήποτε μειονεκτήματα έχουν κατά καιρούς καταγραφεί για τις SBP μεθόδους, ο πολύ καλός σχεδιασμός, ανάλυση και εφαρμογή αυτών των μεθοδολογιών έχει συμβάλλει στην ενσωμάτωσή τους και σε άλλες εργασίες και θέματα με αποτέλεσμα να έχουν βελτιώσει το άνω και κάτω εύρος πολλών προβλημάτων κλάσεως NP-Hard.

3.3.3. Τεχνητή νοημοσύνη

Η τεχνητή νοημοσύνη (ΤΝ) αποτελεί ένα κομμάτι της επιστήμης των υπολογιστών που ασχολείται με την ενοποίηση της νοημοσύνης σε βιολογικό και υπολογιστικό επίπεδο. Η προέλευσή της έχει να κάνει με την επιστήμη της βιολογίας, ενώ η εφαρμογή της με τη χρήση των αρχών που ισχύουν στη φύση για την εύρεση λύσεων σε διάφορα προβλήματα. Σκοπός είναι η τεχνητή νοημοσύνη να καταστήσει

τους υπολογιστές ικανούς και χρήσιμους στην επίλυση προβλημάτων. Στη συνέχεια θα παρουσιαστούν δύο από τις μεθόδους τεχνητής νοημοσύνης, η ικανοποίηση περιορισμών (constraint satisfaction –CS) και τα νευρωνικά δίκτυα (neural networks)

3.3.3.1. Ικανοποίηση Περιορισμών

Οι τεχνικές Ικανοποίησης Περιορισμών (Ι.Π.) στοχεύουν στη μείωση του μεγέθους του χώρου των λύσεων εφαρμόζοντας περιορισμούς στη σειρά στην οποία οι μεταβλητές επιλέγονται και εν συνεχεία καθορίζεται ποιες τιμές θα δοθούν σε ποιες μεταβλητές. Αφού δοθούν οι τιμές στις μεταβλητές τότε μετακινούνται οι ασυνέπειες και η μέθοδος αυτή μετακίνησης των ασυνεπειών ονομάζεται έλεγχος συνέπειας (consistency checking). Το πρόβλημα ικανοποίησης περιορισμών επιλύεται τελικά όταν έχει πραγματοποιηθεί πλήρης κατανομή των μεταβλητών η οποία όμως δεν έρχεται σε αντίθεση με τους περιορισμούς του προβλήματος, γι αυτό άλλωστε ονομάζεται και μέθοδος ικανοποίησης περιορισμών. Παρά το γεγονός ότι η ικανοποίηση περιορισμών ανήκει στις μεθόδους τεχνητής νοημοσύνης εντούτοις σε αυτή χρησιμοποιούνται δέντρα αναζήτησης και εφαρμόζονται αλγόριθμοι διακλάδωσης και οριοθέτησης.

Οι τεχνικές ικανοποίησης περιορισμών έχουν εφαρμοστεί και για την αντιμετώπιση του Π_j προβλήματος. Μια από τις πρώτες εργασίες που πραγματοποιήθηκαν στον τομέα αυτό ήταν αυτή του Erschler *et al.* (1976) όπου προσδιορίζεται πότε $\left[(i \prec j),(j \prec i)\right]$ δεδομένου ότι $M_i = M_j$ [55]. Σε γενικές γραμμές η έρευνα αναφορικά με τη χρήση μεθόδων ικανοποίησης περιορισμών έχει απασχολήσει σχετικά μικρό αριθμό ερευνητών.

3.3.3.2. Νευρωνικά Δίκτυα

Η μέθοδος των Νευρωνικών Δικτύων βασίζεται στο πλαίσιο πάνω στο οποίο είναι οργανωμένος και λειτουργεί ο ανθρώπινος εγκέφαλος. Με τις τεχνικές των Νευρωνικών Δικτύων η επεξεργασία της πληροφορίας γίνεται μέσα σε ένα διασυνδεδεμένο δίκτυο που αποτελείται από παράλληλες μονάδες επεξεργασίας. Αυτού του είδους οι μέθοδοι έχουν τη δυνατότητα υπολογισμών και γενίκευσης και το γεγονός αυτό έχει οδηγήσει στο να γίνουν ιδιαίτερα δημοφιλής και να χρησιμοποιηθούν σε πολλές εφαρμογές στη πραγματική ζωή. Η μέθοδος των Νευρωνικών Δικτύων έχει εφαρμοστεί και για την αντιμετώπιση του Π_j προβλήματος. Μια από τις κυριότερες αρχιτεκτονικές που έχουν εφαρμοστεί στο Π_j είναι η αναζήτηση δικτύου (searching net) και κυρίως τα δίκτυα Hopfield. Το δίκτυα Hopfield είναι αυτοσυσχετιζόμενα μη γραμμικά

δίκτυα (autoassociative non –linear networks) τα οποία έχουν το χαρακτηριστικό ότι διαθέτουν έμφυτες δυναμικές ώστε να ελαχιστοποιούν τη συνάρτηση ενέργειας του συστήματος ή τη συνάρτηση ευστάθειας Lyapunov. Σε αρκετές από τις μεθόδους των δικτύων Hopfield εφαρμόζεται ένα μαθηματικό μοντέλο που έχει σκοπό τη χαρτογράφηση του Π_j ως ένα νευρωνικό δίκτυο. Μια άλλη τεχνική που εφαρμόζεται για την επίλυση του Π_j είναι τα δίκτυα διόρθωσης λάθους (error correcting networks).

Όπως ειπώθηκε και προηγουμένως το δίκτυα Hopfield είναι μια τεχνική των νευρωνικών δικτύων η οποία εφαρμοζόμενη στο Π_j σκοπό έχει να ελαχιστοποιήσει τη συνάρτηση ενέργειας E η οποία βασίζεται στο makespan, δεδομένων μιας σειράς από περιορισμούς προτεραιότητας και πόρων. Κάθε φορά που ένας περιορισμός παραβιάζεται τότε δημιουργείται μια ποινή η οποία έχει ως αποτέλεσμα την αύξηση της συνάρτησης ενέργειας E. Αρκετά πρόσφατα οι Foo και Takefuji (1988a, b) εφάρμοσαν μια τεχνική κωδικοποίησης με σκοπό τη χαρτογράφηση του Π_j σε ένα πίνακα διαστάσεων mn αποτελούμενο από (mn+1) νευρώνες [63], [64]. Λίγο αργότερα οι Foo και Takefuji (1988c) προσπαθώντας να βελτιώσουν τη προηγούμενη εργασία τους για τη μορφοποίηση του Π_j ως ένα πρόβλημα μικτού γραμμικού προγραμματισμού ακεραίων (MIP), κατασκεύασαν ένα Νευρωνικό Δίκτυο Γραμμικού Προγραμματισμού Ακεραίων (Integer Linear Programming Neural Network -ILPNN) [65]. Με τη μέθοδο αυτή η συνάρτηση ενέργειας εκφράζεται από το άθροισμα των χρόνων έναρξής όλων των εργασιών, ενώ η λύση προκύπτει με την απόδοση προσαρμογών γραμμικού προγραμματισμού και ακεραίων (linear program and integer adjustments) μέχρις ότου να υπάρξει σύγκλιση.

Το 1991 ο Van Hulle έδειξε ότι η προσέγγιση που είχαν εφαρμόσει οι Foo και Takefuji δεν εξασφάλιζε εφικτές λύσεις [157]. Το 1994 οι Willem και Rooda [161] χρησιμοποίησαν μια MIP μορφοποίηση για το Π_j όπως οι Foo και Takefuji, αλλά προσπάθησαν να αντιμετωπίσουν μερικές από τις δυσκολίες που υπήρχαν μειώνοντας το χώρο των λύσεων στο δικό τους ILPNN πραγματοποιώντας εκ των προτέρων υπολογισμούς.

Η προσπάθεια να ξεπεραστούν οι περιορισμοί της κατασκευής ILPNN συνεχίστηκαν με τους Zhou *et al.* (1991) οι οποίοι πρότειναν τη κατασκευή ενός Νευρωνικού Δικτύου Γραμμικού Προγραμματισμού (Linear Programming Neural Network –LPNN) [165]. Στο μοντέλο αυτό δεν χρησιμοποίησαν τετραγωνική συνάρτηση ενέργειας, αλλά αντίθετα εφάρμοσαν μια γραμμική συνάρτηση. Αυτό είχε ως αποτέλεσμα να αποτραπεί η ανάγκη για την εφαρμογή μιας συμβατικής μεθόδου

γραμμικού προγραμματισμού ακεραίων η οποία θα απαιτούσε υπερβολικό αριθμό μεταβλητών ελέγχου. Τα αποτελέσματα έδειξαν ότι άσχετα από το στιγμιότυπο τα νευρωνικά δίκτυα αποδίδουν καλύτερα.

Μια άλλη εργασία που βασίζεται στα νευρωνικά δίκτυα για το χρονοπρογραμματισμό είναι αυτή των Sabuncuoglu και Gurgun (1996). Στην εργασία αυτή εφαρμόζεται μια 3-διάστατη δομή η οποία αυξάνεται με έναν εξωτερικό επεξεργαστή και αποδίδει τοπική αναζήτηση υπό τη μορφή του αλγορίθμου του κατωφλίου (threshold algorithm) [138].

3.3.4. Μέθοδοι τοπικής αναζήτησης και μετα–ευρετικές προσεγγίσεις

Όταν θέλουμε να εξάγουμε μια αλγοριθμική λύση για ένα δεδομένο υπολογιστικό πρόβλημα βελτιστοποίησης *P* όπου *R* είναι ένα σύνολο από εφικτές λύσεις στο *P*, πολλές φορές είναι απαραίτητο να ορίσουμε σχηματισμούς. Για παράδειγμα ένας σχηματισμός αποτελείται από ένα πεπερασμένο σύνολο λύσεων, μια συνάρτηση κόστους η οποία πρέπει να βελτιστοποιηθεί και ένας μηχανισμός γέννησης (generation mechanism), ο οποίος δημιουργεί μεταβάσεις από τον έναν σχηματισμό στον άλλον μέσω μικρών αλλαγών. Οι μέθοδοι που βασίζονται στη προηγούμενη περιγραφή ονομάζονται μέθοδοι τοπικής αναζήτησης [2].

Στις μεθόδους τοπικής αναζήτησης ο μηχανισμός γέννησης σχεδιάζει έναν «γείτονα» για κάθε σχηματισμό. Ένα γείτονας $\underline{N}(x)$ είναι μια συνάρτηση η οποία ορίζει μια απλή μετάβαση από τη λύση *x* σε μια άλλη λύση επιδεικνύοντας μια μικρή αλλαγή που μπορεί να θεωρηθεί και ως μικρή διαταραχή. Κάθε λύση $x' \in N(x)$ μπορεί να προσεγγιστεί άμεσα από τη *x* όταν πραγματοποιείται μια τέτοια μετάβαση. Επίσης υποτίθεται πως διακρατείται μια συμμετρία για την πλειοψηφία των αναζητήσεων γειτόνων που σημαίνει ότι η x' είναι γειτονική της *x* αν και μόνο αν η *x* είναι γειτονική της x'. Ο βασικός στόχος αυτού του είδους των στρατηγικών είναι μια προοδευτική διαταραχή των τρεχόντων σχηματισμών με σκοπό την άμεση αναζήτηση βελτιωμένων λύσεων. Η βελτίωση μπορεί να ειδωθεί σε κάθε βήμα από τυποποιημένες ***μεθόδους ανόδου* (*ascent methods*)** ή μπορεί να ειδωθεί αν υπάρχει μεγάλος αριθμός βημάτων από πιο προηγμένες μεθόδους. Στις μεθόδους αυτές οι επιλογή των γειτόνων γίνεται με βάση κάποια κριτήρια επιλογής.

Σχετικά τώρα με το πρόβλημα του χρονοπρογραμματισμού εργασιών η επίλυσή του μπορεί να θεωρηθεί ως μια συλλογή από τοπικές αποφάσεις αναφορικά με το πια

θα είναι η επόμενη διεργασία που θα χρονοπρογραμματιστεί. Οι Dorndorf και Pesch (1995) θεωρούν ότι θα πρέπει να κατασκευαστεί ένα πλαίσιο το οποίο να καθοδηγεί αυτές τις τοπικές αποφάσεις μέσω του πεδίου αναζητήσεων με σκοπό να προσδιοριστεί η καλύτερη λύση σε συγκεκριμένο χρόνο [54]. Σε ένα τέτοιο πλαίσιο οι τοπικές αποφάσεις οδηγούνται πέρα από το τοπικό βέλτιστο από μια μετα–στρατηγική (meta – strategy). Αυτό με τη σειρά του οδηγεί σε ***επαναλαμβανόμενους αλγορίθμους τοπικής αναζήτησης (iterated local search algorithms) ή μετα-ευρετικές προσεγγίσεις*** που επιτρέπει να επιτευχθούν καλές λύσεις. Στη συνέχεια θα αναφερθούμε σε ορισμένες από τις μεθόδους αυτές. Οι μετα –ευρετικές τεχνικές είναι ότι πιο πρόσφατο έχει αναπτυχθεί στις προσεγγιστικές μεθόδους αναζήτησης για την επίλυση σύνθετων προβλημάτων βελτιστοποίησης [125]. Οι μετα–ευρετικές στο Π_j βασίζονται σε στρατηγικές γειτνίασης που αναπτύχθηκαν από διάφορους ερευνητές [77], [115], [158], [124].

3.3.4.1. Μέθοδοι που βασίζονται στο χώρο του προβλήματος

Οι μέθοδοι που βασίζονται στο χώρο προβλήματος είναι μια ειδική κατηγορία ευρετικών η οποία δημιουργεί διαφορετικές λύσεις έναρξης, χρησιμοποιώντας συγκεκριμένες γρήγορες τεχνικές κατασκευής του προβλήματος και οι οποίες ενισχύονται από τη τοπική αναζήτηση. Παραδείγματα τέτοιων μεθόδων αποτελούν η αναζήτηση στο χώρο του προβλήματος και στο χώρο των ευρετικών (search in problem space and in heuristic space) και η μέθοδος της τυχαίας και άπληστης προσαρμοσμένης αναζήτησης (greedy randomized adaptive search procedure – GRASP).

3.3.4.2. Αλγόριθμοι Κατωφλίου

Οι Αλγόριθμοι Κατωφλίου αποτελούν μια από τις πιο δημοφιλής κατηγορίες μεθόδων επαναλαμβανόμενης τοπικής αναζήτησης. Συγκεκριμένα οι αλγόριθμοι κατωφλίου επιλέγουν ένα νέο σχηματισμό αν η διαφορά στο κόστος μεταξύ μιας γειτονικής λύσης και της τρέχουσας λύσης είναι κάτω από ένα δεδομένο επίπεδο κατωφλίου (L) πχ $f(x') - f(x) < L$.

Ένα από τα απλούστερα παραδείγματα τέτοιων αλγορίθμων αποτελεί η ***επαναληπτική βελτίωση* (*iterative improvement -IM*)**. Εδώ το κατώφλι τίθεται 0 και γίνονται αποδεκτοί μόνο οι καλύτεροι σχηματισμοί. Από ένα αρχικό χρονοπρόγραμμα, γενόμενο με τυχαίο τρόπο, κατευθύνεται η αναζήτηση στο τοπικό βέλτιστο από τη

στιγμή που η λύση είναι το ίδιο ή ελάχιστα καλύτερη από όλες τις λύσεις με τις οποίες γειτνιάζει. Η τεχνική της επαναληπτικής βελτίωσης αποτελεί τη πιο απλή κατηγορία τεχνικών επαναλαμβανόμενης τοπικής αναζήτησης και στην ουσία είναι η βάση για άλλες πιο εξελιγμένες μεθόδους.

Μια άλλη από τις μεθόδους που βασίζονται στους αλγορίθμους κατωφλίου είναι οι ***αλγόριθμοι αποδοχής κατωφλίου* (*Threshold Accepting –TA algorithms*)**. Οι αλγόριθμοι αποδοχής κατωφλίου έχουν το χαρακτηριστικό ότι το κατώφλι (*L*) δεν είναι αρνητικό. Αρχικά στο *L* δίνεται μια πολύ μεγάλη τιμή και σταδιακά αυτή μειώνεται στο μηδέν και συνεπώς επιλέγονται μόνο οι καλύτεροι σχηματισμοί. Οι αλγόριθμοι ΤΑ δεν έχουν εφαρμοστεί πολύ στο Π_j. Η μοναδική εφαρμογή έχει πραγματοποιηθεί από τους Aarts *et al.* (1994) σε ένα πρόβλημα 10x10 [3]. Δύο επιμέρους κατηγορίες των αλγορίθμων ΤΑ αποτελούν η ***Great Deluge Algorithm* (GDA) και η *Record-to-Record Travel* (RRT)**

Η μέθοδος ***Βελτιστοποίηση Μεγάλου Βήματος* (*Large Step Optimization*)** αναπτύχθηκε από τους Martin, Otto και Felten (1989, 1992) και είναι μια μέθοδος βελτιστοποίησης δυαδικής φάσης η οποία περιλαμβάνει ένα μεγάλο βήμα και ένα μικρό βήμα. Το μικρό βήμα κυρίως πραγματοποιείται με μετα –ευρετικές, ενώ το μεγάλο βήμα εμπλέκει συγκεκριμένες τεχνικές οι οποίες επιτρέπουν να ξεπεραστεί το τοπικό ελάχιστο [112], [113]. Η τεχνική αυτή είναι σχετικά και καινούργια και συνεπώς έχει περιορισμένη εφαρμογή στο Π_j.

Στη τεχνική ***simulated annealing* (SA)** το κατώφλι είναι θετικό και στοχαστικό. Η SA είναι μια τυχαία προσανατολισμένη τεχνική αναζήτησης η οποία προήλθε από άλλους επιστημονικούς κλάδους όπως αυτόν της στατιστικής φυσικής. Η πιο σημαντική συνεισφορά αναφορικά με τις συναρτήσεις γειτνίασης στο Π_j έχει γίνει με την εργασία των Van Laarhooven *et al.* το 1992 [158]. Συγκεκριμένα περιλαμβάνονται μόνο οι κινήσεις εκείνες που μπορούν να επιτευχθούν αντιστρέφοντας τη σειρά επεξεργασίας ενός γειτονικού ζεύγους κρίσιμων διεργασιών, δεδομένων των συνθηκών υπό τις οποίες πρέπει να πραγματοποιηθεί η επεξεργασία τους στην ίδια μηχανή. Μια τέτοια γειτνίαση βασίζεται στις ακόλουθες ιδιότητες:

- Αν $x \in \Re$ είναι μια εφικτή λύση, τότε ανταλλάσσοντας δύο γειτονικές κρίσιμες διεργασίες οι οποίες απαιτούν την ίδια μηχανή το γεγονός αυτό δεν μπορεί να μας οδηγήσει σε μη εφικτή λύση.

- Αν η ανταλλαγή μεταξύ δύο γειτονικών αλλά μη κρίσιμων διεργασιών οδηγήσει σε μια εφικτή λύση x', τότε η κρίσιμη διαδρομή στη λύση x' δεν μπορεί να είναι μικρότερη από τη κρίσιμη διαδρομή στη *x*, επειδή η κρίσιμη διαδρομή στη *x* εξακολουθεί να υφίσταται στη x'.
- Ξεκινώντας από οποιαδήποτε εφικτή λύση *x*, υπάρχουν κάποιες ακολουθίες κινήσεων οι οποίες θα προσεγγίσουν τη βέλτιστη λύση x^*. Η ιδιότητα αυτή είναι γνωστή και ως ιδιότητα συνδεσιμότητας (connectivity property).

3.3.4.3. Γενετικοί Αλγόριθμοι

Είναι μια τεχνική βελτιστοποίησης η οποία προέρχεται από τη παρατήρηση των φυσικών διαδικασιών των γενετικών αλγορίθμων. Οι γενετικοί αλγόριθμοι βασίζονται σε αφαιρετικά μοντέλα φυσικής εξέλιξης. Η ανάλυση που θα παρουσιαστεί στη συνέχεια βασίζεται στη ταξινόμηση των Cheng *et al.* (1996) οι οποίοι έχουν ταξινομήσει όλα τα σχέδια αναπαράστασης των τελευταίων ετών σε 9 κατηγορίες [41]:

1. Με βάση τις διεργασίες
2. Με βάση τις εργασίες
3. Με βάση τις σχέσεις ζεύγους μεταξύ των εργασιών
4. Με βάση το χρόνο ολοκλήρωσης
5. Τυχαία κλειδιά
6. Με βάση λίστα προτίμησης
7. Με βάση κανόνα προτεραιότητας
8. Με βάση διαζευκτικούς γράφους
9. Με βάση τις μηχανές

Όλα αυτά τα παραπάνω σχέδια αναπαράστασης μπορούν να ομαδοποιηθούν σε δύο βασικές κωδικοποιημένες προσεγγίσεις, άμεσες και έμμεσες. Σύμφωνα με την άμεση προσέγγιση ένα Π_j χρονοπρόγραμμα κωδικοποιείται ως ένα χρωμόσωμα και ένας γενετικός τελεστής (genetic operator) χρησιμοποιείται για να εξελίξει αυτά τα χρωμοσώματα σε καλύτερα χρονοπρογράμματα. Οι κατηγορίες 1 έως 5 που αναφέρθηκαν στη παραπάνω λίστα αποτελούν παραδείγματα αυτής της στρατηγικής. Στην έμμεση προσέγγιση μια ακολουθία από αποφάσεις προτιμήσεων κωδικοποιείται σε ένα χρωμόσωμα και ένας γενετικός τελεστής εφαρμόζεται για να βελτιώσει τη σειρά

των διαφόρων προτιμήσεων. Στη συνέχεια ένα Π_j χρονοπρόγραμμα δημιουργείται από την ακολουθία των προτιμήσεων. Οι κατηγορίες 6 έως 9 είναι παραδείγματα της έμμεσης προσέγγισης.

Μια από τις πρώτες εφαρμογές που είχε η μέθοδος των γενετικών αλγορίθμων στο Π_j ήταν από το Davis (1985) ο οποίος χρησιμοποίησε την έμμεση προσέγγιση [48]. Συγκεκριμένα κατασκεύασε μια προτιμώμενη σειρά από διεργασίες σε κάθε μηχανή. Η προσέγγιση αυτή επεκτάθηκε αργότερα από τους Falkenauer και Bouffouix (1991), οι οποίοι κωδικοποίησαν τις εργασίες που έπρεπε να επεξεργαστούν σε μια μηχανή ως μια λίστα προτιμήσεων που περιλαμβάνει μια σειρά από σύμβολα [56].

Μια από τις πιο πρόσφατες αναπαραστάσεις με βάση λίστες προτίμησης πραγματοποιήθηκε από τους Kobayashi *et al.* (1995), όπου ένα χρωμόσωμα είναι μια σειρά από σύμβολα μήκους *n* και κάθε σύμβολο αναγνωρίζει τη διεργασία η επεξεργασία της οποίας πρέπει να υλοποιηθεί σε μια μηχανή [98]. Το 1992 οι Tamaki και Nishikawa εφάρμοσαν μια έμμεση αναπαράσταση βασιζόμενοι σε διαζευκτικούς γράφους (κατηγορία 8). Στη περίπτωση αυτή ένα χρωμόσωμα περιέχει ένα δυαδικό σύμβολο που αντιστοιχεί σε μια ταξινομημένη λίστα διαζευκτικών ακμών [151].

Αναφορικά με τις άμεσες προσεγγίσεις, μια από τις πρώτες εργασίες έγινε από τους Nakano και Yamada (1991), οι οποίοι δημιούργησαν μια δυαδική κωδικοποίηση βασιζόμενοι στη σχέση προτεραιότητας των διεργασιών που αναφέρονται στην ίδια μηχανή [120].

Μια σχετικά νέα άμεση προσέγγιση είναι αυτή που βασίζεται στα τυχαία κλειδιά (κατηγορία 5) [24]. Για ένα Π_j πρόβλημα κάθε γονίδιο (gene) που είναι ένα τυχαίο κλειδί αποτελείται από δύο μέρη: ένα ακέραιο από το σύνολο $\{1,2,3,\ldots,m\}$ και ένα κλάσμα που δημιουργείται τυχαία από το διάστημα (0,1). Το ακέραιο μέρος του γονιδίου είναι η εκχώρηση της μηχανής, ενώ το κλασματικό μέρος, ταξινομούμενο σε μια μη μειούμενη σειρά, προσδιορίζει την ακολουθία των διεργασιών για κάθε μηχανή.

Παρά τη πληθώρα των γενετικών αλγορίθμων που εφαρμόστηκαν στο Π_j τα αποτελέσματα που αυτές οι μέθοδοι πέτυχαν ήταν αρκετά φτωχά σε συγκεκριμένες καταστάσεις. Για την αντιμετώπιση αυτών των προβλημάτων εφαρμόστηκε μια μέθοδος που ανήκει στην Εξελικτική Υπολογιστική, ονομάζεται ***Γενετική Τοπική Αναζήτηση*** **(*Genetic Local Search* –*GLS*)** και η οποία συνδυάζει τοπική αναζήτηση γειτνίασης και στρατηγική γενετικών αλγορίθμων. Η ανωτερότητα της μεθόδου GLS έναντι των γενετικών αλγορίθμων τονίστηκε από τους Della Croce *et al.* (1994) οι οποίοι

ενσωμάτωσαν διάφορες γειτονικές δομές μέσα σε γενετικούς αλγορίθμους και το γεγονός αυτό οδήγησε σε βελτιωμένα αποτελέσματα [50]. Μια από τις πιο γνωστές και επιτυχημένες εργασίες πάνω στις μεθόδους GLS ήταν αυτή των Dorndorf και Pesch (1995) οι οποίοι πρότειναν δύο προσεγγίσεις για την επίλυση του Π_j. Συγκεκριμένα, η πρώτη προσέγγιση οδηγεί σε ένα πιθανοτικό συνδυασμό 12 κανόνων διεκπεραίωσης με προτεραιότητες (pdrs) και ονομάστηκε P-GA ενώ η δεύτερη προσέγγιση αναφέρεται ως SB-GA και ελέγχει την επιλογή των κόμβων για τη μέθοδο SB(II). Τα αποτελέσματα έδειξαν ότι η SB-GA είναι καλύτερη από τη P-GA [54].

3.3.4.4. Αναζήτηση Ταμπού

Η επαναληπτική τεχνική βελτιστοποίησης Αναζήτηση Ταμπού έχει την αρχή της στην ευφυή επίλυση προβλημάτων και προέρχεται από τις εργασίες του Glover [71], [72], [73], [74]. Πρόκειται για μια απλή και ντετερμινιστικά προσανατολισμένη μεθοδολογία αναζήτησης η οποία περιορίζει την αναζήτηση και επιζητά τοπική βελτιστοποίηση ταξινομώντας το ιστορικό της αναζήτησης στη μνήμη της. Η μεθοδολογία αυτή απαγορεύει τις μετακινήσεις (γι αυτό και ονομάζεται tabu) σε γειτονικές που έχουν συγκεκριμένες ιδιότητες έχοντας ως τελικό σκοπό να οδηγήσει την αναζήτηση μακριά από λύσεις οι οποίες φαίνεται να μοιάζουν ή να αντιγράφουν λύσεις που έχουν επιτευχθεί προγενέστερα. Τα κριτήρια επιτρέπουν να επιλεγεί μια κίνηση tabu αν επιτυγχάνει ένα προκαθορισμένο επίπεδο ποιότητας.

Στην τεχνική της αναζήτησης ταμπού υπάρχουν τριών ειδών μνήμες: η βραχυπρόθεσμη, η μεσαία και η μακροπρόθεσμη. Η μεσαία και η μακροπρόθεσμη μνήμη βοηθούν στο να υπάρχει μια ευρύτερη εξερεύνηση του χώρου των λύσεων. Οι στρατηγικές που βασίζονται στη μεσαίου επιπέδου μνήμη στηρίζονται στη τροποποίηση των κανόνων επιλογής έτσι ώστε αν ενθαρρύνονται κινήσεις και λύσεις που έχουν βρεθεί στο παρελθόν και είναι καλές. Οι μέθοδοι που βασίζονται στη μακροπρόθεσμη μνήμη χωρίζουν την αναζήτηση σε περιοχές που προηγουμένως δεν έχουν εξερευνηθεί. Επίσης τροποποιούν τους κανόνες επιλογής ώστε να ενσωματώνουν στις λύσεις ιδιότητες που δεν χρησιμοποιούνται συχνά.

Η εργασίες των Laguna *et al.* (1991, 1993) ήταν από τις πρώτες εργασίες που εφαρμόστηκαν στο χρονοπρογραμματισμό [103], [104]. Συγκεκριμένα δημιούργησαν τρεις στρατηγικές έρευνας βασιζόμενοι σε απλούς ορισμούς κινήσεων. Οι Barnes και Laguna (1993) πρότεινα έξι πρωτεύοντα στοιχεία που επιτρέπουν αποτελεσματικό

χρονοπρογραμματισμό παραγωγής με χρήση αναζήτησης ταμπού, δίνοντας έμφαση κυρίως στη μεσαίου και μακροπρόθεσμου επιπέδου μνήμη.

Λίγο αργότερα το 1995 ο Hara πρότεινε μια τεχνική γνωστή με το όνομα ***Αναζήτηση Ελαχίστων Συγκρούσεων*** **(*Minimum Conflict Search –MCS*)** χρησιμοποιώντας τους περιορισμούς που ορίζονται στη δομή της βραχυπρόθεσμης μνήμης στην αναζήτηση ταμπού [80]. Η ελάχιστη σύγκρουση είναι η ελάχιστη αποτελεσματική συνθήκη η οποία δεν είναι βέλτιστη. Στο Π_j η ελάχιστη σύγκρουση είναι η κρίσιμη διαδρομή.

Στον αλγόριθμο των Barnes και Chambers (1995) η μέθοδος ταμπού εφαρμόζεται σε ένα προκαθορισμένο εύρος τιμών βραχυπρόθεσμης μνήμης και κάθε φορά που μια νέα καλύτερη λύση έρχεται στο προσκήνιο αποθηκεύεται σε μια λίστα για μελλοντική επανάκτηση. Μετά από ένα συγκεκριμένο χρονικό διάστημα η ακολουθία στη κορυφή της λίστας χρησιμοποιείται ως σημείο έναρξης για μια νέα αναζήτηση ενσωματωμένη με τη μικρότερη τιμή στη βραχυπρόθεσμη μνήμη και εν συνεχεία με όλες τις άλλες τιμές με αυξητική σειρά. Από τη στιγμή που έχουν επιλεγεί όλες οι τιμές στη μνήμη επιλέγεται από τη λίστα η λύση και η διαδικασία συνεχίζεται για όλα τα εναπομείναντα χρονοπρογράμματα [22].

Μια σημαντική εργασία μεθόδου αναζήτησης ταμπού στο Π_j είναι αυτή του Taillard (1994) στην οποία χρησιμοποιείται μια στρατηγική που επιταχύνει την αναζήτηση αποτρέποντας όμως ταυτόχρονα την ανάγκη για εκ νέου υπολογισμό των χρόνων έναρξης όλων των διεργασιών με σκοπό τον προσδιορισμό του κόστους της κίνησης [150]. Παρόλα αυτά η στρατηγική αυτή είναι αποτελεσματική μόνο υπό συγκεκριμένες συνθήκες και θεωρείται μια στρατηγική γρήγορης εκτίμησης.

Η καλύτερη μέθοδος αναζήτησης ταμπού που έχει παρουσιαστεί πιο πρόσφατα είναι αυτή των Nowicki και Smutnicki (1996) [124]. Στη δεδομένη μέθοδο εφαρμόζεται μια υψηλά περιορισμένη γειτνίαση, η οποία χωρίζει μια απλή κρίσιμη διαδρομή σε τμήματα (blocks). Αν υπάρχουν b τμήματα στη γειτονιά τότε για τα τμήματα $1<l<b$ οι πρώτες και οι τελευταίες δύο διεργασίες εναλλάσσονται. Αντίστοιχα για το τμήμα $l=1$ εναλλάσσονται μόνο οι τελευταίες δύο διεργασίες του συγκεκριμένου τμήματος, ενώ για το τμήμα $l=b$ εναλλάσσονται μόνο οι δύο πρώτες διεργασίες του τμήματος.

Οι εργασίες των Aarts [1] και των Ten Eikelder *et al.* [152] παρουσιάζουν τη κατασκευή διαφόρων ακολουθιακών (σειριακών) αλγορίθμων αναζήτησης ταμπού για το Π_j. Ανέπτυξαν μια τεχνική σειριακού μερικού υπολογισμού (partial evaluation) που

ονομάζεται bowtie και επιταχύνει τη στρατηγική εκτίμησης του Taillard (1994) κατά 35-40%. Ο αλγόριθμός τους μοιάζει σε αρκετά μεγάλο βαθμό με αυτόν που εφάρμοσαν οι Nowicki και Smutnicki. Στην ανάλυσή τους υποστήριξαν ότι δεν χρειάζονται περισσότεροι από 20 με 30 παράλληλοι επεξεργαστές για την εφαρμογή του αλγορίθμου.

Μια αρκετά πρόσφατη προσέγγιση είναι αυτή του Thomsen (1997) η οποία έχει εφαρμοστεί σε έναν αριθμό τυποποιημένων προβλημάτων καταφέρνοντας να βελτιώσει το άνω εύρος των προβλημάτων αυτών. Η προσέγγιση αυτή αποτελεί συνδυασμό της αναζήτησης ταμπού με αλγορίθμους διακλάδωσης και οριοθέτησης (BB). Στη τεχνική αυτή η στρατηγική διακλάδωσης και οριοθέτησης χρησιμοποιείται για διαφοροποίηση, αλλά με ένα περιορισμένο αριθμό επαναλήψεων και σε ένα περιορισμένο βάθος στο δέντρο αναζήτησης για να διατηρηθούν χαμηλοί οι υπολογιστικοί χρόνοι [153].

Κεφάλαιο 4:
Θεωρία αυτομάτων και χρονοπρογραμματισμός

4.1. Εισαγωγικά στοιχεία στη θεωρία αυτομάτων

Στο κεφάλαιο αυτό παρατίθενται τα βασικά στοιχεία της θεωρίας αυτομάτων στην οποία βασίζεται η μεθοδολογία της παρούσας διατριβής. Στο πλαίσιο αυτό δίνονται οι βασικοί ορισμοί και αναλύονται τα χαρακτηριστικά των αυτομάτων πεπερασμένων καταστάσεων, ενώ στη συνέχεια παρατίθεται μια πλήρης επισκόπηση των ερευνητικών προσεγγίσεων αντιμετώπισης προβλημάτων προγραμματισμού με χρήση αυτομάτων.

Τα πεπερασμένα αυτόματα και συγκεκριμένα μια επέκταση αυτών με ρολόγια πραγματικών τιμών που ονομάζεται “χρονισμένα αυτόματα (timed automata)” [13] έχουν παρουσιαστεί και εξετάζονται ερευνητικά τα τελευταία χρόνια ως μια σύγχρονη εναλλακτική προσέγγιση για την επίλυση προβλημάτων χρονοπρογραμματισμού. Η κεντρική ιδέα έγκειται στην μοντελοποίηση ενός προβλήματος χρονικού προγραμματισμού με χρήση αυτομάτων πεπερασμένων καταστάσεων και την επίλυση αυτών μέσω ανάλυσης προσεγγισιμότητας. Συγκεκριμένα τα συστήματα τα οποία μοντελοποιούνται ως χρονισμένα αυτόματα αναλύονται χρησιμοποιώντας εργαλεία επαλήθευσης μοντέλου με ενσωματωμένες τεχνικές ελέγχου. Η ανάλυση προσεγγισιμότητας είναι μια τεχνική ελέγχου προκειμένου να εξερευνηθεί το περιβάλλον

του συστήματος το οποίο μοντελοποιείται ως ένα αυτόματο ή ένα συνδεδεμένο σύνολο από αυτόματα, με σκοπό να προσδιοριστεί ένα προσβάσιμο υποσύνολο από καταστάσεις του συστήματος και να αξιολογηθεί κατά πόσον ικανοποιούνται συγκεκριμένες επιθυμητές συνθήκες. Η δύναμη αυτής της προσέγγισης μοντελοποίησης είναι η γραφική αναπαράσταση που βοηθάει στην αναπαράσταση πολύπλοκων συστημάτων με ευκολία και σαφήνεια. Επιπρόσθετα, η επιστημονική έρευνα στον τομέα της ανάλυσης προσεγγισιμότητας των χρονισμένων αυτομάτων έχει οδηγήσει στην επέκταση της τεχνικής από μια ποιοτική αξιολόγηση σε μία ποσοτική αξιολόγηση της συμπεριφοράς των συστημάτων, καθιστώντας την έτσι κατάλληλη και εφαρμόσιμη σε προβλήματα χρονοπρογραμματισμού. Η έρευνα πάνω στα χρονισμένα αυτόματα έχει παράλληλα οδηγήσει σε μια σειρά αποτελεσματικών και φιλικών προς τον χρήστη εργαλείων όπως τα Kronos, Uppaal, Supremica κ.α. τα οποία διευκολύνουν τη μοντελοποίηση και την ανάλυση χρονισμένων αυτομάτων. Σχετική περιγραφή των διαθέσιμων εργαλείων παρατίθεται στο επόμενο κεφάλαιο της εργασίας.

4.2. Αυτόματα Πεπερασμένων Καταστάσεων

Ένα αυτόματο πεπερασμένης κατάστασης είναι ένα ζεύγος της μορφής $A=(Q,\delta)$, όπου Q είναι ένα πεπερασμένο σύνολο καταστάσεων και $\delta \subseteq Q \times Q$ είναι μια σχέση μετάβασης. Η πεπερασμένη εκτέλεση ενός αυτομάτου ξεκινώντας από μια κατάσταση q_0 είναι μια ακολουθία από καταστάσεις

$$q_0 \to q_1 \to \ldots \to q_k$$

τέτοια ώστε $(q_i, q_{i+1}) \in \delta$ για κάθε i, $0 \le i \le k-1$.

Στην ορολογία των γράφων οι καταστάσεις ονομάζονται ***κόμβοι ή κορυφές* (*vertices or nodes*)**, οι μεταβάσεις ***ακμές ή τόξα* (*edges or arcs*)** και τέλος οι εκτελέσεις είναι οι **διαδρομές**. Χρησιμοποιούμε τους όρους $Succ(q)$ και $\Pr e(q)$ για να δηλώσουμε το σύνολο των ακόλουθων και προηγούμενων καταστάσεων μιας δεδομένης κατάστασης q:

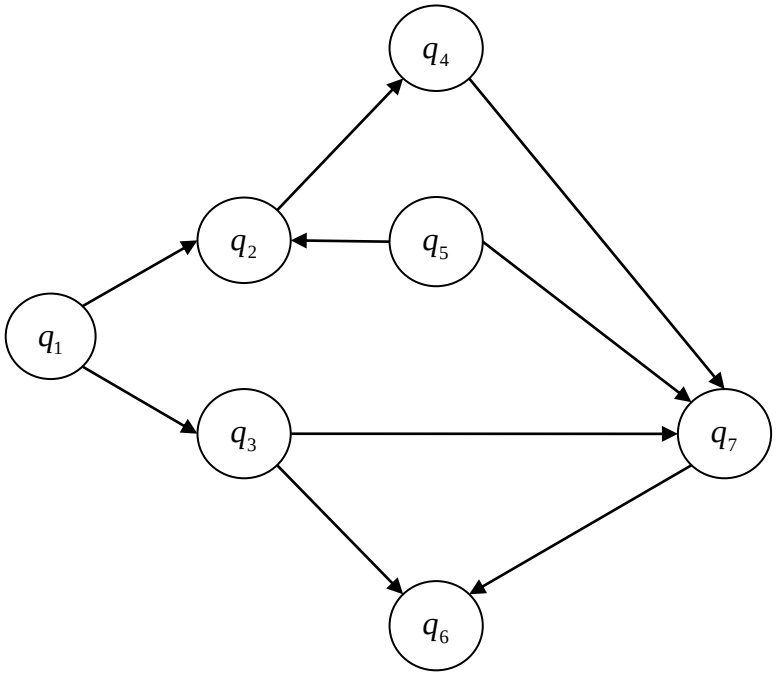
$$Succ(q) = \{q' : (q, q') \in \delta\}$$

και

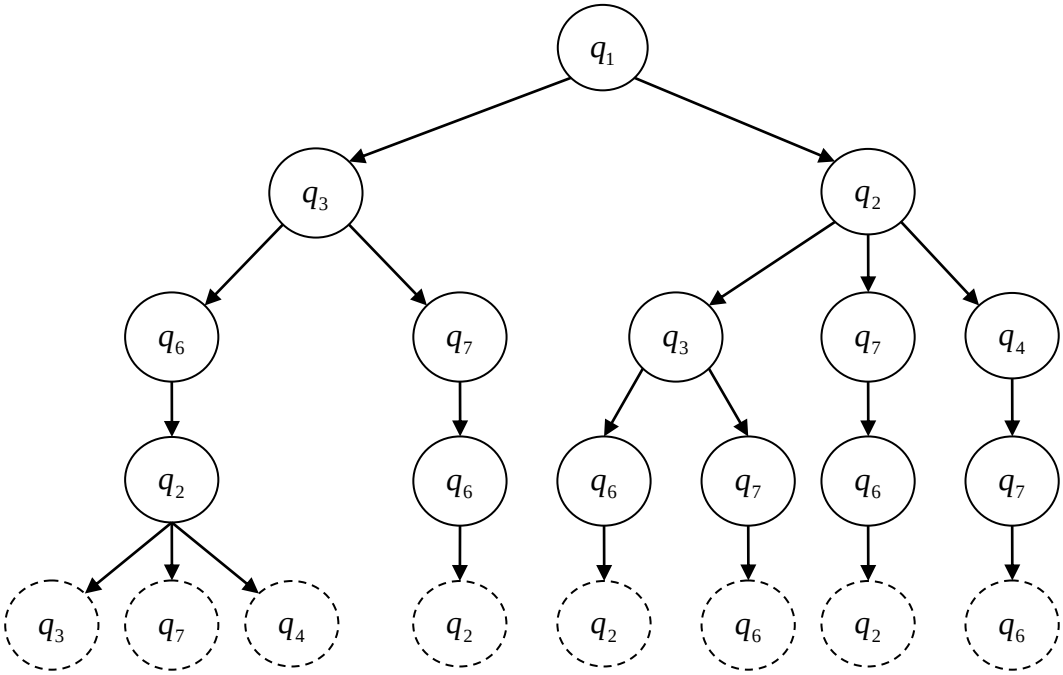
$$\Pr e(q) = \{q' : (q', q) \in \delta\}$$

Για να γίνει καλύτερα αντιληπτή η έννοια των πιθανών εκτελέσεων ενός αυτομάτου που ξεκινάει από μια αρχική κατάσταση, χρησιμοποιούμε το q-ανάπτυγμα

(q-unfolding) του αυτομάτου. Ουσιαστικά πρόκειται για ένα δέντρο που ξεκινάει από μια κατάσταση q και προσθέτουμε κόμβους και μεταβάσεις για κάθε ακόλουθη κατάσταση. Αν το αυτόματο περιλαμβάνει κυκλικές διαδρομές οι οποίες οδηγούν στην αρχική κατάσταση τότε το ανάπτυγμα είναι άπειρο. Σε αντίθετη περίπτωση το ανάπτυγμα είναι πεπερασμένο και το μέγεθός του θα αυξάνεται εκθετικά με το μέγεθος του αυτομάτου. Ένα τέτοιο αυτόματο απεικονίζεται στο Σχήμα 4, ενώ ένα αρχικό μέρος του αναπτύγματός του φαίνεται παρουσιάζεται στο Σχήμα 5 που ακολουθεί.



Σχήμα 4: Ένα αυτόματο A=(Q,δ)



Σχήμα 5: Το q -ανάπτυγμα του αυτομάτου του Σχήματος 4

Στη συνέχεια θα παρουσιαστούν οι διαφορετικές προσεγγίσεις που υπάρχουν στην αναζήτηση γράφων για την επίλυση προβλημάτων προσπελασιμότητας. Ένα τέτοιο πρόβλημα έχει να κάνει με το αν σε ένα αυτόματο με δύο καταστάσεις q και q' υπάρχει διαδρομή που να οδηγεί από την q στη q'. Τέτοια διαδρομή θα υπάρχει αν το ανάπτυγμα του αυτομάτου που ξεκινάει από τη κατάσταση q περιλαμβάνει το κόμβο με την ετικέτα q'. Ένας από τους πιο γνωστούς αλγορίθμους που μπορεί να χρησιμοποιηθεί για ελεγχθεί το γεγονός αυτό, είναι ο ***αλγόριθμος αναζήτησης προτεραιότητας εύρους* (*breath-first search algorithm – BFS*)**. Ο αλγόριθμος αυτός διατηρεί δύο δομές δεδομένων (data structures), ένα σύνολο ήδη εξερευνημένων καταστάσεων, δηλαδή καταστάσεων των οποίων οι ακόλουθες καταστάσεις έχουν εξερευνηθεί και ένα ταξινομημένο σύνολο καταστάσεων που τίθενται προς εξερεύνηση. Ο αλγόριθμος τερματίζει, είτε με τη προσπέλαση της κατάστασης q', οπότε απαντάμε «ναι», είτε με τη προσπέλαση όλων των καταστάσεων που μπορούν να προσπελαστούν από τη q χωρίς όμως να προσπελάσουμε τη q'. Ο αλγόριθμος θα ολοκληρωθεί και το γεγονός αυτό εξασφαλίζεται λόγω του ότι ο το σύνολο των καταστάσεων Q είναι πεπερασμένο.

Αλγόριθμος Προσπελασιμότητας BFS (A, q, q')

$\text{Waiting} := \{q\};$
$Explored := \{\ \};$
while (*Reached = no* and $Waiting \neq \varnothing$) **do**
begin
Pick first $v \in Waiting$;
if $(v \notin Explored)$ **then**
begin
for *every u such that* $v \in Succ(v) \wedge v \notin Explored$ **do**
if $(v = q')$ **then**
Reached := yes;
else
Insert v into the end of Waiting;
Insert v into Explored;
end
Remove v from Waiting;
end
return (*Reached*);

Αξίζει να σημειωθεί πως ο αλγόριθμος BFS (σε αντίθεση με τον DFS όπως θα αναλυθεί παρακάτω) εξερευνά το ανάπτυγμα του αυτομάτου σε επίπεδα. Αυτό σημαίνει ότι η σειρά με την οποία εξερευνά τους διαφόρους κόμβους του δέντρου είναι σταθερή με βάση την απόσταση που βρίσκονται από τη ρίζα. Ο όρος απόσταση εδώ σημαίνει τον αριθμό των μεταβάσεων. Αν θέλουμε να βρούμε μια διαδρομή, όταν αυτή βέβαια υπάρχει, μπορούμε να τροποποιήσουμε τον αλγόριθμο έτσι ώστε να αποθηκεύει κάθε εισερχόμενο κόμβο που μας έχει οδηγήσει στη συγκεκριμένη διαδρομή.

Ένας άλλος αλγόριθμος που χρησιμοποιείται εναλλακτικά του BFS είναι ο ***αλγόριθμος αναζήτησης προτεραιότητας βάθους* (*Depth-First Search algorithm – DFS*).** Ενώ ο αλγόριθμος BFS προχωράει «παράλληλα» αναφορικά με τις διαδρομές, ο DFS προσπαθεί να προσπελάσει τη κατάσταση q μέσω μιας διαδρομής και ελέγχει σε όλο το βάθος αυτής. Αν αυτό δεν καταστεί εφικτό τότε προχωράει στην επόμενη διαδρομή. Από τεχνικής πλευράς η διαφορά μεταξύ των δύο αλγορίθμων είναι το μέρος όπου οι κατά την εξερεύνηση οι νέοι κόμβοι προστίθενται στην λίστα αναμονής. Αν δηλαδή θα είναι στο τέλος (περίπτωση BFS) ή αν θα είναι στην αρχή (περίπτωση DFS).

Με άλλα λόγια η λίστα που ακολουθείται στον αλγόριθμο BFS είναι FIFO (first in first out), ενώ στον DFS είναι LIFO (last in first out).

Αλγόριθμος Προσπελασιμότητας DFS

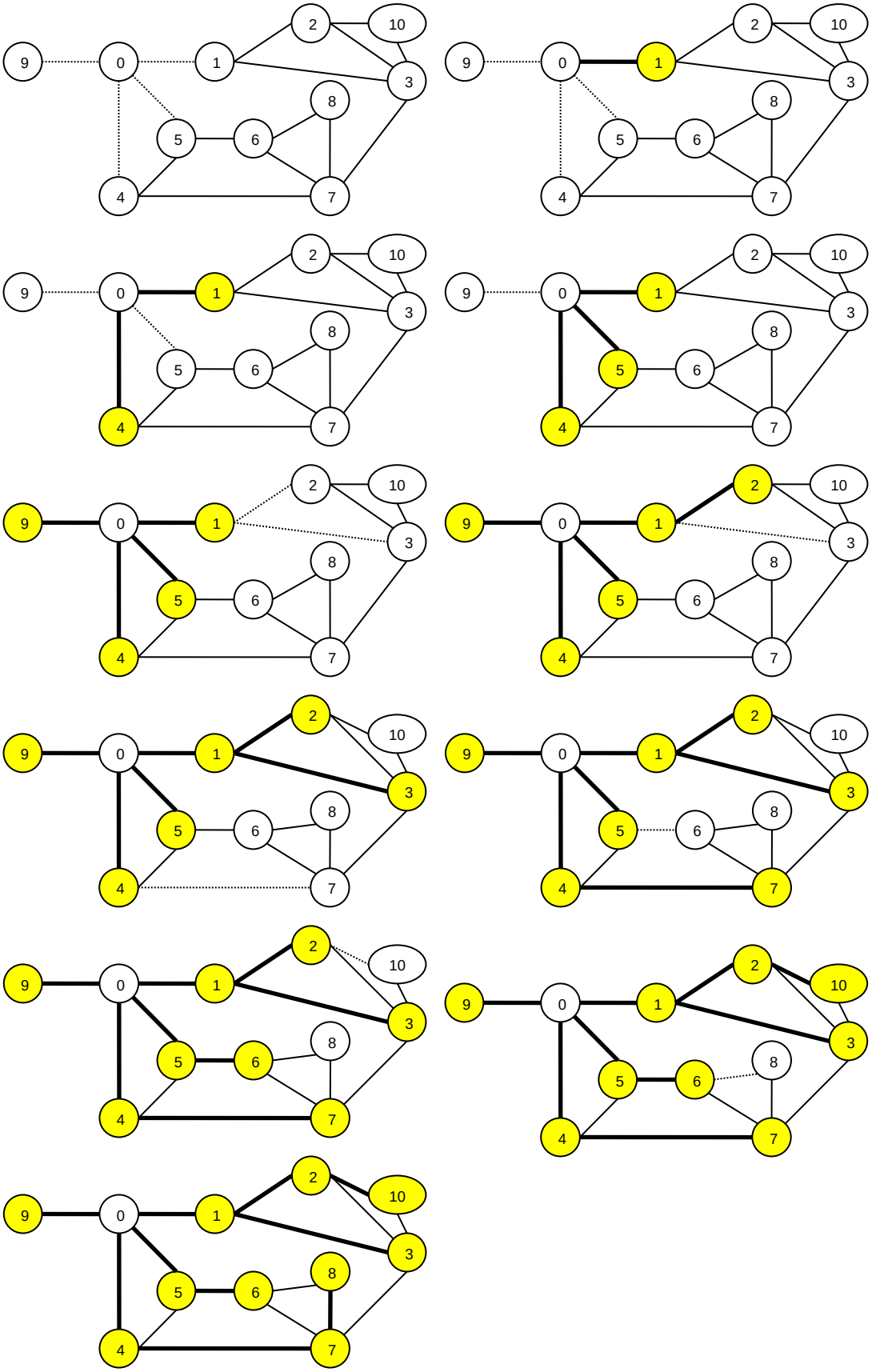
$Waiting := \{q\};$
$Explored := \{\ \};$
$\mathrm{Re}\,ached := no;$
while (*Reached = no and* $Waiting \neq \varnothing$) **do**
begin
  Pick first $v \in Waiting$;
  if ($v \notin Explored$) **then**
  begin
    for *every v such that* $v \in Succ(v) \wedge v \notin Explored$ **do**
      if ($v = q'$**) then**
        Reached := yes;
      else
        Insert v at the begging of Waiting;
    Insert v into Explored;
  end
  Remove v from Waiting;
end
Return (*Reached*);

Οι δύο προαναφερθέντες αλγόριθμοι έχουν ένα βασικό χαρακτηριστικό. Συγκεκριμένα η σειρά με την οποία επιλέγεται κάθε φορά να γίνει η εξερεύνηση των καταστάσεων (κόμβων) μπορεί να έχει σημαντική επίπτωση στη συμπεριφορά του αλγορίθμου, ιδίως στη περίπτωση του DFS όταν υπάρχει διαδρομή.

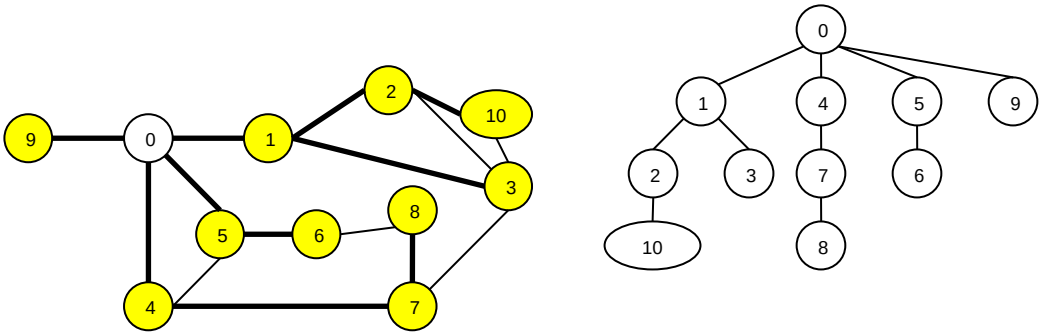
Στη συνέχεια θα εξετάσουμε δύο παραδείγματα. Το πρώτο έχει να κάνει με την εφαρμογή των BFS και DFS αλγορίθμων στη περίπτωση των μη κατευθυνόμενων και το δεύτερο στη περίπτωση των κατευθυνόμενων γράφων.

Παράδειγμα 1ο :

Στο Σχήμα 6 παρουσιάζεται ο τρόπος με τον οποίο εφαρμόζεται ο αλγόριθμος BFS στο συγκεκριμένο παράδειγμα που αναφέρεται σε μη κατευθυνόμενο γράφο.



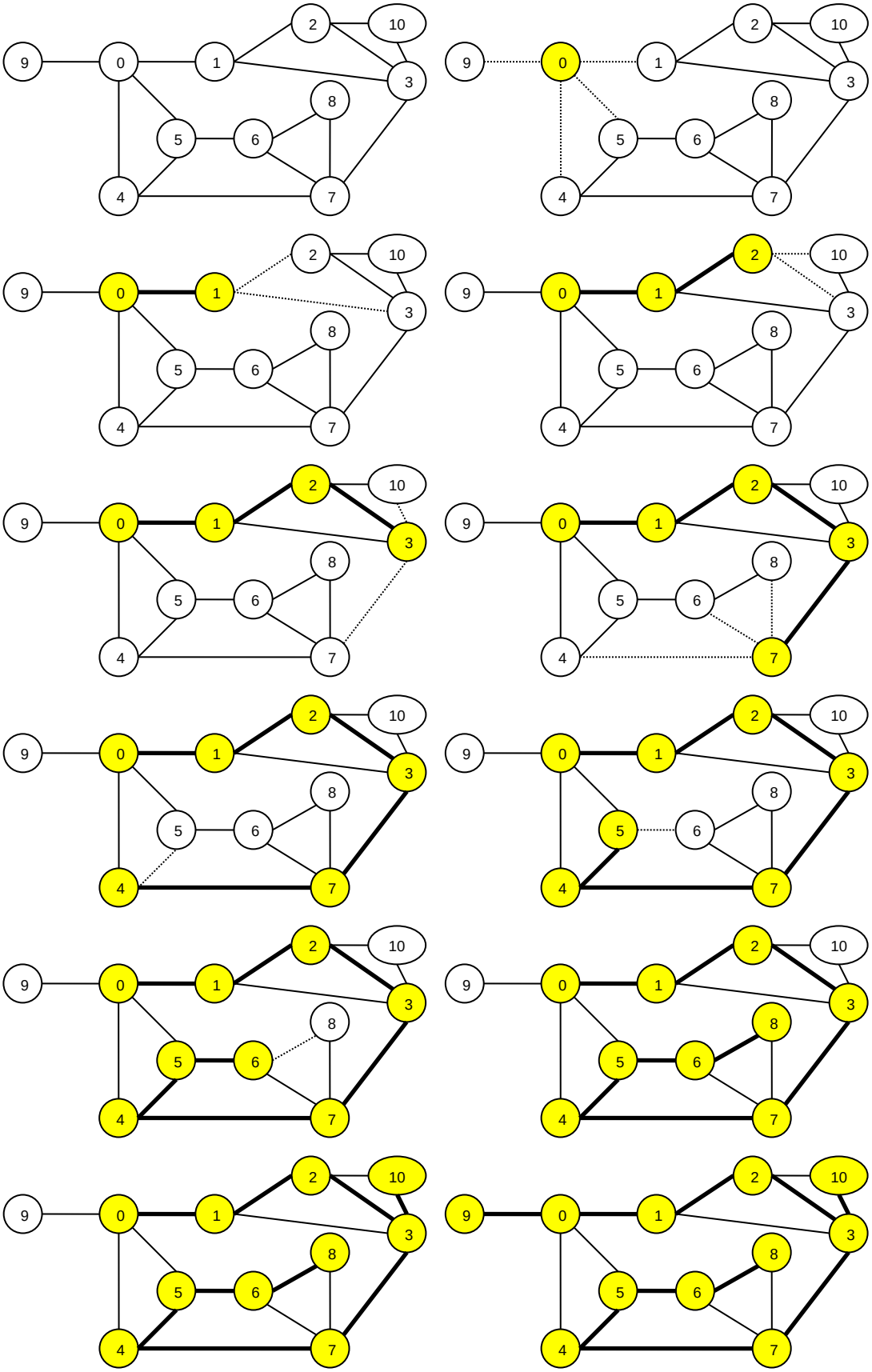
Σχήμα 6: Εφαρμογή του αλγορίθμου BFS σε μη κατευθυνόμενο γράφο



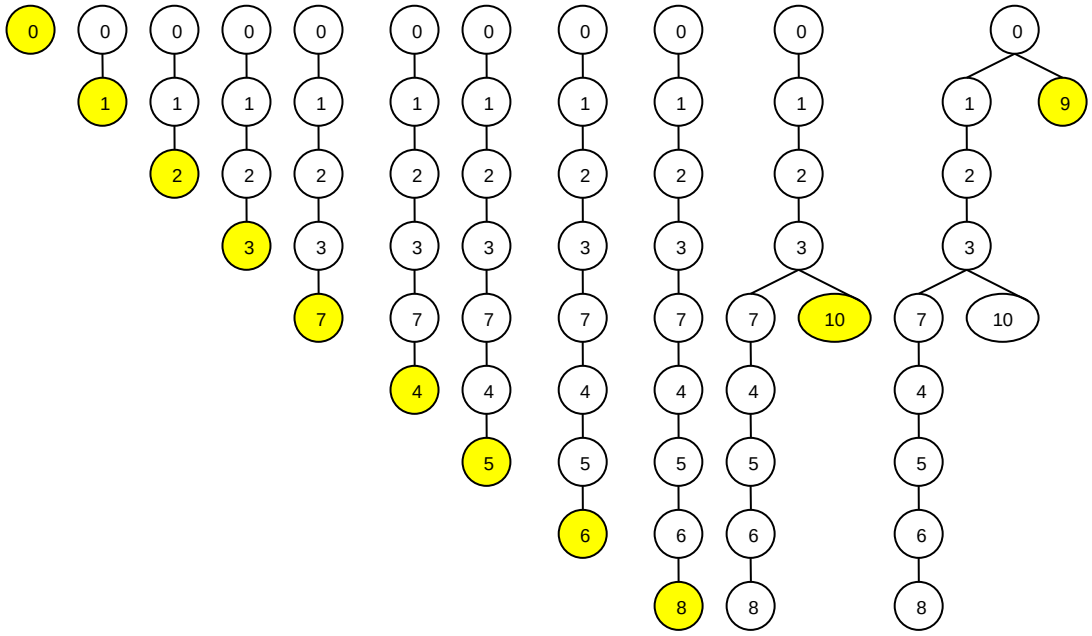
Σχήμα 7: Ο αλγόριθμος BFS και το δέντρο αναζήτησης

Η διαδικασία εφαρμογής του αλγορίθμου BFS στο μη κατευθυνόμενο γράφο του Σχήματος 6 έχει ως εξής: Επιλέγουμε την κορυφή 0 θεωρώντας την ως αφετηρία και τη χαρακτηρίζουμε εξερευνημένη. Στη συνέχεια επισκεπτόμαστε τις γειτονικές της κορυφές (1, 4, 5 και 9) τις οποίες με τη σειρά τους τις χαρακτηρίζουμε και αυτές ως εξερευνημένες. Ακολούθως αφού τις έχουμε θεωρήσει εξερευνημένες επιλέγουμε την 1 (που είναι πρώτη από αυτές που ήδη έχουν εξερευνηθεί) και επισκεπτόμαστε τις γειτονικές της (2 και 3) τις οποίες τις χαρακτηρίζουμε και αυτές εξερευνημένες. Συνεχίζουμε την ίδια διαδικασία με τις γειτονικές κορυφές της 4, της 5 και της 9 κοκ, τηρώντας όμως πάντα τη διάταξη FIFO.

Στο Σχήμα 8 φαίνεται η εφαρμογή του αλγορίθμου DFS στον ίδιο μη κατευθυνόμενο γράφο που παρουσιάστηκε πρωτύτερα. Στη περίπτωση του DFS όλες οι κορυφές πριν ξεκινήσουμε θεωρούνται ανεξερεύνητες. Εν συνεχεία επιλέγουμε μια κορυφή *u* και τη θεωρούμε αυτομάτως ως εξερευνημένη. Αν η *u* έχει γειτονική κορυφή μη εξερευνημένη *w* τότε την επισκεπτόμαστε και την καθιστούμε ως εξερευνημένη. Αν από τη *u* υπάρχουν περισσότερες της μιας ανεξερεύνητες κορυφές τότε επιλέγουμε αυτή με το μικρότερο όνομα λόγω σύμβασης.



Σχήμα 8: Εφαρμογή του αλγορίθμου DFS σε μη κατευθυνόμενο γράφο



Σχήμα 9: Ανάπτυξη του δέντρου αναζήτησης του αλγορίθμου DFS του μη κατευθυνόμενου γράφου του Σχήματος 8

Αρχικά ξεκινάμε από τη κορυφή 0. Από τη 0 μπορούμε να επισκεφτούμε τις κορυφές 1, 4, 5 και 9. Έχουμε αποφασίσει ότι αν υπάρχουν πολλές γειτονικές κορυφές να επιλέγουμε αυτή με το μικρότερο όνομα, οπότε θα επιλέξουμε την 1. Από την 1 μπορούμε να πάμε στη 2 ή στη 3 και για τον ίδιο λόγο που μόλις αναφέραμε θα επιλέξουμε τη 2. Συνεχίζοντας την ίδια διαδικασία μεταβαίνουμε διαδοχικά στις κορυφές 3, 7 και 4. Φτάνοντας στη 4 μπορούμε να μεταβούμε μόνο στη 5 αφού την 0 την έχουμε ήδη εξερευνήσει. Με τον ίδιο τρόπο από τη 5 πάμε στην 6 και μετά στην 8. Φτάνοντας στην 8 παρατηρούμε ότι δεν υπάρχουν άλλες ανεξερεύνητες γειτονικές κορυφές. Σε αυτή τη περίπτωση ακολουθούμε πορεία προς τα πίσω πηγαίνοντας από την 6 στην 5, μετά στη 4, στην 7 και καταλήγουμε στη 3. Από τη 3 μπορούμε να πάμε στη 2 και στη 10. Επειδή όμως τη 2 την έχουμε εξερευνήσει νωρίτερα, θα επισκεφτούμε τη 10. Εν συνεχεία από τη 10 πάμε στη 2, από τη 2 στην 1 και μετά στη 0. Από την 0 μπορούμε να πάμε στις κορυφές 4, 5 και 9. Οι 4 και 5 είναι ήδη εξερευνημένες. Η 9 δεν είναι και συνεπώς την επισκεπτόμαστε και η αναζήτηση ολοκληρώνεται.

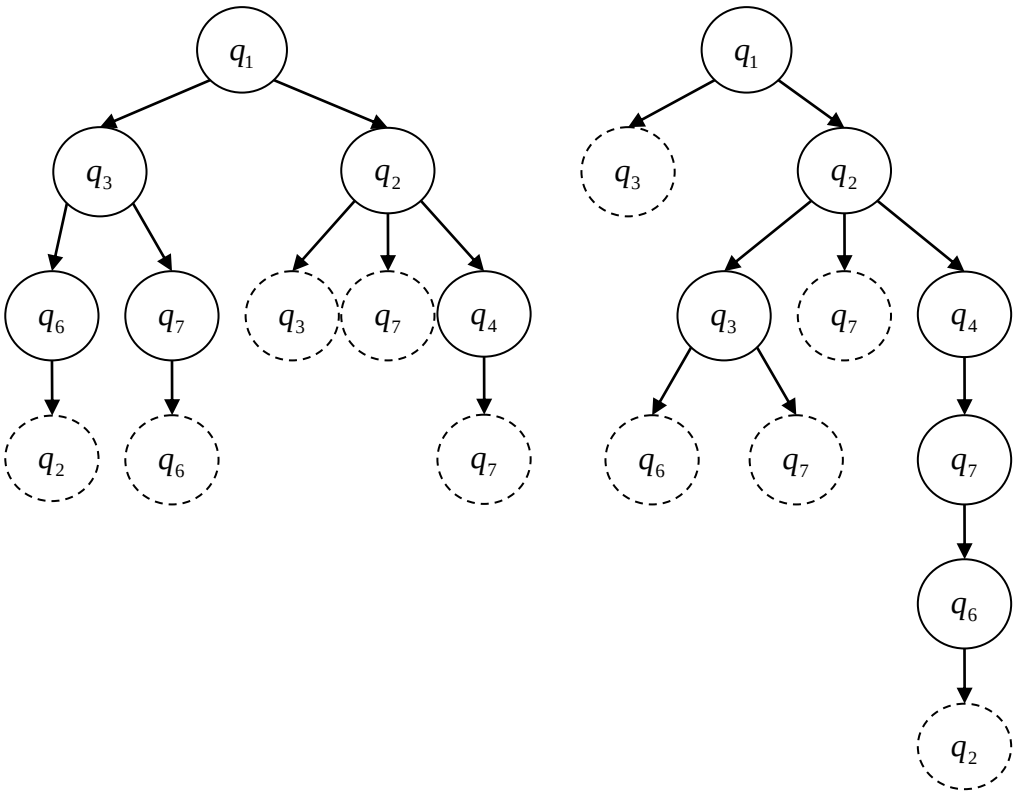
Μέχρι τώρα δείξαμε τον τρόπο που εφαρμόζονται οι αλγόριθμοι BFS και DFS σε μη κατευθυνόμενους γράφους. Στη συνέχεια θα παρουσιάσουμε την εφαρμογή των δύο αυτών αλγορίθμων στη περίπτωση που έχουμε κατευθυνόμενους γράφους.

Παράδειγμα 2ο :

Έστω ότι έχουμε το αυτόματο όπως αυτό απεικονίζεται στο σχήμα που ακολουθεί και δύο προβλήματα προσπελασιμότητας:

- P_1: Είναι η q_5 προσπελάσιμη από τη q_1;
- P_2: Είναι η q_6 προσπελάσιμη από τη q_1;

Για το πρόβλημα P_1 η απάντηση είναι αρνητική διότι και οι δύο αλγόριθμοι τερματίζουν τη αναζήτησή τους υπολογίζοντας όλες τις καταστάσεις που μπορούν να προσπελαστούν από τη q_1 χωρίς όμως να προσπελάσουν τη q_5 . Στο Σχήμα 10 βλέπουμε τη συμπεριφορά των δύο αλγορίθμων υπό την εξής σειρά εξερεύνησης των ακολούθων: $q_3 \prec q_2$ για $Succ(q_1)$ (από τις ακόλουθες καταστάσεις της q_1 η q_3 θα έχει προτεραιότητα της q_2) και $q_3 \prec q_7 \prec q_4$ για $Succ(q_2)$ (από τις ακόλουθες καταστάσεις της q_2 η q_3 θα έχει προτεραιότητα της q_1 και η q_1 θα έχει προτεραιότητα της q_4). Οι διακεκομμένες γραμμές στους κόμβους του Σχήματος δείχνουν το σημείο στο οποίο η αναζήτηση σταματάει, δηλαδή στο κόμβο εκείνο του οποίου οι ακόλουθοι δε θα εξερευνηθούν διότι έχουν ήδη εξερευνηθεί σε άλλες διαδρομές.

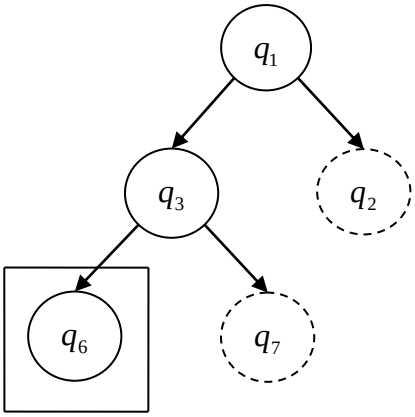


Αναζήτηση Προτεραιότητας Εύρους BFS

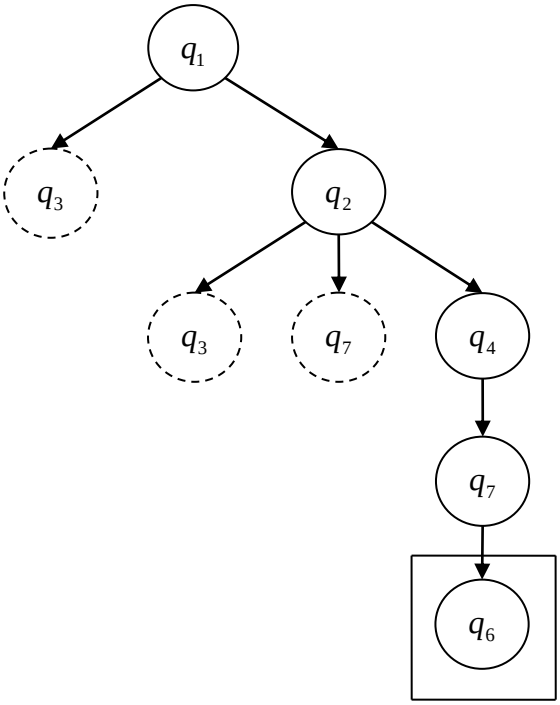
Αναζήτηση Προτεραιότητας Βάθους DFS

Σχήμα 10: Η αναζήτηση των δέντρων του προβλήματος P_1 με τους αλγορίθμους BFS και DFS

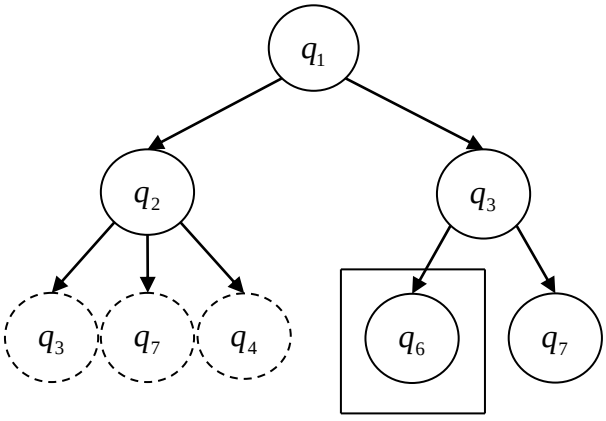
Στο Σχήμα 11 παρουσιάζεται η αναζήτηση των δέντρων για το πρόβλημα P_2 ακολουθώντας πάντα την ίδια σειρά διαδοχής. Παρατηρούμε ότι ο BFS βρίσκει τη συντομότερη διαδρομή ενώ ο DFS βρίσκει τη μακρύτερη ($q_1 \to q_2 \to q_4 \to q_7 \to q_6$). Στο Σχήμα 11 επίσης παρουσιάζεται η ευαισθησία που επιδεικνύει ο DFS στις αλλαγές της σειράς των ακολούθων. Συγκεκριμένα φαίνεται η συμπεριφορά του αλγορίθμου όταν οι ακόλουθοι του q_1 ταξινομούνται σύμφωνα με τη σχέση $q_2 \prec q_3$. Στη περίπτωση αυτή ο BFS βρίσκει την ίδια συντομότερη διαδρομή όπως και προηγουμένως, ενώ τώρα πια και ο DFS βρίσκει την ίδια διαδρομή.



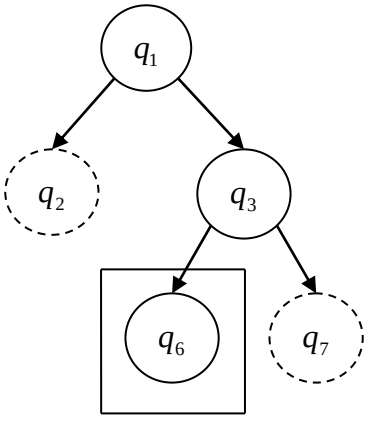
Αναζήτηση Προτεραιότητας Εύρους (BFS)



Αναζήτηση Προτεραιότητας Βάθους (DFS)



Αναζήτηση Προτεραιότητας Εύρους (BFS)



Αναζήτηση Προτεραιότητας Βάθους (DFS)

Σχήμα 11: Τα δέντρα αναζήτησης για το πρόβλημα P_2 με σχέση προτεραιότητας ακολούθων $q_3 \prec q_2$ ***(επάνω) και*** $q_2 \prec q_3$ ***(κάτω)***

Σε συγκεκριμένα πεδία εφαρμογών ενδεχομένως να υπάρχει κάποια χρήσιμή συνάρτηση η οποία να χρησιμοποιείται για μια πιο καλή μεθοδολογία αναζήτησης. Για παράδειγμα αν ο χώρος των καταστάσεων περιλαμβάνει ν –άδες της άλγεβρας Boole και οι δυναμικές ορίζονται με ένα συγκεκριμένο τρόπο, η απόσταση Hamming μεταξύ μιας κατάστασης και της q' θα μας δώσει μια ένδειξη για το πόσο κοντά βρισκόμαστε στο στόχο μας. Ένας αλγόριθμος ο οποίος διατηρεί τη λίστα αναμονής ταξινομημένη σύμφωνα με ένα τέτοιο μέτρο είναι ο ***αλγόριθμος αναζήτησης προτεραιότητας βέλτιστου*** **(*best–first search algorithm*)** ο οποίος θα αξιοποιηθεί, κατάλληλα προσαρμοζόμενος, σε σημαντικό βαθμό στην παρούσα εργασία.

Οι αλγόριθμοι που αναφέρθηκαν πρωτύτερα είχαν το χαρακτηριστικό ότι εξερευνούσαν το αυτόματο σε ***ευθεία* (*forward*)** δηλαδή από τη q στη q'. Η ίδια διαδικασία της εξερεύνησης θα μπορούσε να πραγματοποιηθεί ***ανάστροφα*** (***backward***) δηλαδή ξεκινώντας από τη κατάσταση q' και υπολογίζοντας όλες τις προηγούμενες μέχρι να προσπελάσουμε τη q. Ένας τρόπος για να προχωρήσουμε σε ανάστροφη αναζήτηση σε ένα αυτόματο $A=(Q,\delta)$ είναι να κατασκευάσουμε ένα αυτόματο $A'=(Q,\delta')$ όπου δ' είναι το αντίστροφο της σχέσης μετάβασης δ. Επιλύνοντας το πρόβλημα προσπέλασης από τη q' στη q ($q' \to q$) στο αυτόματο A' είναι σα να επιλύουμε το πρόβλημα προσπέλασης από τη q στη q' ($q \to q'$) στο αυτόματο A αλλά μέσω ανάστροφης αναζήτησης.

Ένα άλλο σημαντικό στοιχείο είναι ότι οι γράφοι μεταβάσεων στο αυτόματο είναι μεγάλου μεγέθους και συνεπώς δεν είναι δυνατών να αποθηκευτούν στη μνήμη του υπολογιστή, με αποτέλεσμα να δημιουργούνται ***εν κινήσει* (*on the fly*)** κατά τη διάρκεια της αναζήτησης.

Οι ακόλουθοι σε ένα συνολικό αυτόματο δημιουργούνται επιλέγοντας κάθε φορά μια μετάβαση από ένα από τα στοιχεία και μετασχηματίζοντας τη συνολική κατάσταση σύμφωνα με την επίδραση που έχει αυτή η μετάβαση.

Τα μοντέλα τα οποία βασίζονται σε αυτόματα πεπερασμένων καταστάσεων (finite-state automata) μπορούν να εκφράσουν μόνο ***ποιοτικές*** χρονικές σχέσεις μεταξύ των γεγονότων σε ένα σύστημα. Έτσι π.χ. μπορούμε να πούμε ότι ένα γεγονός προηγείται κάποιου άλλου, αλλά δεν μπορούμε να προσδιορίσουμε με φυσικό τρόπο τη ποσότητα του χρόνου που χωρίζει αυτά τα δυο γεγονότα. Μάλιστα σε πολλά πεδία εφαρμογών, όπως για παράδειγμα σε προγραμματισμό πραγματικού χρόνου (real-time programming) ή στη ανάλυση χρονισμού κυκλωμάτων (circuit timing analysis), η

ακρίβεια αυτών των συστημάτων εξαρτάται από τις σχετικές ταχύτητες τους και το περιβάλλον τους. Έτσι χρειαζόμαστε μια φόρμα όπου θα μας δίνεται η δυνατότητα να εκφράσουμε ***ποσοτικά*** στοιχεία του χρόνου των συστημάτων, όπως για παράδειγμα τους χρόνους απόκρισης των προγραμμάτων (response time of programs), καθυστερήσεις μετάδοσης σε λογικές πύλες (propagation delays in local gates) κ.α.. Αναφορικά με τα προβλήματα προγραμματισμού παραγωγής μέσω, αυτής της φόρμας θα μπορούσαμε να εκφράσουμε τη διάρκεια ενός βήματος (step) μιας εργασίας λαμβάνοντας ως περιορισμό το χρόνο που περνάει μεταξύ των μεταβάσεων (transitions) αρχής και τέλους του εν λόγω βήματος.

Στα τέλη της δεκαετίας του ’80 έγιναν πολλές προσπάθειες σχετικά με νέες μεθοδολογίες που σκοπό είχαν να μοντελοποιήσουν ποσοτικά το χρόνο. Ένα τέτοιο μοντέλο ήταν το χρονισμένο αυτόματο (timed automaton) των Alur και Dill το οποίο μάλιστα πολύ σύντομα έγινε δημοφιλές και αποδείχθηκε ιδιαίτερα χρήσιμο [13]. Το συγκεκριμένο μοντέλο είναι αρκετά ικανό στο να μπορεί να εκφράζει όλα τα προβλήματα πρακτικού ενδιαφέροντος που εμπεριέχουν την έννοια του χρόνου (timing problems).

Τα χρονισμένα αυτόματα αποτελούν ουσιαστικά μια ειδική κατηγορία υβριδικών συστημάτων, δηλαδή συστημάτων τα οποία συνδυάζουν διακριτές μεταβάσεις και συνεχή εξέλιξη. Στη συνέχεια παρουσιάζουμε το μοντέλο χρονισμένων αυτομάτων σταδιακά, ερευνώντας αρχικά προσεγγίσεις που έχουν να κάνουν με την πρόσθεση της έννοιας του χρόνου - ποσοτικά πλέον - στα αυτόματα πεπερασμένων καταστάσεων.

4.3. Μοντελοποίηση του χρόνου

Θεωρούμε δυο διαδικασίες P_1 και P_2 με χρόνους 3 και 2 χρονικές μονάδες για την κάθε μια αντίστοιχα. Κάθε διαδικασία μπορεί:

i. να βρίσκεται σε μια κατάσταση αναμονής, δηλαδή να περιμένει πριν ξεκινήσει να εκτελείται,

ii. να βρίσκεται σε μια ενεργή κατάσταση, δηλαδή να εκτελείται, ή

iii. να έχει ολοκληρωθεί, εφόσον έχει περάσει αρκετή ώρα που ήταν σε ενεργή κατάσταση

Η κατάσταση κάθε διαδικασίας, όταν αυτή είναι ενεργή, προσδιορίζεται από τη ποσότητα του χρόνου από τη στιγμή που η διαδικασία έχει ξεκινήσει. Συνεπώς αν θεωρήσουμε ότι το πέρασμα του χρόνου πραγματοποιείται κατά μια χρονική μονάδα κάθε φορά, μπορούμε να μοντελοποιήσουμε τις διαδικασίες P_1 και P_2 με χρόνους εκτέλεσης 3 και 2 χρονικές μονάδες αντίστοιχα, χρησιμοποιώντας τα αυτόματα A_1 και A_2 όπως παρουσιάζονται στο Σχήμα 12. Με βάση λοιπόν την παραπάνω ανάλυση το σύνολο των καταστάσεων του A_1 είναι $\{\overline{p}_1, 0, 1, 2, 3, \underline{p}_1\}$, όπου $\overline{p}_1$ είναι η αρχική κατάσταση που δείχνει ότι η διαδικασία είναι ακόμα μη ενεργή –δεν έχει ξεκινήσει να εκτελείται-, $\{0,1,2,3\}$ είναι οι καταστάσεις που αναπαριστούν τη ποσότητα του χρόνου που περνάει όταν η διαδικασία είναι πλέον ενεργή, δηλαδή εκτελείται και $\underline{p}_1$ είναι η τελική κατάσταση που δείχνει ότι η διαδικασία έχει πλέον ολοκληρωθεί. Το αυτόματο περιλαμβάνει δυο τύπους μεταβάσεων:

i. **Άμεσες μεταβάσεις (immediate transitions)** όπως “αρχή” και “τέλος”, όπου δεν αναλώνεται χρόνος και

ii. Μεταβάσεις στις οποίες ενυπάρχει το ***πέρασμα του χρόνου (time-passage transitions)*** και χρησιμοποιώντας τον όρο “παλμός” (“tick”), δηλώνουμε κάθε φορά το πέρασμα μιας χρονικής μονάδας.

Οι συμπεριφορές των διαφόρων διαδικασιών αποτελούν τις ***εκτελέσεις (runs)*** του αυτομάτου και περιλαμβάνουν τις ακολουθίες των καταστάσεων καθώς και τους δυο τύπους των μεταβάσεων.

Παράδειγμα: Η συμπεριφορά όταν η διαδικασία P_1 περιμένει μια χρονική μονάδα και εν συνεχεία ξεκινάει να εκτελείται είναι η ακόλουθη εκτέλεση:

$$\overline{p}_1 \xrightarrow{tick} \underline{p}_1 \xrightarrow{start1} 0 \xrightarrow{tick} 1 \xrightarrow{tick} 2 \xrightarrow{tick} 3 \xrightarrow{end1} \underline{p}_1 \qquad (1)$$

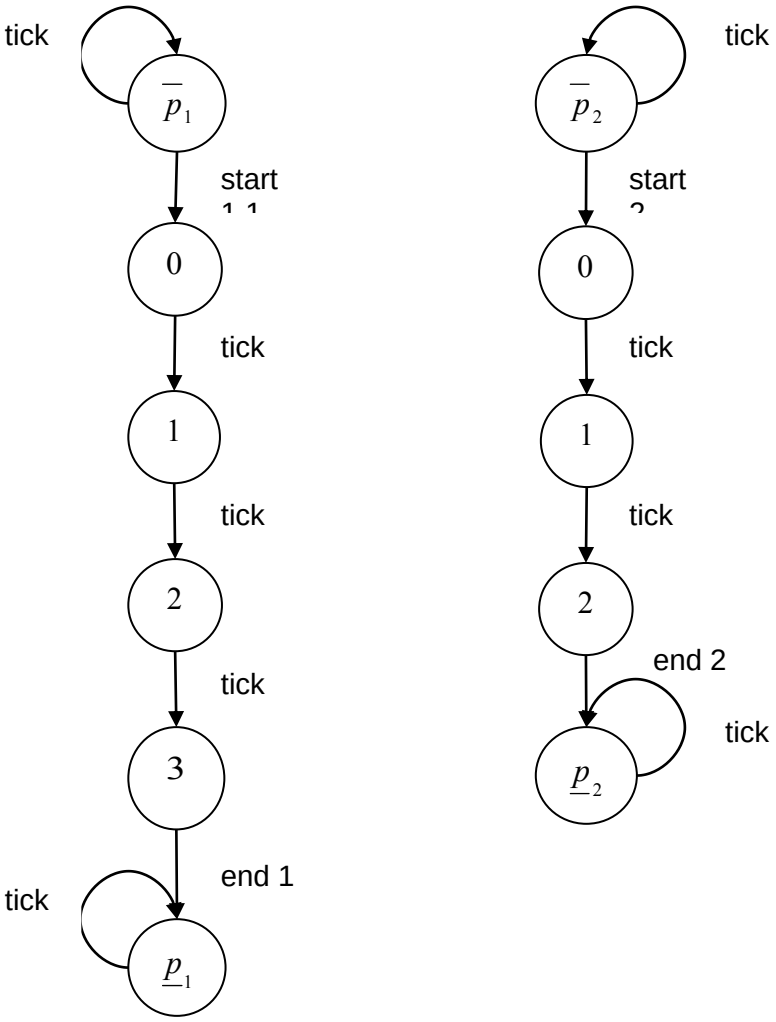
Παράδειγμα: Η συμπεριφορά όταν η P_1 περιμένει δυο χρονικές μονάδες και μετά αρχίζει να εκτελείται είναι:

$$\overline{p}_1 \xrightarrow{tick} \underline{p}_1 \xrightarrow{tick} \underline{p}_1 \xrightarrow{start1} 0 \xrightarrow{tick} 1 \xrightarrow{tick} 2 \xrightarrow{tick} 3 \xrightarrow{end1} \underline{p}_1 \qquad (2)$$

Παράδειγμα: Η συμπεριφορά όταν η P_2 περιμένει δυο χρονικές μονάδες και εν συνεχεία ξεκινάει να εκτελείται είναι η εξής εκέλεση:

$$\overline{p}_2 \xrightarrow{tick} \overline{p}_2 \xrightarrow{tick} \overline{p}_2 \xrightarrow{start2} 0 \xrightarrow{tick} 1 \xrightarrow{tick} 2 \xrightarrow{end2} \underline{p}_2 \qquad (3)$$

Η διάρκεια κάθε μιας διαδικασίας συνολικά είναι στη πραγματικότητα ο αριθμός των μεταβάσεων στις οποίες υπάρχει η έννοια του περάσματος του χρόνου και υπολογίζεται μετρώντας τον αριθμό των “παλμών” (“ticks”). Έτσι, στο πρώτο παράδειγμα η διάρκεια είναι 4 χρονικές μονάδες (4 παλμοί), στο δεύτερο είναι 5 χρονικές μονάδες (5 παλμοί) και τέλος στο τρίτο είναι 4 χρονικές μονάδες (4 παλμοί).



Σχήμα 12: Τα αυτόματα A_1 και A_2 των διαδικασιών P_1 και P_2 αντίστοιχα

Όταν δυο ή περισσότερα αυτόματα “εκτελούνται” παράλληλα, κάθε ένα μπορεί να λάβει τις άμεσου τύπου μεταβάσεις ανεξάρτητα από το άλλο, αλλά οι μεταβάσεις που απαιτούν το πέρασμα του χρόνου, δηλαδή την πραγματοποίηση παλμών, πρέπει να είναι συγχρονισμένες σε όλα. Αυτό σημαίνει ότι κάθε φορά που μια διαδικασία λαμβάνει μια μετάβαση παλμού τότε και όλες οι άλλες πρέπει να τη λάβουν επίσης. Η επίδραση λοιπόν μιας μετάβασης παλμού σε μια διαδικασία που εκτελείται (ευρισκόμενη σε μια κατάσταση i) είναι ουσιαστικά η μετακίνηση της διαδικασίας από την κατάσταση i στην κατάσταση $i+1$.

Στο

Σχήμα 13 το αυτόματο $A = A_1 \parallel A_2$ ξεκινάει από μια αρχική κατάσταση $(\overline{p}_1, \overline{p}_2)$ στην οποία και οι δυο διαδικασίες βρίσκονται σε αναμονή. Κάθε μια από τις διαδικασίες μπορεί να ξεκινήσει να εκτελείται μετά από οποιονδήποτε αριθμό παλμών και όλες οι πιθανές συμπεριφορές του συστήματος αντιστοιχούν στις διάφορες εκτελέσεις του A.

Παράδειγμα: Η συμπεριφορά όταν οι δυο διαδικασίες περιμένουν 2 χρονικές μονάδες και μετά ξεκινούν φαίνεται παρακάτω:

$$(\overline{p}_1, \overline{p}_2) \xrightarrow{tick} (\overline{p}_1, \overline{p}_2) \xrightarrow{tick} (\overline{p}_1, \overline{p}_2) \xrightarrow{start1} (0, \overline{p}_2) \xrightarrow{start2} (0,0) \xrightarrow{tick} (1,1) \xrightarrow{tick} (2,2) \xrightarrow{end2} (2, \underline{p}_2) \xrightarrow{tick} (3, \underline{p}_2) \xrightarrow{end1} (\underline{p}_1, \underline{p}_2)$$

Η διάρκεια της εκτέλεσης είναι 5 χρονικές μονάδες.

Παράδειγμα: Η συμπεριφορά όπου η διαδικασία P_1 ξεκινάει αμέσως ενώ η P_2 ξεκινάει 2 χρονικές μονάδες αργότερα αναπαριστάται από την ακόλουθη εκτέλεση:

$$(\overline{p}_1, \overline{p}_2) \xrightarrow{start1} (0, \overline{p}_2) \xrightarrow{tick} (1, \overline{p}_2) \xrightarrow{tick} (2, \overline{p}_2) \xrightarrow{start2} (2,0) \xrightarrow{tick} (3,1) \xrightarrow{end1} (\underline{p}_1,1) \xrightarrow{tick} (\underline{p}_1,2) \xrightarrow{end2} (\underline{p}_1, \underline{p}_2)$$

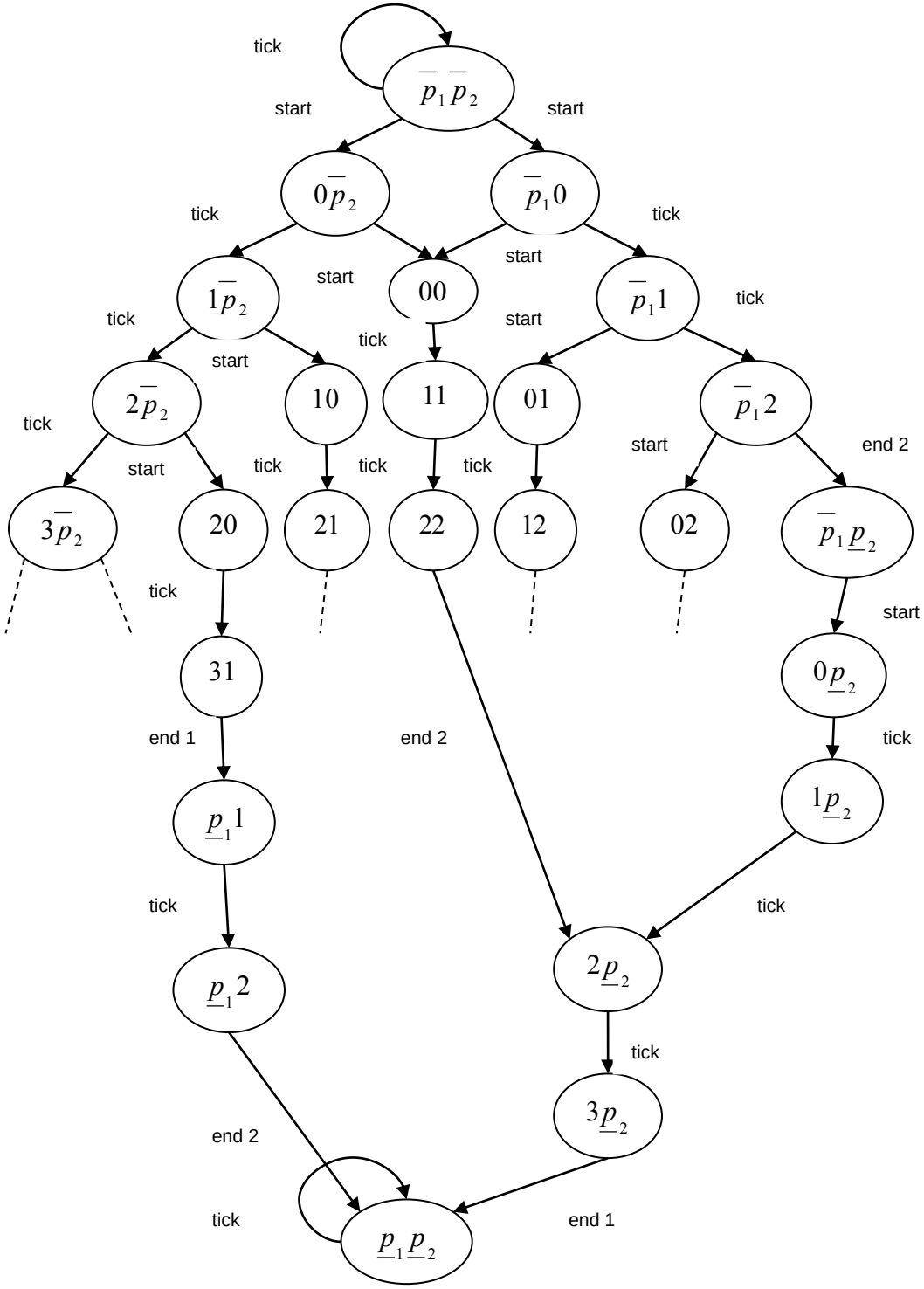
Η διάρκεια της εκτέλεσης είναι 4 χρονικές μονάδες.

Παράδειγμα: Η συμπεριφορά του αυτομάτου όταν η P_2 ξεκινάει αμέσως και η P_1 ξεκινάει μια χρονική μονάδα μετά το τέλος της P_2 είναι:

$$(\overline{p}_1, \overline{p}_2) \xrightarrow{start\,2} (\overline{p}_1, 0) \xrightarrow{tick} (\overline{p}_1, 1) \xrightarrow{tick} (\overline{p}_1, 2) \xrightarrow{end\,2} (\overline{p}_1, \underline{p}_2) \xrightarrow{tick}$$
$$(\overline{p}_1, \underline{p}_2) \xrightarrow{start\,1} (0, \underline{p}_2) \xrightarrow{tick} (1, \underline{p}_2) \xrightarrow{tick} (2, \underline{p}_2) \xrightarrow{tick} (3, \underline{p}_2) \xrightarrow{end\,1}$$
$$(\underline{p}_1, \underline{p}_2)$$

Η διάρκεια της εκτέλεσης είναι 6 παλμοί, δηλαδή 6 χρονικές μονάδες.

Όπως φαίνεται και από τα παραπάνω παραδείγματα δεδομένου ότι κάθε μια από τις διαδικασίες μπορεί να έχει μια μετάβαση αρχής (start transition) οποιαδήποτε στιγμή ανεξάρτητα από την άλλη, είναι δυνατόν να προκύψουν πάρα πολλοί συνδυασμοί τιμών χρονιστών (clock values) που κάθε ένας αντανακλά σε μια διαφορετική επιλογή στο παρελθόν. Αν σε αυτό λάβει κανείς υπόψη του ότι κάλλιστα θα μπορούσε να επιλεχθεί διαφορετική χρονική κλίμακα, το γεγονός αυτό μπορεί να οδηγήσει σε ένα τεράστιο αριθμό καταστάσεων καθώς ο αριθμός των διαδικασιών θα αυξάνεται κάθε φορά. Παρόλα αυτά το κυριότερο πλεονέκτημα που απορρέει από την συγκεκριμένη αναπαράσταση είναι ότι μας επιτρέπει να είμαστε εξοικειωμένοι με το πλαίσιο των αυτομάτων και να μπορούμε να εφαρμόσουμε αλγορίθμους_προσπέλασης για την εύρεση της συντομότερης διαδρομής σε χρονικά συστήματα (timing systems) λαμβάνοντας ως στάθμιση 1 για “μεταβάσεις παλμών” (tick transitions) και 0 για άμεσες μεταβάσεις.



Σχήμα 13: Το συνολικό αυτόματο $A = A_1 \parallel A_2$

4.4. Μεταβλητές χρονιστών/ ρολόγια

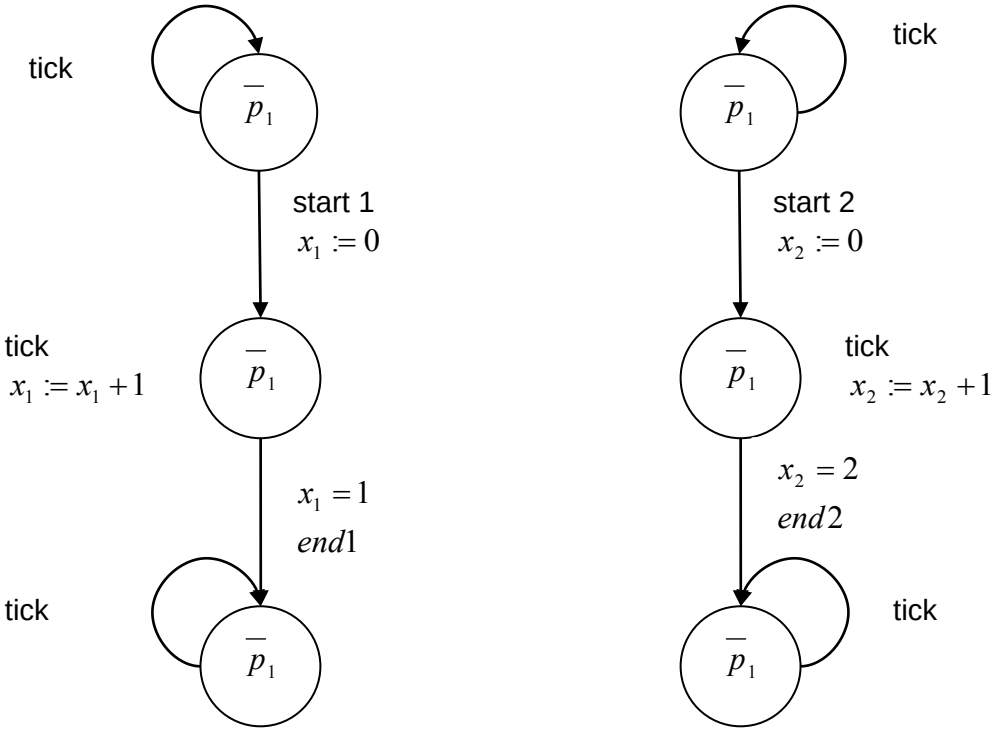
Μια πιο συμβατική και λιγότερο εκτενής αναπαράσταση τέτοιων αυτομάτων μπορεί να επιτευχθεί με τη χρήση ειδικών βοηθητικών μεταβλητών που αναπαριστούν το πέρασμα του χρόνου. Έτσι δε θα χρειάζεται πλέον να κωδικοποιούμε το χρόνο που περνάει και να τον παρουσιάζουμε μέσα σε κάθε κατάσταση ξεχωριστά όπως κάναμε προηγουμένως. Αυτού του είδους οι μεταβλητές λέγονται μεταβλητές χρονισμού, ρολόγια ή μετρητές και μηδενίζονται όταν μια διαδικασία εισέρχεται σε μια ενεργή κατάσταση και εν συνεχεία η τιμή τους αυξάνεται με κάθε παλμό κατά μια μονάδα. Το Σχήμα 14 απεικονίζει τα αυτόματα A_1 και A_2 του Σχήματος 12, χρησιμοποιώντας τις μεταβλητές χρονισμού X_1 και X_2. Στη πραγματικότητα μια κατάσταση σε ένα αυτόματο είναι ένα ζεύγος της μορφής (*q*,*v*) όπου *q* είναι μια συγκεκριμένη κατάσταση και *v* είναι η τιμή της μεταβλητής. Οι μεταβλητές χρονισμού μπορούν να λάβουν μόνο θετικές ακέραιες τιμές, ενώ το σύμβολο $\perp$ υποδηλώνει ότι σε μια συγκεκριμένη κατάσταση ο χρονιστής (clock) δεν έχει ακόμα ενεργοποιηθεί.

Παράδειγμα: Η εκτέλεση του αυτομάτου A_1 του παραδείγματος 1 με την νέα αναπαράσταση θα είναι A_1':

$$(\overline{p}_1,\perp) \xrightarrow{tick} (\overline{p}_1,\perp) \xrightarrow{start1} (p_1,0) \xrightarrow{tick} (p_1,1) \xrightarrow{tick} (p_1,2) \xrightarrow{tick} (p_1,3) \xrightarrow{end1} (\underline{p}_1,\perp)$$

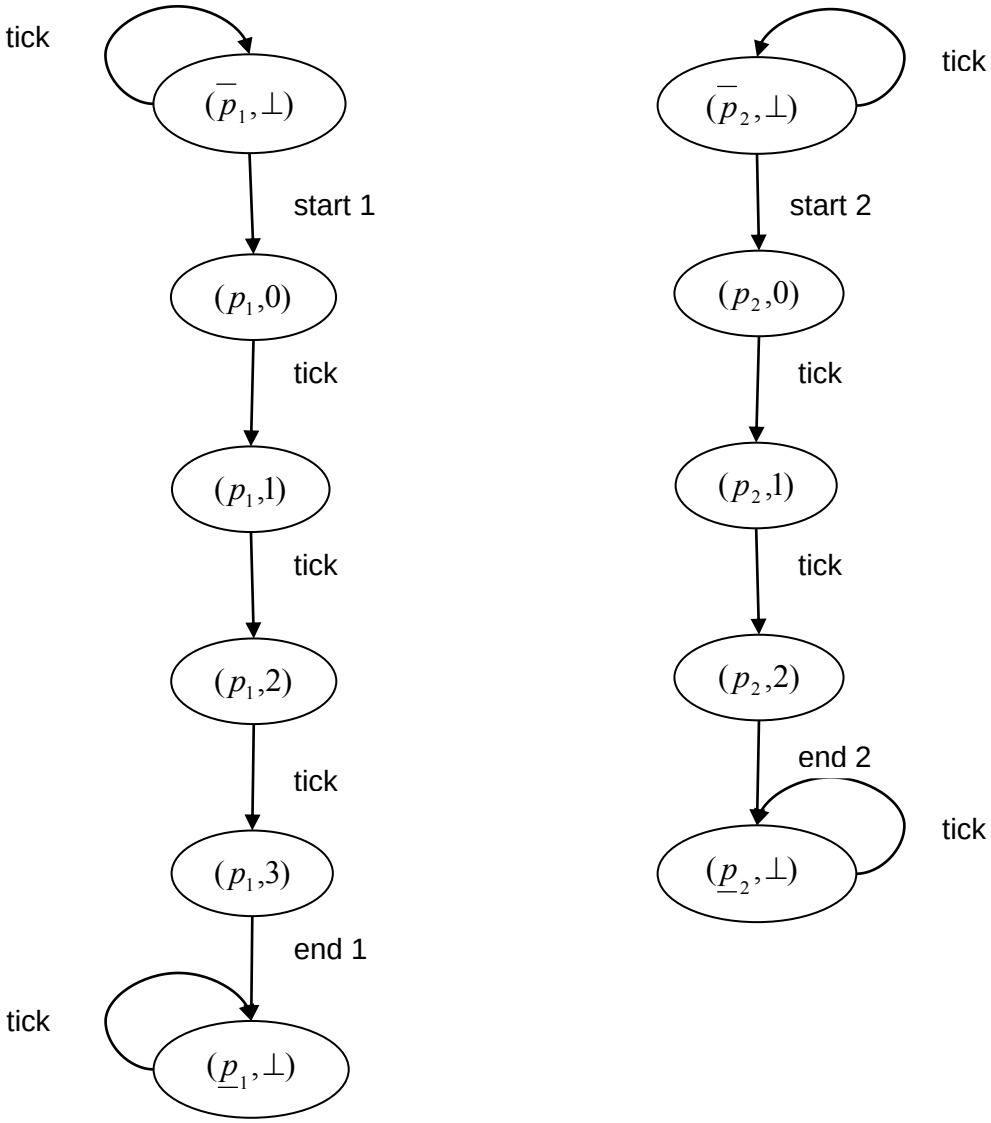
Παράδειγμα: Η εκτέλεση του αυτομάτου A_2 του παραδείγματος 2 είναι A_2':

$$(\overline{p}_2,\perp) \xrightarrow{tick} (\overline{p}_2,\perp) \xrightarrow{start2} (p_{2,}0) \xrightarrow{tick} (p_2,1) \xrightarrow{tick} (p_2,2) \xrightarrow{end2} (\underline{p}_2,\perp)$$



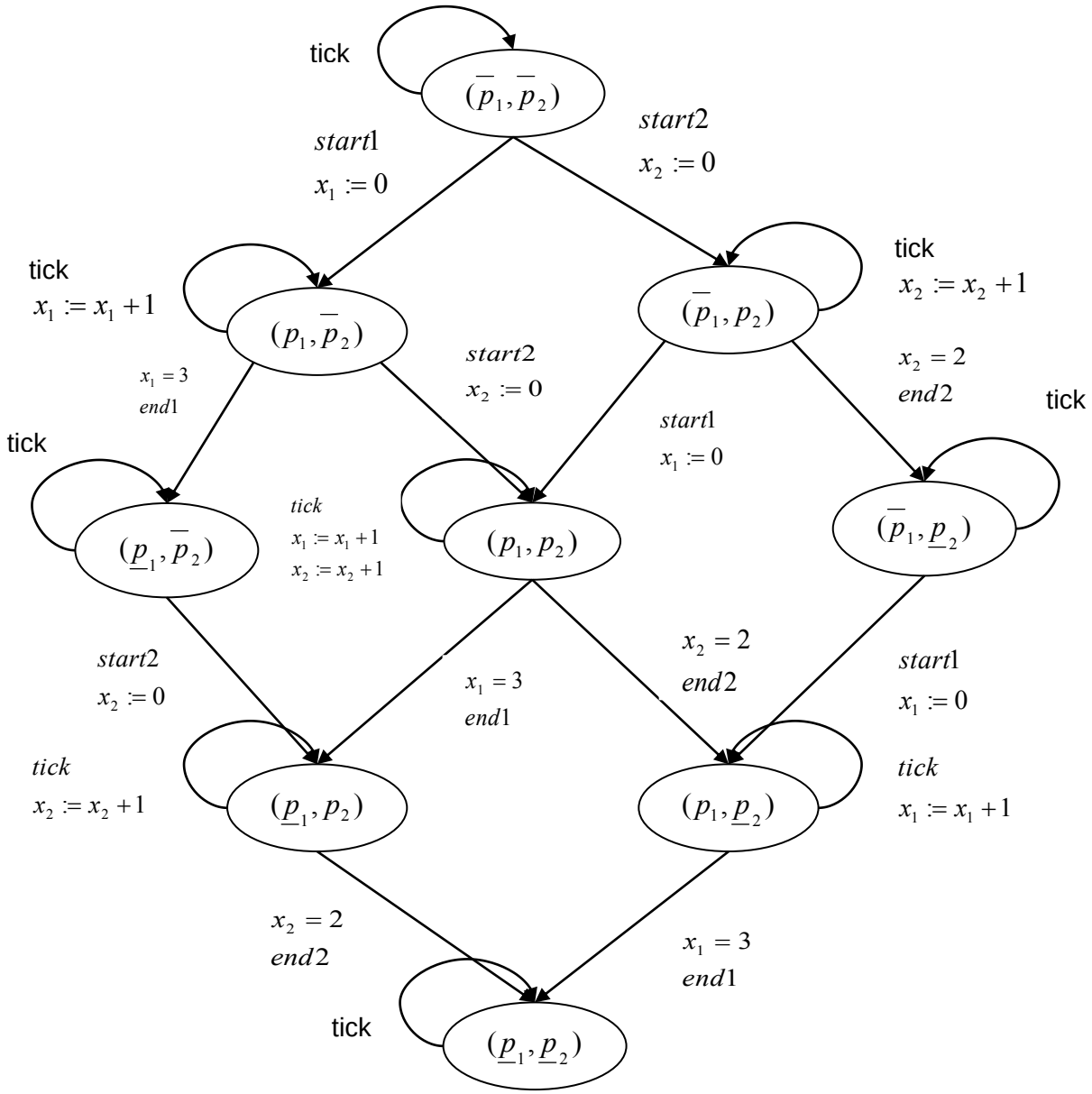
Σχήμα 14: Τα αυτόματα A_1' και A_2' με τη χρήση μεταβλητών χρονισμού

Αξίζει να σημειώσουμε ότι η διαφορά μεταξύ των δύο τρόπων στην αναπαράσταση έχει απλά συνταχτικό χαρακτήρα. Αν επεκτείνουμε τα αυτόματα A_1' και A_2' προσθέτοντας τις τιμές των χρονιστών μέσα στις καταστάσεις θα λάβουμε τα ισομορφικά αυτόματα A_1 και A_2 που απεικονίζονται στο Σχήμα 15.



Σχήμα 15: Τα αυτόματα A_1' ***και*** A_2' ***χρησιμοποιώντας τις τιμές των χρονιστών μέσα στις καταστάσεις***

Αν συνθέσουμε τα A_1' και A_2' τότε θα πάρουμε το ολικό αυτόματο A' που απεικονίζεται στο Σχήμα 16. Είναι εμφανές ότι με το νέο συμβολισμό το αυτόματο A' δείχνει λιγότερο πολύπλοκο σε σχέση με το αυτόματο A του Σχήματος 13



Σχήμα 16: Το ολικό αυτόματο $A' = A'_1 \parallel A'_2$

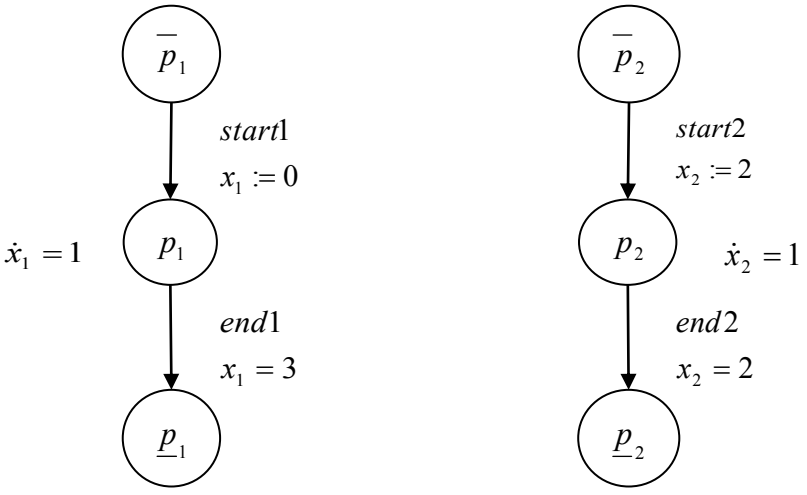
Στη πραγματικότητα η απλοϊκότητα στη παρουσίαση του A' είναι καθαρά φαινομενική. Ας εξετάσουμε για παράδειγμα την κατάσταση (p_1, p_2) όπου οι δυο διαδικασίες είναι ενεργές, δηλαδή εκτελούνται. Φεύγοντας από αυτή την κατάσταση υπάρχουν δυο πιθανές μεταβάσεις. Οι μεταβάσεις αυτές είναι προς τις καταστάσεις $(\underline{p}_1, p_2)$ και $(p_1, \underline{p}_2)$ που "ελέγχονται" από τους χρονιστές $X_1 = 3$ και $X_2 = 2$ αντίστοιχα. Η κατάσταση από μόνη της δε μας λέει το πότε λαμβάνεται κάθε μια από τις μεταβάσεις καθώς αυτό εξαρτάται από τις τιμές των χρονιστών, το οποίο με τη σειρά του εξαρτάται από το παρελθόν (πότε δηλαδή οι χρονιστές μηδενίζονται). Επιπλέον η

εφαρμογή ενός αλγορίθμου προσπέλασης για το A' μπορεί να απαιτεί την επέκτασή του σε A. Παρόλα αυτά η νέα αυτή μορφή μας επιτρέπει να χρησιμοποιούμε συμβολικές μεθόδους και ταυτόχρονα να εργαζόμαστε με αυθαίρετες μονάδες χρόνου.

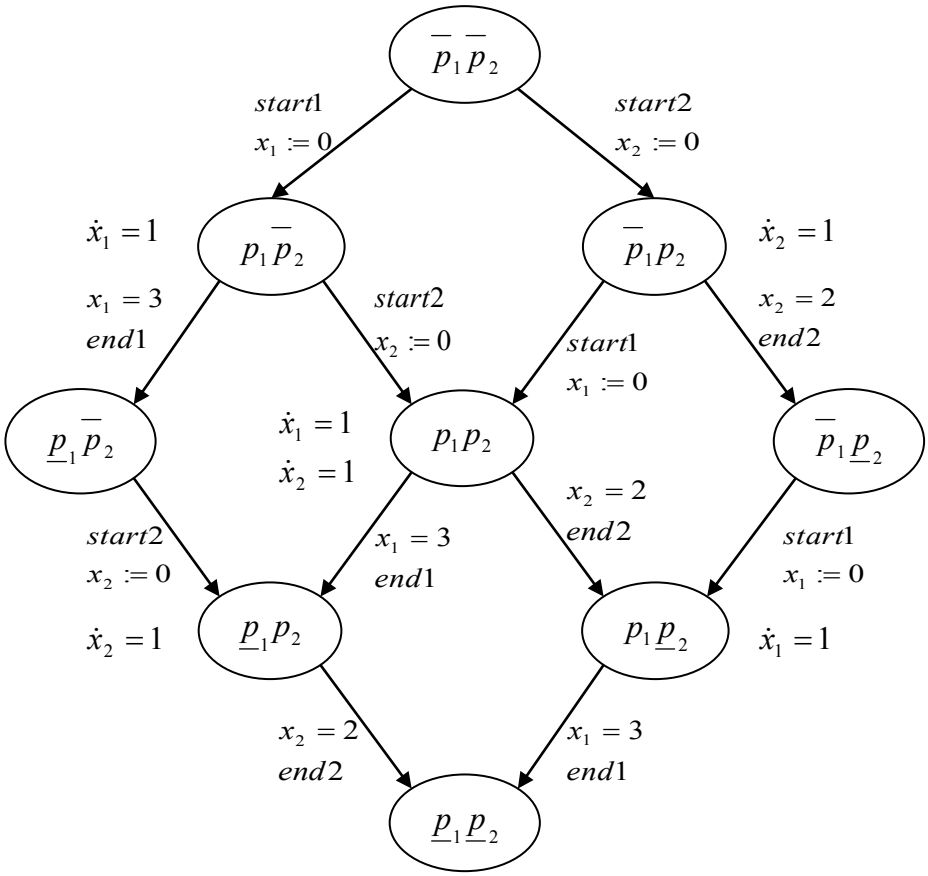
Πολλές φορές ένα σύνολο καταστάσεων μπορεί, αντί των προηγούμενων μεθόδων, να παρασταθεί χρησιμοποιώντας ένα τύπο. Έτσι για παράδειγμα, αν έχουμε δυο διαδικασίες με διάρκειες εκτέλεσης d_1 και d_2 χρονικές μονάδες αντίστοιχα όπου $d_1 < d_2$ και υποθέσουμε ότι η είσοδος της δεύτερης σε ενεργή κατάσταση γίνεται με δυο χρονικές μονάδες διαφορά από τη πρώτη, τότε θα έχουμε ένα σύνολο από τιμές χρονιστών που θα έχουν την ακόλουθη μορφή:

$$\{(2,0),(3,1),(4,2),(5,3),\ldots,(d_1,d_1-2)\}$$

Είναι φανερό ότι στη περίπτωση αυτή το σύνολο των τιμών των χρονιστών εξαρτάται από το d_1. Παρόλα αυτά, το παραπάνω σύνολο θα μπορούσε να χαρακτηριστεί καλύτερα από τον τύπο $X_1 - X_2 = 2 \wedge X_1 \leq d_1$ όπου δεν υπάρχει πια εξάρτηση από το d_1 και ο οποίος μας βοηθάει να εργαζόμαστε με πυκνό χρόνο (dense time) και όχι με συγκεκριμένες μονάδες χρόνου. Αυτός ο συμβολισμός επιτρέπει να συμβαίνουν τα γεγονότα οποιαδήποτε στιγμή κατά μήκος του άξονα του πραγματικού χρόνου, ενώ παράλληλα οι χρονιστές θεωρούνται ως συνεχείς μεταβλητές που περιέχουν το παράγωγο 1 μέσα στις ενεργές καταστάσεις. Στα παρακάτω σχήματα παρουσιάζονται τα δύο αυτόματα που προαναφέραμε.



Σχήμα 17: Τα δυο χρονισμένα αυτόματα



Σχήμα 18: Το ολικό αυτόματο A'

4.5. 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.

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

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

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.

Σε ένα χρονισμένο 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:

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

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* (ή *glc)* 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 $(p_1, \overline{p}_2)$. Στην περίπτωση αυτή ο χρονιστής χαρακτηρίζεται ως **μη ενεργός (inactive)** και διακρίνεται με το σύμβολο $\perp$.

Παράδειγμα: Παρακάτω παρουσιάζεται η εκτέλεση του αυτομάτου A'' του Σχήματος 18 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:

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

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

$$(p_1, \overline{p}_2, 0, \perp) \xrightarrow{2,5} (p_1, \overline{p}_2, 2.5, \perp)$$

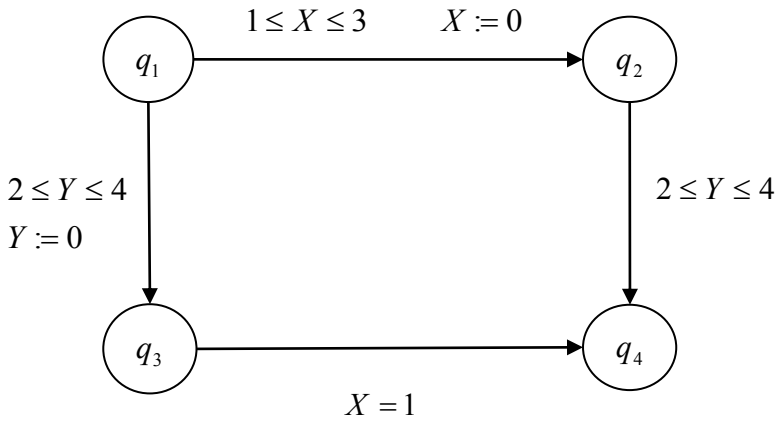
μπορεί να γραφτεί ως

$$(p_1, \overline{p}_2, 0, \perp) \xrightarrow{0,9} (p_1, \overline{p}_2, 0.9, \perp) \xrightarrow{0,6} (p_1, \overline{p}_2, 1.5, \perp) \xrightarrow{1,0} (p_1, \overline{p}_2, 2.5, \perp)$$

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 19 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 19: Ένα χρονισμένο αυτόματο

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

- 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.
- 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 και διασταυρώνεται με το βοηθό κάθε μιας μετάβασης με σκοπό να προσδιορίσουμε το σχηματισμό στον οποίο η μετάβαση θα πραγματοποιηθεί, ενώ στη συνεχεία εφαρμόζονται οι λειτουργίες επαναφοράς στους σχηματισμούς.

Ας εξετάσουμε το αυτόματο του Σχήματος 19. Ξεκινώντας από τη συμβολική κατάσταση $(q_1, X = Y = 0)$, και αφήνοντας το χρόνο να περάσει προσεγγίζουμε τη συμβολική κατάσταση $(q_1, X = Y \ge 0)$. Η τομή με το βοηθό της μετάβασης στο q_2 δίνει $(q_1, 1 \le X = Y \le 3)$ και ο μηδενισμός του X μας οδηγεί τελικά στη $(q_2, X = 0 \wedge 1 \le Y \le 3)$ από όπου ξεκινάμε ξανά κοκ. Ολόκληρος ο υπολογισμός του αυτομάτου απεικονίζεται στο Σχήμα 20. 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.

Μία 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. Έτσι προκύπτει ένα σύνολο από σχηματισμούς $\{(q, Z) : z \in Z\}$.

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:

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

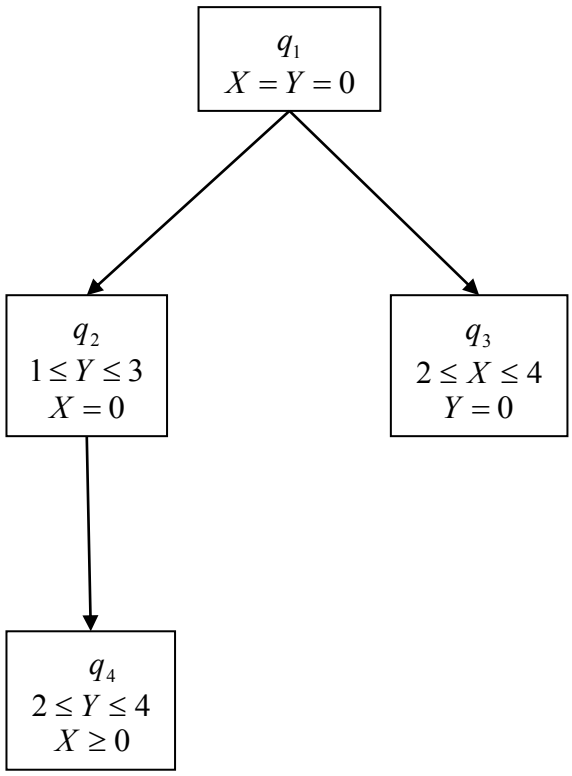
- Οι ακόλουθες καταστάσεις της συμβολικής κατάστασης (q, Z) είναι το σύνολο όλων των δ-ακολούθων:

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

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 παραγωγικών συστημάτων είναι μη κυκλικά ή ακυκλικά (acyclic). Παρακάτω παρουσιάζεται ένας αλγόριθμος ο οποίος υπολογίζει όλους τους προσπελάσιμους σχηματισμούς σε ένα μη κυκλικό αυτόματο ξεκινώντας από το σχηματισμό $(s,0)$.

Αλγόριθμος Ευθείας Προσπέλασης σε μη κυκλικά 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 20: Υπολογισμός ευθείας προσπέλασης για το χρονισμένο αυτόματο του Σχήματος 19

Από τη στιγμή που το αυτόματο είναι μη κυκλικό 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.

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.

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

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

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

- 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 $\delta = (q', \phi, p, q) \in \Delta$.

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

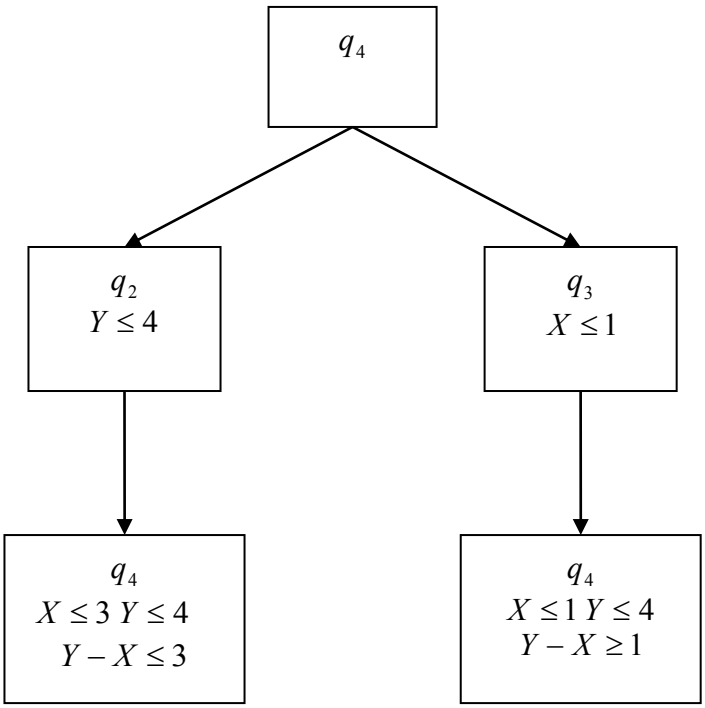
- Οι προηγούμενες καταστάσεις της συμβολικής κατάστασης (q,Z) είναι ένα σύνολο που περιλαμβάνει όλους τους σχηματισμούς από τους οποίους μπορεί κανείς να προσπελάσει τη (q,Z) λαμβάνοντας οποιαδήποτε μετάβαση δ που πραγματώνεται με το πέρασμα του χρόνου.

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

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

Αλγόριθμος Ανάστροφης Προσπέλασης σε μη κυκλικά Χρονισμένα Αυτόματα

$Waiting := \{(f,H)\}$;
while $Waiting \neq \varnothing$ **do**
  Pick $(q,Z) \in Waiting$;
  For every $(q',Z') \in \Pr e(q,Z)$;
    Insert (q',Z') *into Waiting;*
  Remove (q,Z) *from Waiting;*
end



Σχήμα 21: Υπολογισμός ανάστροφης προσπέλασης για το αυτόματο του Σχήματος 19

4.6. Αυτόματα και χρονικός προγραμματισμός εργασιών

Στο πλαίσιο του χρονοπρογραμματισμού παραγωγής, η μοντελοποίηση σε αυτόματα αναλύεται σε μοντελοποίηση των εργασιών, των πόρων, των χρονικών περιορισμών και των λοιπών στοιχείων του προβλήματος ως ανεξάρτητα χρονισμένα αυτόματα, με αναπαράσταση των σχέσεων ανάμεσα σε αυτά μέσω ετικετών συγχρονισμού. Τα ανεξάρτητα αυτά αυτόματα μπορούν στη συνέχεια να συντεθούν ώστε να αναπαραστήσουν ολόκληρο το μοντέλο του προβλήματος χρονοπρογραμματισμού. Το αυτόματο που θα προκύψει από τη σύνθεση αναπαριστάται από ένα γράφημα προσεγγισιμότητας το οποίο περιλαμβάνει μια αρχική τοποθεσία στην οποία καμία εργασία δεν έχει ξεκινήσει, και τουλάχιστον μια τοποθεσία – στόχο, στην οποία όλες οι εργασίες έχουν τελειώσει. Η ανάλυση προσεγγισιμότητας εκτελείται στο συνδυασμένο αυτόματο ώστε να βρεθεί μια διαδρομή από την αρχική κατάσταση στην τελική η οποία και εκπληρώνει κάποιο κριτήριο βελτιστοποίησης. Έτσι, σε ένα πρόβλημα χρονοπρογραμματισμού εργασιών, κάθε μονοπάτι από την αρχική κατάσταση στην

τελική, αντιπροσωπεύει μια έγκυρη διαδρομή και το μονοπάτι το οποίο εκπληρώνει κάποιο κριτήριο βελτιστοποίησης αντιπροσωπεύει τη βέλτιστη διαδρομή [129].

Η δημιουργική ιδέα της χρησιμοποίησης χρονισμένων αυτομάτων για την επίλυση προβλημάτων χρονοπρογραμματισμού αρχικά προτάθηκε στο [57], όπου ο συγγραφεας A.Fehnker αντιμετωπίζει το πρόβλημα επίλυσης ενός προβλήματος χρονοπρογραμματισμού σε μία ολοκληρωμένη μονάδα χάλυβα μέσω της μοντελοποίησης της διαδικασίας σε χρονισμένα αυτόματα και χρησιμοποιώντας ως αλγόριθμο επαλήθευσης του μοντέλου το εργαλείο Uppaal. Ωστόσο, η προσέγγιση που προτάθηκε ήταν μόνο ποιοτική (π.χ να καθοριστεί ένα εφικτό πρόγραμμα για ένα γνωστό δεδομένο σύνολο παραγγελιών με προθεσμίες) και όχι ποσοτική (π.χ να καθοριστεί το βέλτιστο πρόγραμμα ανάμεσα σε άλλα εφικτά προγράμματα). Στο [58] ο Fehnker επέκτεινε την προσέγγιση που ο ίδιος είχε παρουσιάσει νωρίτερα ώστε να μοντελοποιήσει προβλήματα job shop ως δίκτυα χρονισμένων αυτομάτων και επέκτεινε τον αλγόριθμο προσεγγισιμότητας μέσω της ενσωμάτωσης ευρετικών κανόνων όπως FIFO (First-in-first-out), LIFO (Last-in-first-out). Επίσης επισήμανε το πλεονέκτημα της χρήσης διαδικασιών οριοθέτησης κατά την ανάλυση προσεγγισιμότητας η οποία εγγυάται την εύρεση βέλτιστης λύσης. Στο [123], οι συγγραφείς πρότειναν μια προσέγγιση κατά την οποία είναι δυνατή η σύνθεση προγραμμάτων λειτουργίας για την βέλτιστη παραγωγή μιας παραγωγικής μονάδας χρησιμοποιώντας ένα αλγόριθμο προσεγγισιμότητας.

Ένας συστηματικός τρόπος μοντελοποίησης job shop προβλημάτων με στόχο την ελαχιστοποίηση του makespan χρησιμοποιώντας χρονισμένα αυτόματα παρουσιάστηκε επίσης στο [5]. Οι συγγραφείς πρότειναν αρκετούς αλγορίθμους αναζήτησης από το πεδίο της θεωρίας γραφημάτων ώστε να διερευνήσουν το πεδίο αναζήτησης με σκοπό να υπολογιστεί η συντομότερη διαδρομή πάνω στο γράφημα προσεγγισιμότητας. Ανέφεραν ότι η προτεινόμενη προσέγγιση είναι αποτελεσματική σε σύγκριση με τυχαία δημιουργημένα προγράμματα που προτάθηκαν από την επιστημονική κοινότητα για την επίλυση job shop προβλημάτων. Στο [8] οι συγγραφείς εφάρμοσαν την προσέγγιση των χρονισμένων αυτομάτων προκειμένου να αντιμετωπίσουν ένα πρόβλημα χρονοπρογραμματισμού παράλληλων μηχανών όπου οι εργασίες υπόκεινται σε συγκεκριμένους περιορισμούς.

Η επέκταση των χρονισμένων αυτομάτων σε χρονισμένα αυτόματα με βάρη [25], η οποία περιλαμβάνει κόστη μετάβασης καθώς και τιμές κόστους για περιόδους παραμονής σε διάφορες τοποθεσίες, μπορεί να χρησιμοποιηθεί για τον σχηματισμό

περισσότερο πολύπλοκων αντικειμενικών συναρτήσεων και συνεπώς ενθαρρύνει περισσότερο τη χρήση τους σε προβλήματα χρονοπρογραμματισμού. Η ανάλυση προσεγγισιμότητας με σκοπό την βελτιστοποίηση κόστους που υλοποιείται μέσω των χρονισμένων αυτομάτων με βάρη στοχεύει στην εύρεση διαδρομής από την αρχική κατάσταση στην τελική με το βέλτιστο δυνατό κόστος.

Αντίστοιχα με πολλές άλλες προσεγγίσεις που χρησιμοποιούνται σε προβλήματα χρονοπρογραμματισμού, η χρήση χρονισμένων αυτομάτων υποφέρει από το πρόβλημα της ύπαρξης πολλών συνδυασμών και συνεπώς προκειμένου να έχει αυξημένη αποτελεσματικότητα απαιτείται η μείωση του χώρου αναζήτησης. Στο [128] οι συγγραφείς πρότειναν τεχνικές μείωσης του πεδίου αναζήτησης με σκοπό να αυξήσουν την αποτελεσματικότητα της ανάλυσης προσεγγισιμότητας για την επίλυση Job shop προβλημάτων. Εφάρμοσαν την sleep-set μέθοδο με σκοπό να επιλυθούν job shop προβλήματα χρονοπρογραμματισμού και να μειωθούν διαδρομές από το δέντρο αναζήτησης οι οποίες ήταν περιττές και οδηγούσαν στην ίδια λύση. Οι συγγραφείς επισήμαναν την αποτελεσματικότητα των προτεινόμενων τεχνικών μείωσης στην επίλυση προβλημάτων στα οποία στόχος είναι η μείωση του makespan. Τα αποτελέσματα τους εμφανώς οδήγησαν στο συμπέρασμα ότι η αποτελεσματικότητα αυτής της προσέγγισης είναι σαφώς μεγαλύτερη συγκρινόμενη με τις κλασικές τεχνικές της επιχειρησιακής έρευνας. Αυτό ενθάρρυνε ακόμα περισσότερο τη χρήση των χρονισμένων αυτομάτων στην επίλυση πολύπλοκων και μεγάλης κλίμακας προβλημάτων χρονοπρογραμματισμού.

Στο [127] οι συγγραφείς εισήγαγαν την ιδέα υπολογισμού κάτω ορίων για την ελαχιστοποίηση του makespan ενσωματώνοντας γραμμικό προγραμματισμό (LP) στην ανάλυση προσεγγισιμότητας με σκοπό να μειωθεί ο χώρος αναζήτησης. Η τεχνική αυτή βελτίωσε την συνολική απόδοση σε μεγάλο βαθμό, ειδικά για προβλήματα μεγάλης κλίμακας, καθώς με αυτόν τον τρόπο ένα μεγάλο μέρος του δέντρου αναζήτησης εξαλείφθηκε λόγω των κάτω ορίων όπως υπολογίστηκαν από την μέθοδο γραμμικού προγραμματισμού. Ωστόσο, συνολικά η αποτελεσματικότητα της μεθόδου δε βελτιώθηκε και αυτό γιατί η μείωση του δέντρου αναζήτησης αντισταθμίστηκε από την αύξηση του υπολογιστικού χρόνου για την εύρεση λύσης στα προκύπτοντα προβλήματα γραμμικού προγραμματισμού.

Η εφαρμογή μιας προσέγγισης με χρονισμένα αυτομάτων για την επίλυση job shop προβλημάτων με εναλλακτικές διαδρομές και ημερομηνίες λήξης παρουσιάστηκε στο [126]. Ωστόσο, το πρόβλημα επαναδιατυπώθηκε έτσι ώστε οι ημερομηνίες λήξης να

θεωρούνται ως προθεσμίες και ο στόχος είναι να βρεθεί ένα εφικτό πρόγραμμα τέτοιο ώστε όλες οι παραγγελίες να εκτελούνται εντός των προθεσμιών. Σε μια πιο πρόσφατη εργασία του 2011, οι συγγραφείς στο ανέπτυξαν ένα πλαίσιο για την επίλυση προβλημάτων χρονοπρογραμματισμού στη βάση μια αρχιτεκτονικής πολυεπεξεργαστών, όπου οι εργασίες με σχέσεις αλληλεξάρτησης πρέπει να εκτελούνται μέσα σε συγκεκριμένες προθεσμίες [106].

Το χαρακτηριστικό όλων σχεδόν των εφαρμογών και που έχουν παρουσιαστεί στο πεδίο του χρονοπρογραμματισμού μέσω χρονισμένων αυτομάτων είναι ότι θεωρούν ως αντικειμενική συνάρτηση της ελαχιστοποίηση του συνολικού χρόνου υλοποίησης των εργασιών (makespan).

Παράλληλα η πλειονότητα των εργασιών που έχουν δημοσιευθεί επικεντρώνεται στην αντιμετώπιση του κλασικού προβλήματος job shop. Έτσι, δεν έχουν αντιμετωπιστεί περιπτώσεις που οι εργασίες μπορούν να εκτελεστούν μέσω εναλλακτικών διαδρομών ή περιπτώσεις που οι εργασίες να μπορούν να διακοπούν προσωρινά και να ολοκληρωθούν σε μεταγενέστερο χρόνο για χάρη της βελτιστοποίησης του συστήματος.

Τέλος, παρά την εμφάνιση προσεγγίσεων μοντελοποίησης του κόστους εντός των χρονισμένων αυτομάτων με την προσθήκη βαρών, αυτές περιορίζονται στην προσθήκη στοιχείων κόστους στις μεταβάσεις και την εφαρμογή αλγορίθμων συντομότερης διαδρομής. Η μελέτη της βιβλιογραφίας καταδεικνύει ότι απουσιάζουν ολοκληρωμένες προσεγγίσεις αντιμετώπισης σύνθετων παραγωγικών συστημάτων με μοντελοποίηση των διαφορετικών τύπων κόστους που στην πραγματικότητα υπάρχουν. Η παρούσα διατριβή έρχεται να συνεισφέρει στον τομέα αυτό εξετάζοντας σύνθετα συστήματα που επεκτείνουν το μοντέλο του προβλήματος job-shop αλλά και συστήματα στα οποία έμφαση σε επίπεδο βελτιστοποίησης δίνεται στο κόστος.

Κεφάλαιο 5: Εργαλεία μοντελοποίησης διακριτών συστημάτων και αυτομάτων

Στο κεφάλαιο αυτό παρουσιάζονται τα σημαντικότερα λογισμικά εργαλεία που έχουν δημιουργηθεί για την μοντελοποίηση αυτομάτων και, ευρύτερα, συστημάτων διακριτών γεγονότων. Όλα έχουν αναπτυχθεί από ερευνητικά ινστιτούτα και Πανεπιστήμια και διατίθενται ελεύθερα στη διάθεση του κοινού για μη κερδοσκοπική χρήση. Στη συνέχεια παρατίθενται συνοπτικά τα βασικά χαρακτηριστικά των διαθέσιμων εργαλείων μοντελοποίησης, ενώ ιδιαίτερη ανάλυση γίνεται για το εργαλείο Uppaal, το οποίο επιλέχθηκε ως το βασικό εργαλείο για την υλοποίηση των μοντέλων που αναπτύσσονται στην παρούσα διατριβή.

5.1. Ελεύθερα διαθέσιμα λογισμικά

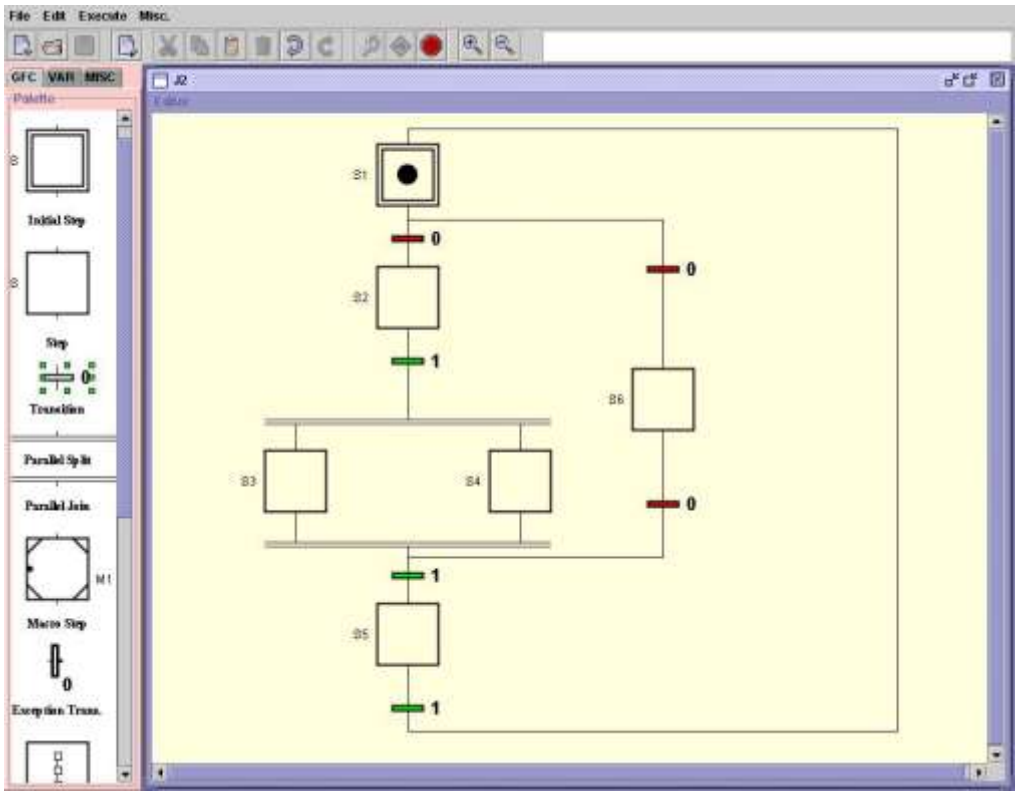
5.1.1. JGrafchart

Η JGrafchart είναι μια υλοποίηση της Grafchart [78], μιας γραφικής γλώσσας προγραμματισμού για εφαρμογές ακολουθιακού ελέγχου. Η όλη υλοποίηση έχει γίνει σε Java και είναι βασισμένη στη Grafcet ή αλλιώς Sequential Function Charts (SFC),

στα δίκτυα Petri, στα διαγράμματα καταστάσεων και σε γλώσσες προγραμματισμού υψηλού επιπέδου. Η JGrafchart απαρτίζεται από δύο μέρη:

- το εργαλείο γραφικής σχεδίασης και
- το σύστημα εκτέλεσης.

Οι δυνατότητες για προσομοίωση και επαλήθευση είναι τα κύρια προτερήματα της JGrafchart, ιδίως για Grafcet-SFC που αποτελούν πρότυπο της IEC 31131-3 ως γλώσσα προγραμματισμού για Προγραμματιζόμενους Λογικούς Ελεγκτές. Στο σχήμα που ακολουθεί απεικονίζεται μια εικόνα του λογισμικού εργαλείου.

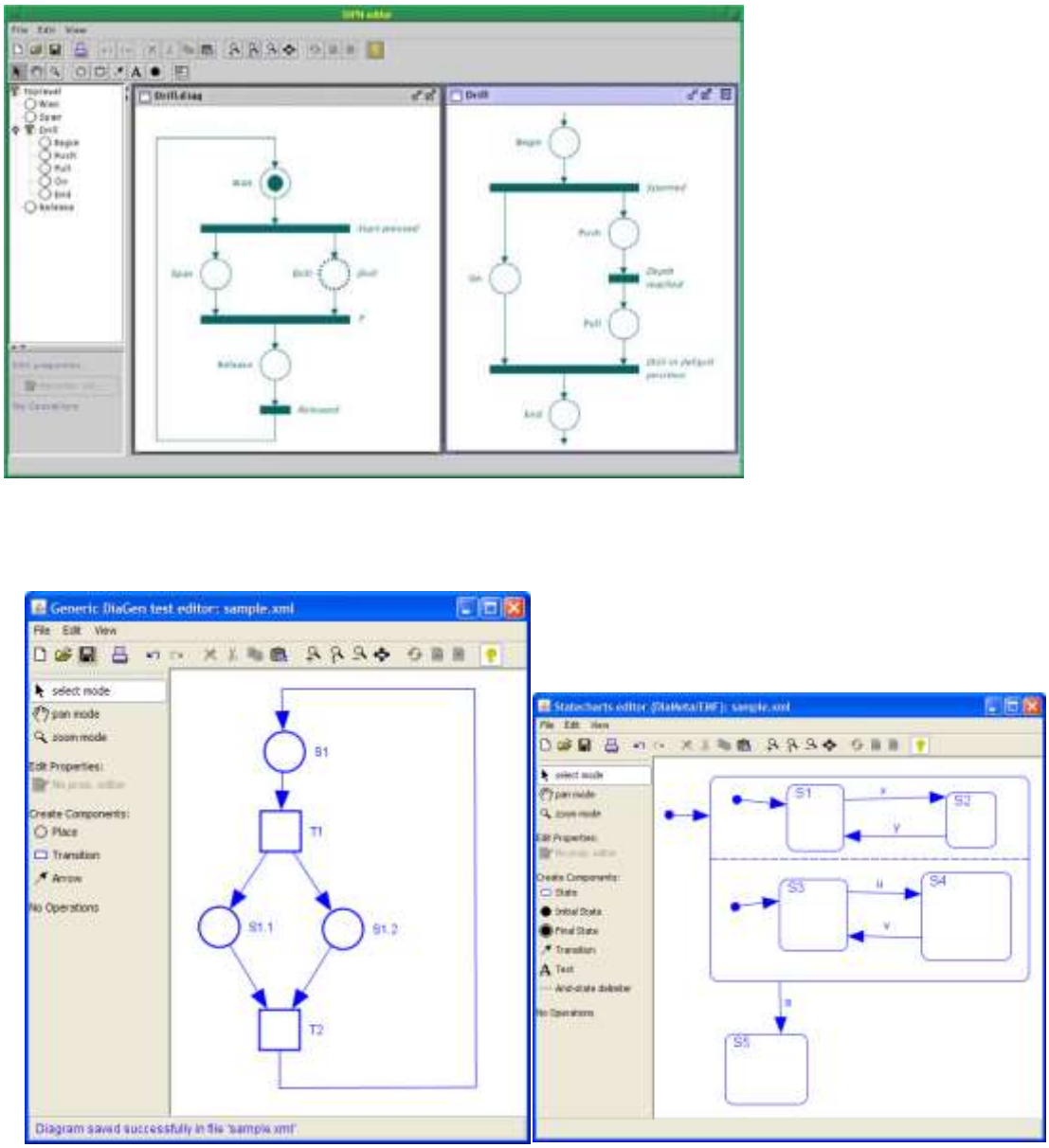


Σχήμα 22: Το περιβάλλον του λογισμικού JGrafchart

5.1.2. DiaGen

Το DiaGen [86] είναι ένα σύστημα για την εύκολη δημιουργία εργαλείων σχεδίασης διαγραμμάτων ισχυρών δυνατοτήτων. Επιγραμματικά αναφέρεται η δημιουργία εργαλείων σχεδίασης για:

- διαγράμματα κλιμακωτής λογικής (Ladder logic diagrams),
- διαγράμματα για ιεραρχικά δίκτυα Petri,
- διαγράμματα καταστάσεων και
- διαγράμματα ροής (flow charts)

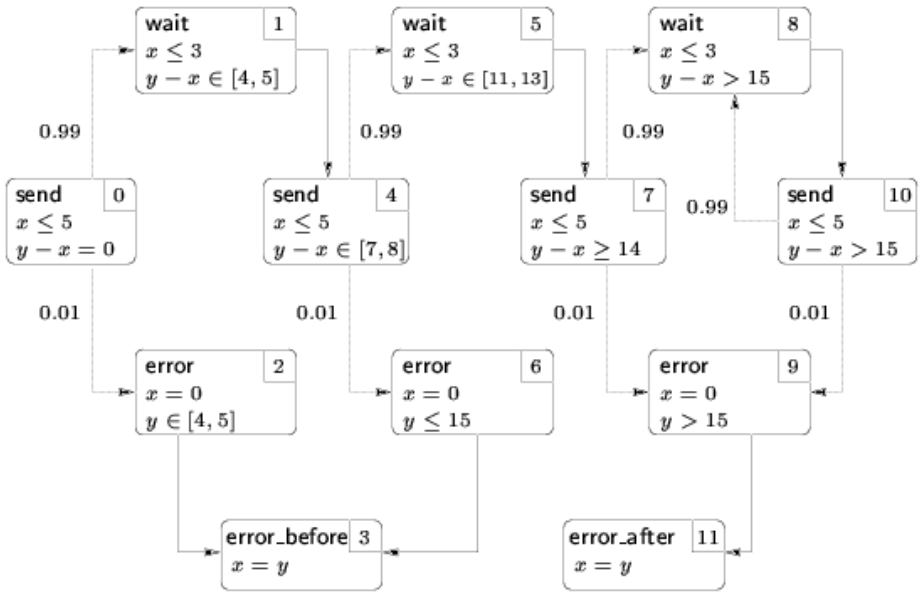


Σχήμα 23: Το περιβάλλον του λογισμικού DiaGen

Τα παραγόμενα εργαλεία μπορούν να χρησιμοποιηθούν και ως εργαλεία εξομοίωσης και επαλήθευσης συστημάτων. Στο Σχήμα 23 απεικονίζονται ενδεικτικά παραδείγματα χρήσης για διάφορα διαγράμματα.

5.1.3. KRONOS

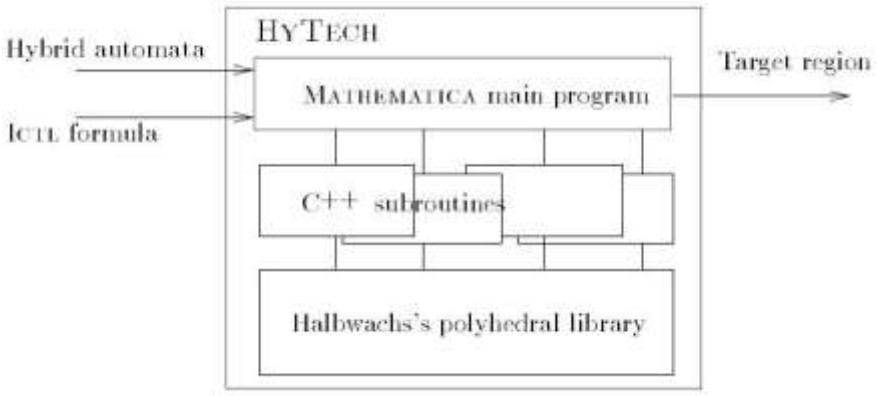
Το KRONOS [87] είναι ένα λογισμικό εργαλείο επαλήθευσης, για συστήματα πραγματικού χρόνου. Είναι βασισμένο στη θεωρία των χρονισμένων αυτομάτων και της χρονικής λογικής (temporal logic). Συγκεκριμένα, τα μέρη ενός συστήματος πραγματικού χρόνου μοντελοποιούνται ως χρονισμένα αυτόματα, ενώ οι απαιτήσεις επαλήθευσης εκφράζονται μέσω της χρονικής λογικής TCTL. Μερικές από τις κύριες λειτουργίες του KRONOS είναι: εμπρόσθια και οπίσθια ανάλυση, βελτιστοποίηση χρήσης των ρολογιών του μοντέλου, δημιουργία παραδειγμάτων εκτέλεσης και δημιουργία διαγραμμάτων αποφάσεων.



Σχήμα 24: Ενδεικτικό διάγραμμα από το λογισμικό KRONOS

5.1.4. HyTech

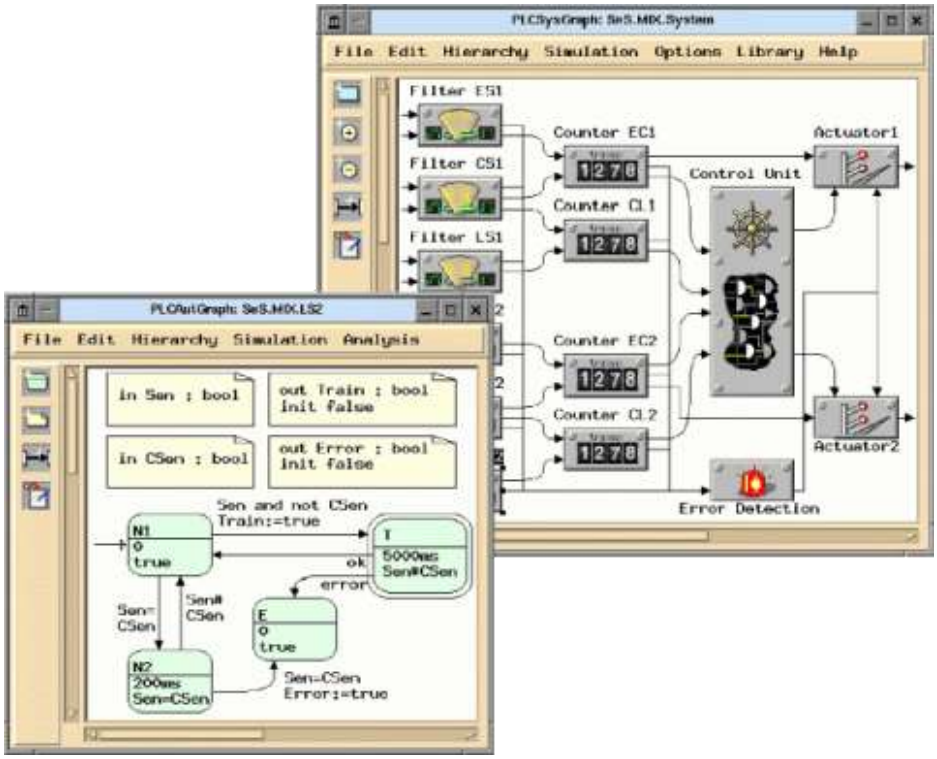
Το HyTech [88] είναι ένα λογισμικό εργαλείο για την ανάλυση ενσωματωμένων συστημάτων. Τα διάφορα μέρη του συστήματος μοντελοποιούνται ως γραμμικά υβριδικά αυτόματα, δηλαδή υβριδικά αυτόματα των οποίων οι συνεχείς συνιστώσες διέπονται από γραμμικές συναρτήσεις των μεταβλητών και των παραγώγων τους. Το HyTech χρησιμοποιείται κυρίως για τον υπολογισμό των συνθηκών κάτω από τις οποίες το σύστημα ικανοποιεί τις δεδομένες απαιτήσεις. Η ικανοποίηση των απαιτήσεων επαληθεύεται με χρήση συμβολικών μοντέλων ελέγχου. Σε περίπτωση αποτυχίας επαλήθευσης, το HyTech παράγει κώδικα για την διάγνωση του λάθους. Η δομή του λογισμικού πακέτου HyTech φαίνεται σχηματικά στο Σχήμα 25.



Σχήμα 25: Σχηματικά η δομή και λειτουργία του λογισμικού HyTech

5.1.5. Moby/PLC

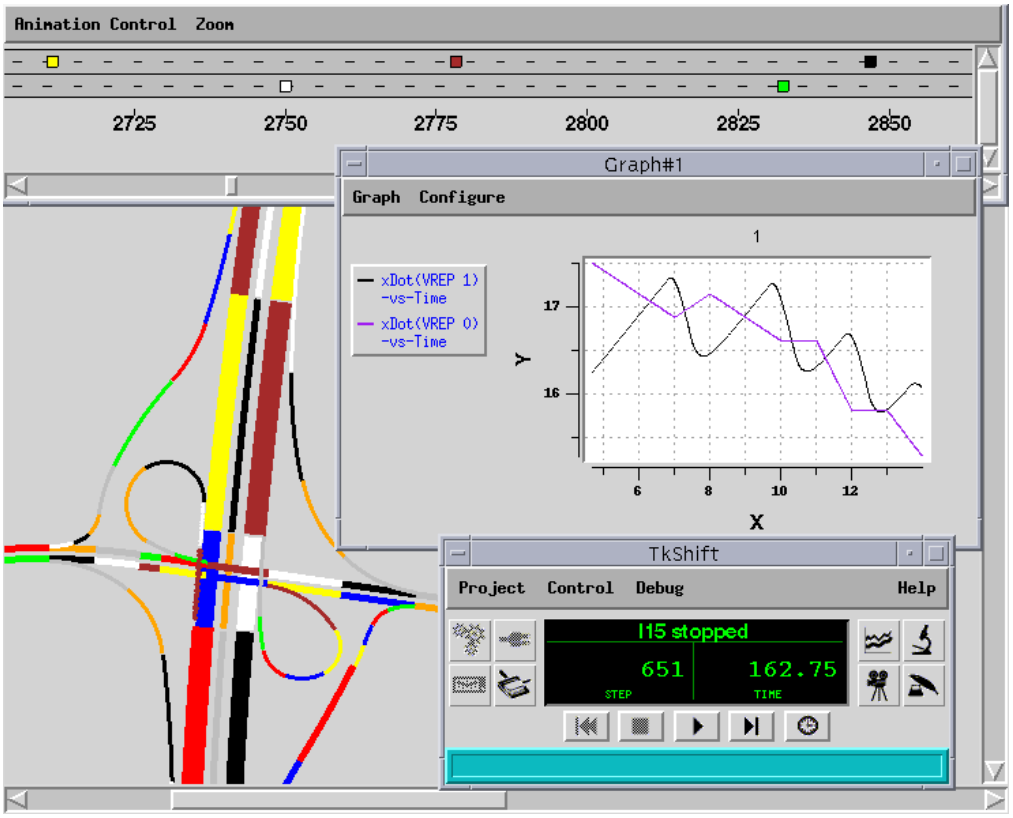
Το Moby/PLC [118] υποστηρίζει την ανάπτυξη κατανεμημένων προγραμμάτων ελέγχου πραγματικού χρόνου για Προγραμματιζόμενους Λογικούς Ελεγκτές. Αρχικά, το σύστημα μοντελοποιείται ως ένα δίκτυο από PLC-αυτόματα με χρήση του βασικού εργαλείου του Moby/PLC, ενός γραφικού σχεδιαστή. Στη συνέχεια το σύστημα μπορεί να επεξεργαστεί από τα υπόλοιπα εργαλεία του λογισμικού πακέτου Moby/PLC. Ενδεικτικά αναφέρουμε τις λειτουργίες εξομοίωσης και επαλήθευσης του όλου συστήματος. Τέλος, ένας κατάλληλος μεταγλωττιστής παράγει κώδικα για PLCs σε μορφή δομημένου κειμένου (structured text). Οθόνες του παραθυρικού περιβάλλοντος του Moby/PLC απεικονίζονται στο Σχήμα 26.



Σχήμα 26: Το περιβάλλον του λογισμικού Moby/PLC

5.1.6. Shift

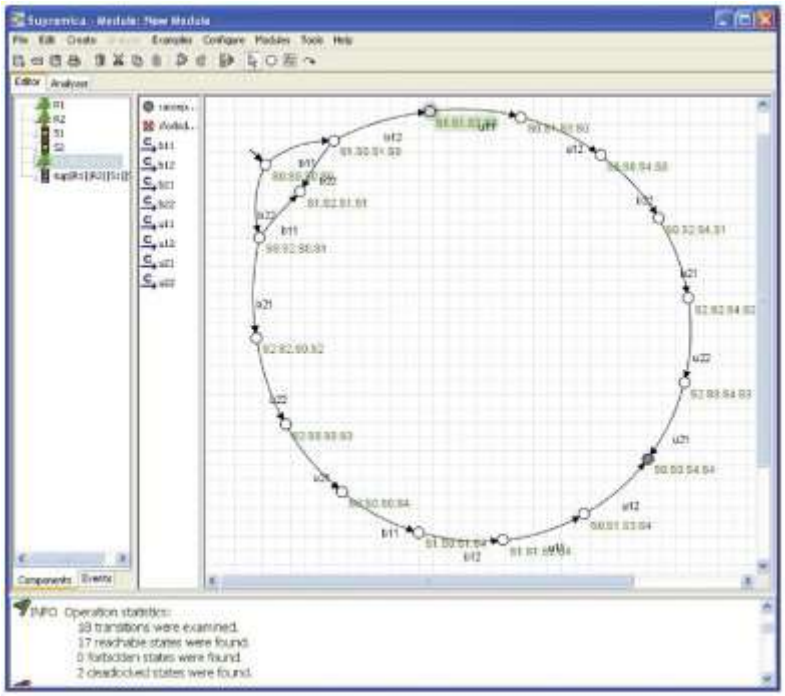
Το Shift [139] είναι μια γλώσσα προγραμματισμού για την περιγραφή δυναμικών δικτύων αποτελούμενων από υβριδικά αυτόματα. Τέτοια συστήματα αποτελούνται από διάφορα μέρη, τα οποία μπορεί να δημιουργούνται, να συνδέονται μεταξύ τους ή και να καταστρέφονται καθώς το σύστημα εξελίσσεται. Τα διάφορα μέρη παρουσιάζουν υβριδική συμπεριφορά, αποτελούμενη από φάσεις συνεχούς μεταβολής που διαχωρίζονται από διακριτές μεταβάσεις. Αλληλεπίδραση μεταξύ των μερών μπορεί να γίνει μέσω των εισόδων και των εξόδων κάθε μέρους, χωρίς όμως να αποκλείεται η ανεξαρτησία κάποιου τμήματος. Το Shift από την παρουσίαση του είχε βρει κύριο χώρο εφαρμογής τα συστήματα διαχείρισης κυκλοφορίας σε αυτοκινητοδρόμους. Μια οπτική απεικόνιση του αντίστοιχου λογισμικού πακέτου παρουσιάζεται στο Σχήμα 27.



Σχήμα 27: Το περιβάλλον του λογισμικού Shift

5.1.7. Supremica

Το Supremica είναι ένα λογισμικό εργαλείο για μεθοδική σύνθεση και επαλήθευση συστημάτων ελέγχου. Συνήθως χρησιμοποιείται δημιουργώντας μοντέλο του υπό έλεγχο συστήματος και των επιθυμητών αποκρίσεων κλειστού βρόχου. Ακολούθως το Supremica παράγει έναν κατάλληλο επόπτη που να ικανοποιεί τις δοθείσες προδιαγραφές. Τέλος, το Supremica έχει την δυνατότητα παραγωγής εκτελέσιμου κώδικα για κάθε συμβατό, κατά IEC 61131, PLC. Στο Σχήμα 28 εικονίζεται μια ενδεικτική οθόνη του περιβάλλοντος του Supremica [149].



Σχήμα 28: Το περιβάλλον του λογισμικού Supremica

5.2. Το λογισμικό μοντελοποίησης και επαλήθευσης UPPAAL

Το UPPAAL αποτελεί ένα ολοκληρωμένο περιβάλλον το οποίο συνδυάζει εργαλεία κατάλληλα για το σχεδιασμό, την προσομοίωση και την επαλήθευση συστημάτων πραγματικού χρόνου που μοντελοποιούνται ως δίκτυα χρονισμένων αυτομάτων, εμπλουτισμένα με τύπους δεδομένων (ακέραιοι, πίνακες, κ.λπ.). Στο UPPAAL κάθε ένα από αυτά τα χρονισμένα αυτόματα συνιστά μια διεργασία, ενώ το δίκτυό τους περιγράφει συστήματα πραγματικού χρόνου. Πρόκειται για συστήματα των οποίων η ορθότητα δεν εξαρτάται μόνο από την έξοδο αλλά και από το χρόνο κατά τον οποίο η έξοδος αυτή παράγεται [132].

Το UPPAAL (UPsala-AALborg) έχει αναπτυχθεί σε συνεργασία του Τμήματος Πληροφορικής του Πανεπιστημίου της Uppsala, και του Τμήματος Επιστήμης Υπολογιστών του Πανεπιστημίου του Aalborg [85]. Στις τυπικές περιοχές εφαρμογής του UPPAAL περιλαμβάνονται ελεγκτές πραγματικού χρόνου και πρωτόκολλα επικοινωνίας, όπου οι προδιαγραφές χρόνου θεωρούνται κρίσιμες [85]. Η πρώτη έκδοση του Uppaal κυκλοφόρησε το 1995. Από τότε το εργαλείο βρίσκεται σε διαρκή ανάπτυξη. Τα πειράματα και οι βελτιώσεις περιλαμβάνουν δομές δεδομένων, μερική μείωση της τάξης, μείωση της συμμετρίας, καθοδήγηση με βάση το κόστος κ.α. [68].

Η ανάπτυξη του εργαλείου Uppaal έχει ως άξονα την αποδοτικότητα στη λειτουργία του. Ο περιορισμός στην ανάλυση προσεγγισιμότητας υπήρξε αποφασιστικός για την αποδοτικότητα της μονάδας ελέγχου του μοντέλου του UPPAAL. Ένας άλλος σημαντικός παράγοντας για την αποδοτικότητα υπήρξε η εφαρμογή της τεχνικής της επαλήθευσης κατά τη διάρκεια της εξερεύνησης του χώρου καταστάσεων (on-the-fly verification) που συνδυάζεται με τη συμβολική τεχνική που περιορίζει προβλήματα επαλήθευσης σε προβλήματα χειρισμού και επίλυσης απλών περιορισμών [94].

Ένα δεύτερο κριτήριο που ελήφθη υπόψη κατά την ανάπτυξη του UPPAAL είναι η φιλικότητα προς το χρήστη. Για να διευκολυνθεί η εύρεση λαθών στις περιγραφές των συστημάτων, η μονάδα που ευθύνεται για την επαλήθευση των προς εξέταση ιδιοτήτων (verifier) είναι εφοδιασμένη με τη δυνατότητα για υποστήριξη διαγνωστικών διαδρομών (diagnostic traces). Όταν η επαλήθευση μιας συγκεκριμένης ιδιότητας έχει αποβεί επιτυχής ή το αντίθετο, τότε ένα παράδειγμα παράγεται που επιδεικνύει γιατί η ιδιότητα ικανοποιήθηκε ή όχι από το αναλυμένο μοντέλο του συστήματος. Η φιλικότητα προς το χρήστη έχει βελτιωθεί περαιτέρω με την ανάπτυξη γραφικού περιβάλλοντος για την σχεδίαση των διαφόρων συστατικών στοιχείων ενός μοντέλου που υιοθετεί το πρότυπο του δικτύου των χρονισμένων αυτομάτων [109].

Σαν εργαλείο αποτελείται από δύο βασικά μέρη:

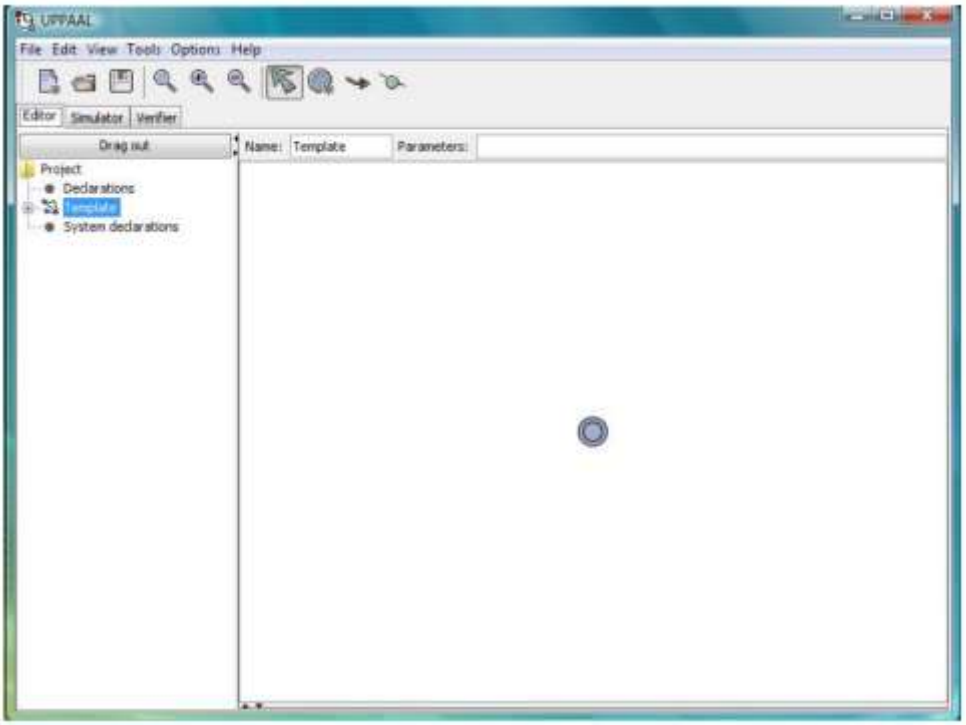
- το Γραφικό περιβάλλον χρήστη (Graphical User Interface - GUI) ή που εκτελείται στο σταθμό εργασίας του χρήστη και
- τη Μηχανή ελέγχου μοντέλων (Model-checker engine) που εκτελείται στον ίδιο υπολογιστή με το περιβάλλον χρήστη, αλλά υπάρχει δυνατότητα να εκτελεστεί και σε κάποιο τρίτο, πιο αποδοτικό, σύστημα/ εξυπηρετητή ελέγχοντας διάφορα μοντέλα που έχουν αναπτυχθεί σε οποιοδήποτε υπολογιστή έχει εγκατεστημένο το UPPAAL [96].

Ο server του UPPAAL παρέχει ένα αποδοτικό υπολογισμό του συμβολικού χώρου του συστήματος κατά τη διαδικασία επαλήθευσης μιας δοσμένης ιδιότητας. Το GUI εκκινεί το πρόγραμμα του server κάθε φορά που ο χρήστης αποφασίζει να ανανεώσει τον προσομοιωτή ή την μονάδα επαλήθευσης με ένα νέο μοντέλο συστήματος. Το GUI και ο server επικοινωνούν μέσω μηχανισμού σύνδεσης με TCP/IP sockets. Η σύνδεση αυτή εγκαθίσταται μετά το ξεκίνημα του server.

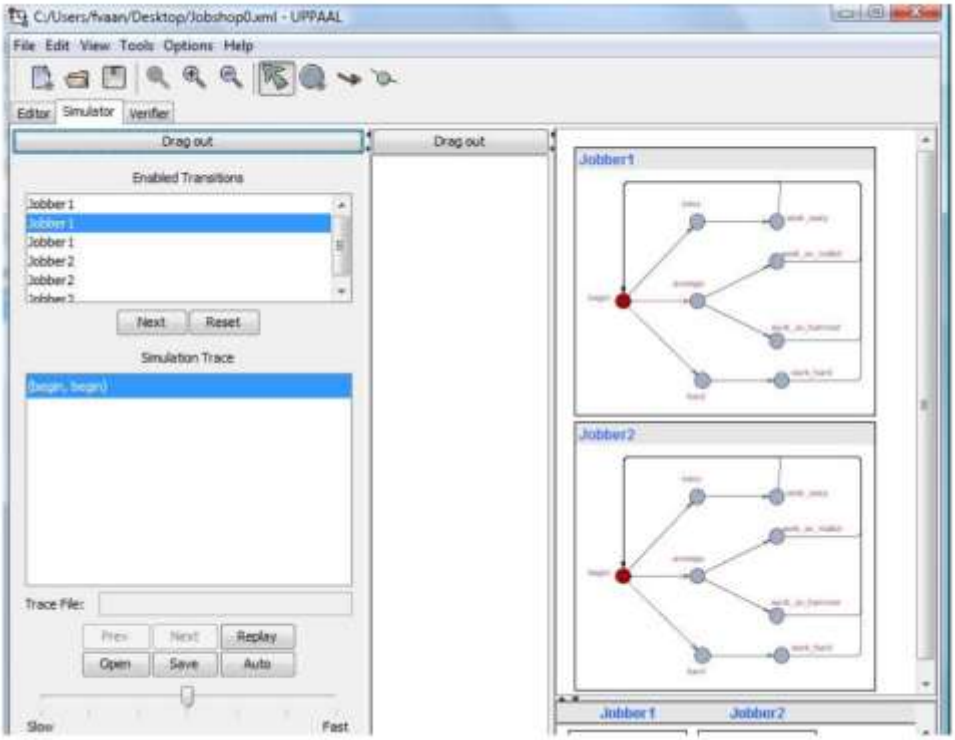
Το γραφικό περιβάλλον του UPPAAL αποτελείται από τρία κύρια μέρη, προσβάσιμα μέσω τριών καρτελών στο κύριο παράθυρο:

Τη **μονάδα σύνταξης του μοντέλου (system editor)** που μπορεί να χρησιμοποιηθεί για την κατασκευή μοντέλων. Το μοντέλο αποτελείται από ένα φραγμένο αριθμό από παράλληλες διαδικασίες (*concurent processes*), κάθε μία από τις οποίες περιγράφεται από ένα χρονισμένο αυτόματο. Κάθε διαδικασία ενδέχεται να περιλαμβάνει τοπικής εμβέλειας (local) ρολόγια και μεταβλητές, τα οποία μπορούν να χρησιμοποιηθούν μόνο μέσα σε αυτή και μπορεί να κάνει χρήση των γενικής εμβέλειας (global) ρολογιών και μεταβλητών. Οι διαδικασίες επικοινωνούν μεταξύ τους μέσω των μεταβλητών γενικής εμβέλειας και των δυαδικών συγχρονισμών.

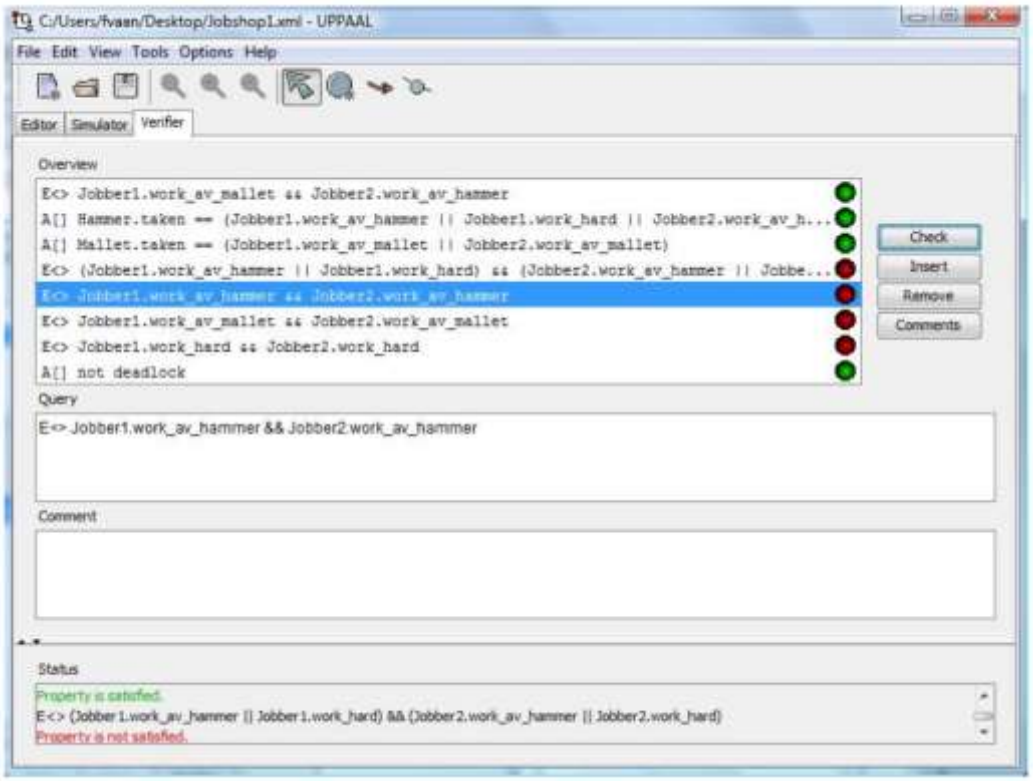
Τον **προσομοιωτή**, στον οποί μπορεί ο χρήστης να απεικονίσει τη συμπεριφορά των μοντέλων και τη **μονάδα επαλήθευσης (verifier),** στην οποία μπορεί να αναλυθεί η συμπεριφορά των μοντέλων και δέχεται κατάλληλα μορφοποιημένες ιδιότητες για να επαληθευτούν για ένα συγκεκριμένο μοντέλο αυτομάτων και απεικονίζει το αποτέλεσμα: σωστό ή λάθος αν η ιδιότητα ικανοποιείται ή όχι αντιστοίχως [67].



Σχήμα 29: Περιβάλλον του system editor [67]



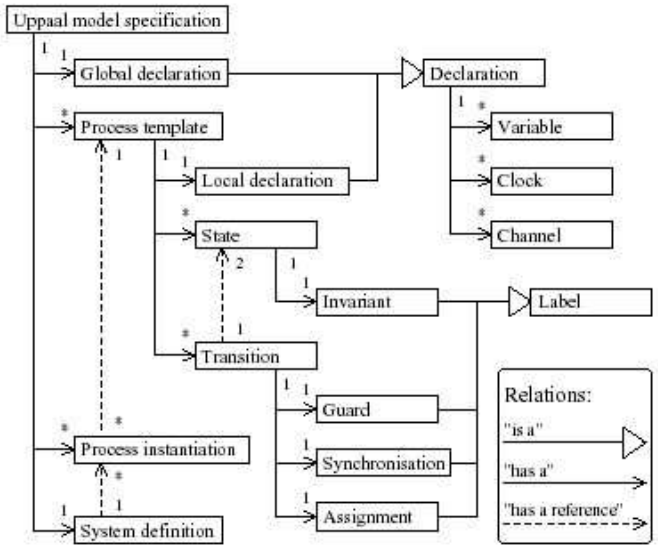
Σχήμα 30: Περιβάλλον του simulator [67]



Σχήμα 31: Περιβάλλον του verifier [67]

Στο UPPAAL, τα συστήματα μοντελοποιούνται με τη χρήση χρονισμένων αυτομάτων. Για να παρέχει ένα περισσότερο εκφραστικό μοντέλο και να διευκολύνει το έργο της μοντελοποίησης, το UPPAAL επεκτείνει τα timed automata με γενικότερους τύπους μεταβλητών δεδομένων όπως λογικές (boolean) και ακέραιες μεταβλητές. Ο τελικός σκοπός είναι η μορφοποίηση μια γλώσσας μοντελοποίησης που να είναι όσο το δυνατό περισσότερο συναφής με τις υψηλού επιπέδου και πραγματικού χρόνου γλώσσες προγραμματισμού που περιέχουν διάφορους τύπους δεδομένων. Κάτι τέτοιο ενδεχομένως θα δημιουργούσε προβλήματα λήψης απόφασης (decidability problems) κατά τον έλεγχο του μοντέλου.

Παρακάτω απεικονίζεται μια ολοκληρωμένη προδιαγραφή μοντέλου χρονισμένων αυτομάτων στο UPPAAL όπως παρέχεται από τους δημιουργούς του λογισμικού.



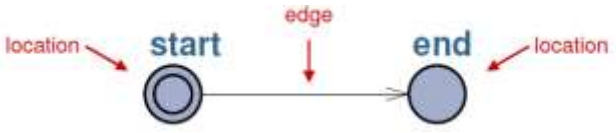
Σχήμα 32: Δομή μοντέλου χρονισμένων αυτομάτων στο UPPAAL

5.2.1. Τα συστατικά ενός μοντέλου στο UPPAAL

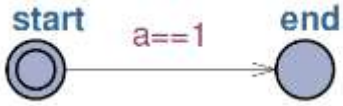
- **Οι εκφράσεις**: Οι εκφράσεις στο UPPAAL μοιάζουν πολύ με τις εκφράσεις γλωσσών προγραμματισμού, όπως η Java και η C++. Είναι δυνατή η χρήση λογικών τελεστών (and, or, not), ισότητας και ανισότητας, αριθμητικών και εκχώρησης τιμής. Οι τελεστές συνοψίζονται στον εξής πίνακα:

()	Καθορίζει την σειρά αποτίμησης των εκφράσεων	<	Μικρότερο από
[]	Indexing σε πίνακα	<=	Μικρότερο ή ίσο από
.	Lookup operator για πρόσβαση στις local μεταβλητές	==	Τελεστής ισότητας
!	Λογική άρνηση	!=	Τελεστής ανισότητας
++	Αύξηση	>=	Μεγαλύτερο ή ίσο από
--	Μείωση	>	Μεγαλύτερο από
-	Integer subtraction (can also be used as unary negation)	&	Τελεστής and για bits
+	Άθροιση ακεραίων	^	Τελεστής xor για bits
*	Γινόμενο ακεραίων	|	Τελεστής or για bits
/	Διαίρεση ακεραίων	&&	Λογικό and
%	Modulo	||	Λογικό or
<<	Αριστερή μετακίνηση bit	?:	Τελεστής If-then-else
>>	Δεξιά μετακίνηση bit	not	Λογική άρνηση
<?	Ελάχιστο	and	Λογικό and
>?	Μέγιστο	or	Λογικό or
: += -= *= %= &= |= >>= ^=	Τελεστής ανάθεσης τιμής	imply	Λογική συνεπαγωγή

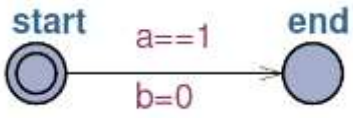
- **Οι χρονιστές ή ρολόγια** είναι μεταβλητές που μπορούν να αξιολογήσουν σε ένα πραγματικό αριθμό και οποίες μπορούν να ορισθούν σε κάθε αυτόματο προκειμένου να μετρηθεί την πρόοδο του χρόνου. Όλα τα ρολόγια εξελίσσονται με τον ίδιο ρυθμό, προκειμένου να εκπροσωπεί το παγκόσμια πρόοδο του χρόνου. Η πραγματική τιμή ενός ρολογιού μπορεί είτε να δοκιμαστεί ή επαναφερθεί (όχι να ανατεθεί). Δεδομένου ότι το εργαλείο UPPAAL είναι ειδικά σχεδιασμένο για την επαλήθευση των συστημάτων πραγματικού χρόνου, τα ρολόγια αποτελούν ένα θεμελιώδη μέσο μοντελοποίησης και επαλήθευσης. Ένα μοντέλο UPPAAL είναι χτισμένο ως ένα σύνολο ταυτόχρονων διαδικασιών, κάθε μια από τις οποίες είναι γραφικά σχεδιασμένη ως χρονισμένο αυτόματο. Κάθε χρονισμένο αυτόματο αντιπροσωπεύεται από ένα γράφημα το οποίο έχει θέσεις (locations) ως κόμβους και ακμές (edges) ως τόξα ανάμεσα στις θέσεις.



- **Οι μεταβάσεις.** Οι άκρες είναι ενημερωμένες με τους παράγοντες guards (φύλακες), updates (ενημερώσεις), synchronizations (συγχρονισμούς) και selections (επιλογές). Ένας φύλακας είναι μια έκφραση που χρησιμοποιεί τις μεταβλητές και ρολόγια του μοντέλου, ώστε να υποδεικνύει πότε η μετάβαση είναι ενεργοποιημένη. Σημειώνεται εδώ ότι πολλές ακμές μπορούν να ενεργοποιηθούν σε μία συγκεκριμένη χρονική στιγμή, αλλά μόνο μία από αυτές θα δράσει ώστε να οδηγήσει σε διαφορετικές δυνατότητες.



Μια ενημερωμένη έκδοση (update) είναι μια έκφραση που αξιολογείται το συντομότερο καθώς το αντίστοιχο άκρο βρίσκεται σε δράση. Η αξιολόγηση αλλάζει την κατάσταση του συστήματος.

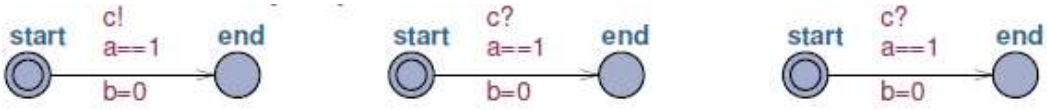


Ο συγχρονισμός είναι ο βασικός μηχανισμός που χρησιμοποιείται για συντονίζει τη δράση των δύο ή περισσότερες διεργασίες. Μοντελοποιεί, για παράδειγμα, την επίδραση των μηνυμάτων. Προκαλεί δύο (ή περισσότερες) διεργασίες να λάβουν μια μετάβαση την ίδια στιγμή. Όταν ένα κανάλι c δηλώνεται, τότε μια διαδικασία θα έχει επισημασμένη την άκρη ως c! Και η άλλη (ες) διαδικασία (ες) ως c?. Υπάρχουν τρία είδη συγχρονισμών, τα οποία περιγράφονται ακολούθως:

- **Regular channel**: δηλώνεται ως chan c. Όταν μια διαδικασία βρίσκεται σε μια θέση από την οποία υπάρχει μια μετάβαση που επισημαίνεται με το c!, ο μόνος τρόπος για τη μετάβαση να είναι ενεργοποιημένη είναι μια άλλη διαδικασία να είναι σε μια θέση από την οποία υπάρχει μετάβαση που επισημαίνεται με c? και το αντίστροφο. Εάν σε μια συγκεκριμένη στιγμή, υπάρχουν διάφοροι τρόποι για να έχουν ζεύγος c! και c?, ένας από αυτούς είναι μη- ντετερμινιστικά επιλεγμένος κατά τον έλεγχο μοντέλου. Η ενημέρωση σε μια άκρη συγχρονισμού για c! εκτελείται πριν από την ενημέρωση σε μια άκρη συγχρονισμού για c?.

- **Urgent channel:** δηλώνεται ως urgent chan c. Είναι παρόμοιο με το regular channel, εκτός από το γεγονός ότι δεν είναι δυνατό να καθυστερήσει στο σταθμό προέλευσης, αν είναι εφικτό να ενεργοποιήσει το συγχρονισμό πάνω από το regular channel. Αυτό σημαίνει ότι δεν μπορεί να περάσει χρόνος, αλλά μπορεί να εναλλάσσει άλλες μεταβάσεις που δεν απαιτούν χρόνο για να περάσει. Σημειώνεται ότι τα ρολόγια «φύλακες» δεν επιτρέπονται στις άκρες πάνω από ένα urgent channel.

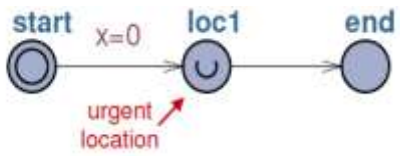
- **Broadcast channel**: δηλώνεται ως broadcast chan c. Όταν μία διεργασία βρίσκεται σε μια θέση από την οποία υπάρχει μια μετάβαση που επισημαίνεται με c! και μία ή περισσότερες διεργασίες είναι σε θέσεις από τις οποίες υπάρχει μια μετάβαση επισημασμένη με c?, όλες οι μεταβάσεις είναι ενεργοποιημένες. Ωστόσο, εάν δεν υπάρχουν διεργασίες στις θέσεις από τις οποίες υπάρχει μια μετάβαση επισημασμένη με c?, η μετάβαση επισημασμένη με c! είναι ενεργοποιημένη ούτως ή άλλως. Παρατηρείται ότι τα ρολόγια «φύλακες» δεν επιτρέπονται στις άκρες που λαμβάνουν πάνω σε ένα broadcast channel. Η ενημέρωση σχετικά με την άκρη εκπομπής εκτελείται πρώτα. Η ενημέρωση σχετικά με τις ακμές που λαμβάνουν εκτελούνται αριστερά προς τα δεξιά, σύμφωνα με τη διάταξη που οι διεργασίες καταχωρήθηκαν κατά τον ορισμό του συστήματος.



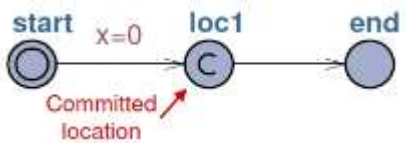
- **Οι θέσεις.** Αναφορικά με τις θέσεις (locations), μπορούν να έχουν ένα προαιρετικό όνομα, το οποίο χρησιμεύει στο να εντοπίζεται η θέση κατά τη διάρκεια του ελέγχου και της τεκμηρίωσης του μοντέλου. Οι θέσεις μπορεί να είναι τριών ειδών:
- **Initial (αρχικές),** των οποίων η ύπαρξη σε κάθε πρότυπο είναι υποχρεωτική. Κάθε πρότυπο πρέπει να έχει προετοιμαστεί σωστά, πράγμα που σημαίνει ότι πρέπει να ξεκινήσει σε μια συγκεκριμένη θέση. Ως εκ τούτου, κάθε

πρότυπο πρέπει να έχει ακριβώς μία θέση που επισημαίνεται ως αρχική. Οι αρχικές θέσεις προσδιορίζονται με διπλό κύκλο.

- **Urgent (επείγουσες),** οι οποίες παγώνουν το χρόνο, δηλαδή δεν επιτρέπεται να παρέλθει χρονικό διάστημα όσο μια διαδικασία βρίσκεται σε εξέλιξη. Η κάθε θέση πρέπει να εγκαταλειφθεί προτού περάσει ο χρόνος. Οι υπόλοιπες μεταβάσεις μπορούν να πραγματοποιηθούν προηγουμένως εφόσον δεν απαιτούν την παρέλευση χρόνου. Στο μοντέλο, αυτού του είδους οι θέσεις επισημαίνονται με U.

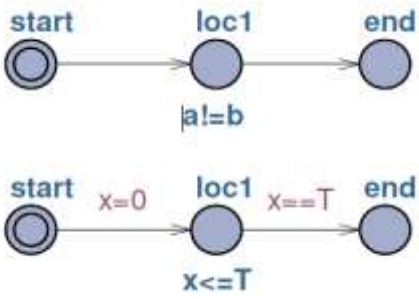


- **Committed (δεσμευμένες),** οι οποίες επίσης παγώνουν το χρόνο. Όταν ένα μοντέλο έχει ενεργές μία ή περισσότερες τέτοιες θέσεις τότε καμία άλλη διεργασία, πέρα από αυτές που εγκαταλείπουν τις επισημασμένες θέσεις, δεν επιτρέπεται να ενεργοποιηθεί. Οι συγκεκριμένες θέσεις είναι χρήσιμες στο να δημιουργούν ατομικές ακολουθίες. Μέσα στο μοντέλο επισημαίνονται με C.



- **Normal (κανονικές),** δηλαδή όλες οι θέσεις του κάθε αυτομάτου οι οποίες δεν έχουν χαρακτηριστεί ως Initial, Urgent ή Committed.

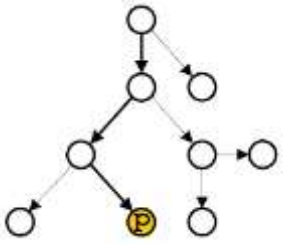
Τόσο στις αρχικές όσο και στις κανονικές θέσεις μπορούν να αποδοθούν ορισμένες σταθερές, οι οποίες εκφράζουν τις συνθήκες που πρέπει να ικανοποιούνται όσο το αυτόματο παραμένει σε αυτή τη θέση. Οι σταθερές αυτές μπορούν να σχετίζονται είτε με τα ρολόγια είτε με τις μεταβλητές [96].



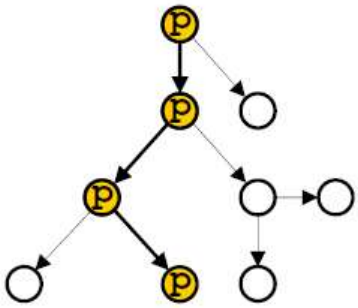
5.2.2. Επαλήθευση στο UPPAAL

Εφόσον έχει χρησιμοποιηθεί ο προσομοιωτής (simulator), για να εξασφαλιστεί ότι το μοντέλο λειτουργεί και συμπεριφέρεται όπως ακριβώς το προς μοντελοποίηση σύστημα (και μερικές φορές για να ανιχνευτούν τυχόν λάθη στον αρχικό σχεδιασμό), η επόμενη φάση είναι να γίνει έλεγχος κατά πόσο επαληθεύονται οι ιδιότητες του συστήματος. Αρχικά λοιπόν θα πρέπει να γίνουν γνωστές αυτές οι ιδιότητες και να «επισημοποιηθούν» και κατά δεύτερον να μεταφραστούν αυτές οι ιδιότητες στη γλώσσα επερωτήσεων του UPPAAL (Uppaal query language). Τα είδη των ιδιοτήτων που μπορούν να ελέγχονται άμεσα χρησιμοποιώντας τα UPPAAL ερωτήματα είναι αρκετά απλά. Οι σχεδιαστές έχουν υιοθετήσει αυτή την προσέγγιση, αντί να επιτρέπουν σύνθετα ερωτήματα, προκειμένου να βελτιωθεί η απόδοση του εργαλείου. Για το λόγο αυτό, η επαλήθευση των σύνθετων ιδιοτήτων μπορεί να απαιτεί τον έλεγχο πολλών διαφορετικών ερωτημάτων και ακόμη και την προσθήκη στο μοντέλο ειδικά σχεδιασμένων "ελεγμένων αυτομάτων". Οι ακριβείς κατηγορίες ιδιοτήτων που μπορούν να εκφραστούν στη γλώσσα επερωτήσεων του UPPAAL είναι:

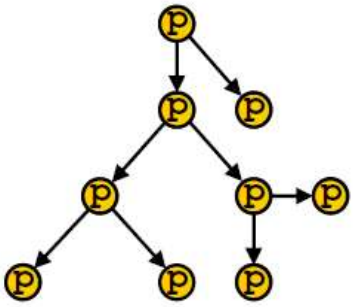
- **Reachability properties:** η συγκεκριμένη κατάσταση παραμένει σε κάποιο δεδομένο σταθμό των ενδεχόμενων συμπεριφορών του μοντέλου. Πάντα εκφράζονται με τον τύπο E < > p “Exists eventually p”, που σημαίνει ότι υπάρχει μια διαδρομή εκτέλεσης όπου το p τελικά παραμένει.



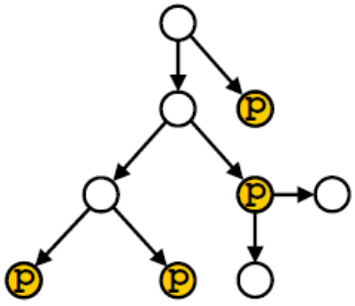
- **Safety properties:** Η κατάσταση παραμένει ίδια σε όλους τους σταθμούς της διαδρομής της εκτέλεσης. Υπάρχουν δύο διαφορετικές περιπτώσεις:
- E [ ] p “Exists globally p”, που σημαίνει ότι υπάρχει ένα μονοπάτι εκτέλεσης κατά το οποίο το p παραμένει σε όλους τους σταθμούς του μονοπατιού



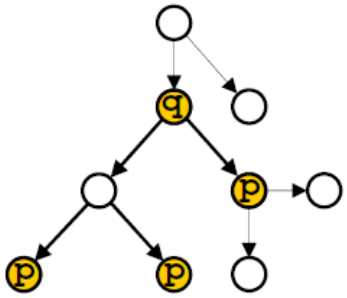
- A [ ] p “Always globally p”, όπου για κάθε μονοπάτι εκτέλεσης το p διατηρείται σε όλο το μονοπάτι



- **Liveness properties:** μια συγκεκριμένη κατάσταση είναι εγγυημένο ότι θα διατηρηθεί περιστασιακά (σε μια δεδομένη στιγμή). Υπάρχουν και εδώ δύο περιπτώσεις:
- A<>p “always eventually p” όπου για κάθε μονοπάτι εκτέλεσης το p συγκρατείται μόνο για ένα σταθμό του μονοπατιού.



- q→p “q always leads to p”, όπου κάθε μονοπάτι που ξεκινά από ένα σταθμό που παραμένει το q καταλήγει πάντα σε ένα σταθμό που παραμένει το p.



- **Deadlock properties** που είτε είναι δυνατές είτε όχι σε ένα μοντέλο. Κατάσταση deadlock επικρατεί εφόσον είναι δυνατό το μοντέλο να εξελίσσεται σε μια διαδοχική στάση χωρίς να περιμένει κάποιο χρονικό διάστημα ή μια μετάβαση μεταξύ θέσεων. Δύο τυπικά παραδείγματα είναι:
    - E < > deadlock: “Exists deadlock”
    - A [ ] not deadlock: “There is no deadlock”

Σημειώνεται εδώ ότι η λέξη «deadlock» μπορεί να χρησιμοποιηθεί στο εσωτερικό οποιασδήποτε έκφρασης επισημοποιώντας μια συγκεκριμένη ιδιότητα. Όταν χρησιμοποιείται έλεγχος μοντέλου για την επαλήθευση ενός συστήματος, έχει συνήθως μελετηθεί αν υπάρχει αδιέξοδο στην μοντέλο ή όχι [96].

Κεφάλαιο 6:
Ανάπτυξη πρότυπων μοντέλων χρονικού προγραμματισμού εργασιών με χρονισμένα αυτόματα

Όπως αναλύθηκε στο Κεφάλαιο 4 το κλασικό πρόβλημα job-shop έχει προσεγγισθεί στη βιβλιογραφία μέσω της ανάπτυξης ορισμένων απλουστευμένων μοντέλων τα οποία μέσω ανάλυσης προσπελασιμότητας ελαχιστοποιούν το συνολικό χρόνο ολοκλήρωσης των εργασιών (makespan). Παρά το γεγονός όμως ότι το makespan αναφέρεται στις περισσότερες ερευνητικές εργασίες ως η βασική αντικειμενική συνάρτηση βελτιστοποίησης, στην πράξη ελάχιστα παραγωγικά συστήματα λειτουργούν με άξονα την ελαχιστοποίηση αυτού. Αντίθετα, μια εταιρεία δίνει συνήθως μεγαλύτερη έμφαση σε άλλα χαρακτηριστικά τα οποία σχετίζονται είτε με το κόστος παραγωγής είτε με την εξυπηρέτηση του πελάτη. Τέτοια χαρακτηριστικά είναι η ελαχιστοποίηση του μέσου χρόνου καθυστέρησης (που σχετίζεται με την υπέρβαση του χρόνου παράδοσης των παραγγελιών), η ελαχιστοποίηση του αριθμού των καθυστερημένων παραγγελιών και η ελαχιστοποίηση του χρόνου αναμονής των μηχανών ή του συνολικού χρόνου που αυτές παραμένουν ενεργοποιημένες. Επιπρόσθετα, ακόμα και οι επιχειρήσεις που έχουν δομή παραγωγικού συστήματος «τύπου jobshop», στην πλειονότητα των περιπτώσεων δε

λειτουργούν βάσει των περιορισμών του κλασικού προβλήματος jobshop. Συνήθως αυτό συμβαίνει είτε γιατί κάποιος πόρος της παραγωγής δύναται να είναι πολλαπλός (πχ να υπάρχουν 2 ή περισσότερες ίδιες μηχανές), είτε γιατί οι ίδιες οι συνταγές που καθορίζουν τη ροή μιας παραγγελίας στους παραγωγικούς πόρους (μηχανές) παρέχουν δυνατότητα η ίδια εργασία να εκτελεστεί από κάποιο εναλλακτικό πόρο (συνήθως με κάποια επιβάρυνση είτε κόστους είτε χρόνου επεξεργασίας).

Στο πλαίσιο που αναλύθηκε ανωτέρω, στόχος του κεφαλαίου αυτού αποτελεί η ανάπτυξη ενός ολοκληρωμένου μοντέλου το οποίο από τη μία θα μοντελοποιεί διαφορετικές αντικειμενικές συναρτήσεις προς βελτιστοποίηση και από την άλλη θα διευρύνει το μοντέλο παραγωγής προκειμένου να καλύπτονται περιπτώσεις παραγωγικών μοντέλων που αποκλίνουν του τυπικού προβλήματος jobshop.

Στο επόμενο κεφάλαιο το μοντέλο θα επεκταθεί περαιτέρω προκειμένου να καλύψει μία ακόμα περίπτωση που συναντάται σε πραγματικά παραγωγικά συστήματα: τη δυνατότητα κάποια εργασία που εκτελείται σε μία μηχανή να διακοπεί πριν ολοκληρωθεί, να χρησιμοποιηθεί η μηχανή αυτή για μια άλλη – πιο επείγουσα – εργασία και στη συνέχεια η πρώτη εργασία να συνεχιστεί και να ολοκληρωθεί στην ίδια μηχανή.

6.1. Ανάπτυξη βασικού μοντέλου

Ξεκινώντας την ανάπτυξη του βασικού μοντέλου χρονοπρογραμματισμού με χρονισμένα αυτόματα στο οποίο θα ενσωματωθούν τα χαρακτηριστικά που αναφέρθηκε προηγούμενα, παρατίθενται οι βασικοί ορισμοί και στη συνέχεια αναλύεται το προτεινόμενο μοντέλο.

6.1.1. Χαρακτηριστικά προβλήματος

Έστω M είναι ένα πεπερασμένο σύνολο πόρων (μηχανών). 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 διεργασιών (βημάτων) της J,
- $\mu : \{1, \ldots, \mathrm{k}\} \to \mathrm{M}$ δείχνει ποια μηχανή χρησιμοποιείται σε κάθε βήμα και
- $d : \{1, \ldots, k\} \to \aleph$ δείχνει τη διάρκεια κάθε διεργασίας.

Στο πρόβλημα χρονοπρογραμματισμού, θεωρούμε υπάρχει ένα σύνολο εργασιών $J = \{J^1, \ldots, J^n\}$ με $J^i = (k^i, \mu^i, d^i)$ και $i = 1, \ldots, n$.

Επίσης θεωρούμε 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 $K = \{1,\ldots,k\}$.

6.1.2. Εφικτά χρονοπρογράμματα

Ένα εφικτό χρονοπρόγραμμα ενός προβλήματος 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 τη διεργασία j^{th} σε κάποιο χρόνο t και απασχολεί τη μηχανή $\mu^i(j)$.

Κάθε πρόγραμμα παραγωγής πρέπει να ικανοποιεί τις παρακάτω συνθήκες/ προϋποθέσεις:

i. Διάταξη: Διεργασίες που αναφέρονται στην ίδια εργασία εκτελούνται με συγκεκριμένη σειρά. Αν $(i,j,t) \in S$ και $(i,j',t') \in S$ τότε $j \le j'$ και $t \le t'$.

ii. Επικάλυψη και μη αντικατάσταση: Κάθε διεργασία εκτελείται συνεχώς μέχρι την πλήρη ολοκλήρωσή της (δεν υπάρχει δυνατότητα «προσωρινής» διακοπής της). Για κάθε $i \in J$ και $j \in K$ το σύνολο $\{t : (i,j,t) \in S\}$ είναι ένα σύνολο της μορφής $[r, r+d]$ για $r \in T$ και $d = d^i(j)$.

iii. Αμοιβαίος αποκλεισμός: Δύο διεργασίες δύο διαφορετικών εργασιών οι οποίες εκτελούνται την ίδια χρονική στιγμή δεν χρησιμοποιούν την ίδια μηχανή (αμοιβαίος αποκλεισμός/ 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 $\mu^i(j) \neq \mu^{i'}(j')$.

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 $(i,j,t) \in S$.

Από τους προηγούμενους ορισμούς που δώσαμε αναφορικά με τα χρονοπρογράμματα εξάγονται δυο συναρτήσεις:

- Η συνάρτηση κατανομής των μηχανών *α*: $\mathrm{M} \times \mathrm{T} \to J$

Η συνάρτηση αυτή ορίζεται ως *α*(*m*,*t*)=*i* αν $(i, j, t) \in S$ και $\mu^i(j) = m$. Δείχνει ποια εργασία απασχολεί ποια μηχανή και για πόσο χρόνο.

- Η συνάρτηση προόδου των εργασιών *β*: $J \times T \to M$

  Η συνάρτηση αυτή ορίζεται ως *β*(*i*,*t*)=*m* αν $(i, j, t) \in S$ και $\mu^i(j) = m$. Δείχνει ποια μηχανή χρησιμοποιείται από ποια εργασία σε δεδομένο χρόνο.

Μια μηχανή ή μια εργασία δύναται σε συγκεκριμένες χρονικές στιγμές να είναι ***αδρανής (idle).*** Το σύμβολο $\perp$ χρησιμοποιείται για κα εκφράσει την κατάσταση αυτή.

Συγκεκριμένα:

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

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

- 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

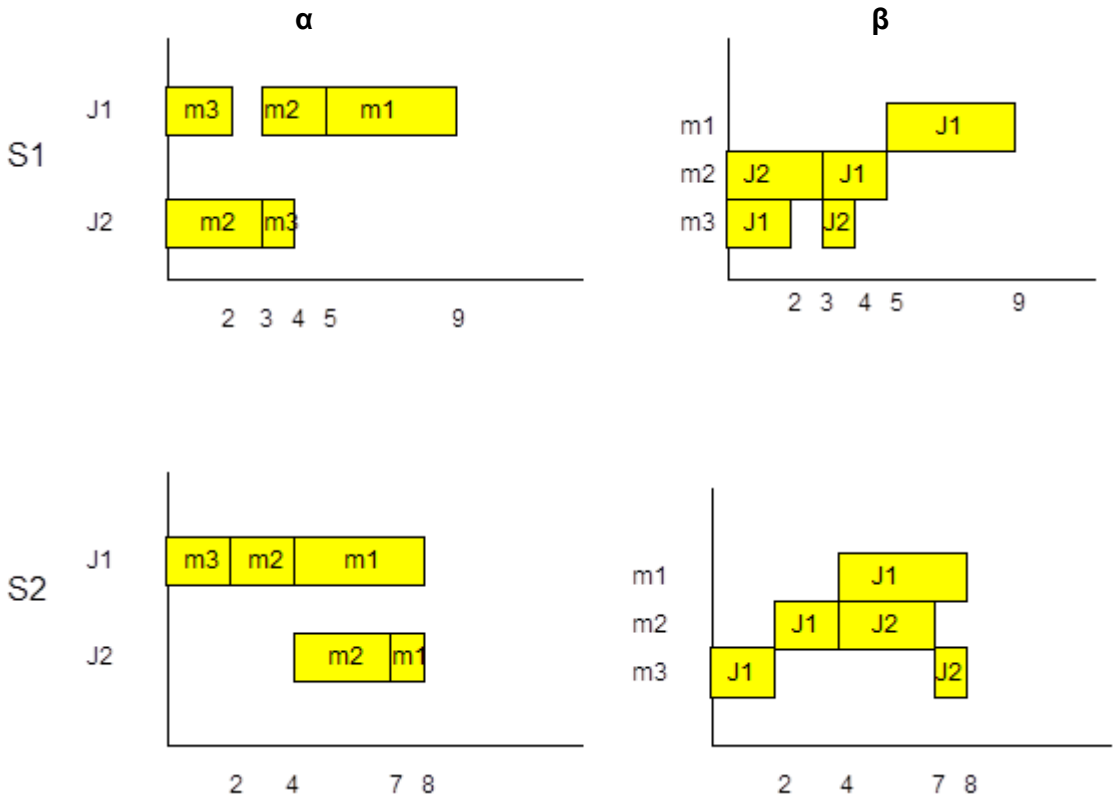
$$s(i, j-1) + d_i(j-1) < t < s(i, j) \text{ και } \alpha(\mathrm{m}, \mathrm{t}) = \perp$$

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

$$J^1 = (m_3,2),(m_2,2),(m_1,4) \text{ και}$$

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

Δύο πιθανά χρονοπρογράμματα απεικονίζονται στο Σχήμα 33. 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 33: Δύο χρονοπρογράμματα S_1 και 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 33 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.

6.2. Μοντελοποίηση με τη χρήση χρονισμένων αυτομάτων

6.2.1. Μαθηματική μοντελοποίηση

Πρώτο στάδιο δημιουργίας του προτεινόμενου γενικού μοντέλου είναι η μοντελοποίηση των εργασιών προς επεξεργασία. Αρχικά κατασκευάζουμε για κάθε εργασία ξεχωριστά $J = (k, m, d)$ ένα χρονισμένο αυτόματο με ένα χρονιστή *c* και 2k+1 καταστάσεις.

Όπως είναι σαφές μια διεργασία οποιασδήποτε εργασίας μπορεί:

i. να περιμένει πριν ξεκινήσει να εκτελείται,

ii. να εκτελείται ή

iii. να έχει ολοκληρωθεί αφού βρισκόταν σε κατάσταση εκτέλεσης.

Για να μοντελοποιηθεί η παραπάνω συμπεριφορά για κάθε διεργασία *j* τέτοια 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 μιας συγκεκριμένης διεργασίας και η κατάσταση *m* η οποία δείχνει ότι εκτελείται η διεργασία. Η αρχική κατάσταση είναι η $\overline{\mu}(1)$ όπου η εργασία *J* δεν έχει ξεκινήσει ακόμα, ενώ η τελική κατάσταση είναι η *f* όπου έχει ολοκληρωθεί η εκτέλεση όλων των διεργασιών της *J*. Στις καταστάσεις εκείνες στις οποίες οι εργασίες δεν έχουν ξεκινήσει την εκτέλεση των διεργασιών τους ($\overline{m}$) οι αντίστοιχοι χρονιστές *c* είναι αρχικά *μη ενεργοί,* ενώ με την έναρξη εκτέλεσης των εργασιών αρχίζουν να λειτουργούν ξεκινώντας από το μηδέν.

Στο πρώτο μοντέλο που κατασκευάζουμε, όπου δεν επιτρέπεται η διακοπή διεργασιών πριν την ολοκλήρωσή τους, 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 $d(j)$. Στο σημείο αυτό επισημαίνεται ότι στο επόμενο κεφάλαιο της εργασίας εξετάζεται και η περίπτωση των διακοπτόμενων διεργασιών, όπου μια διεργασία μπορεί προσωρινά να διακοπεί και να συνεχίσει σε μεταγενέστερο χρόνο.

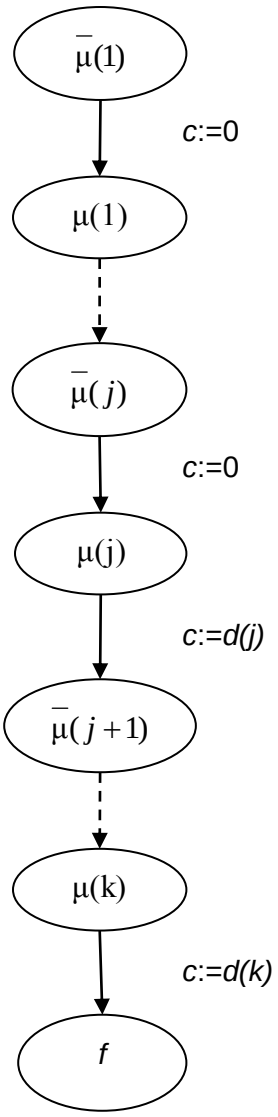
Έστω $\overline{M} = \left\{\overline{m} : m \in M\right\}$ και έστω $\overline{\mu} : \mathrm{K} \to \overline{\mathrm{M}}$ είναι μια βοηθητική συνάρτηση τέτοια ώστε $\overline{\mu}(j) = \overline{m}$ για οποιαδήποτε $\mu(j) = \mathrm{m}$.

Χρονισμένο αυτόματο μίας εργασίας: 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 34: Το αυτόματο που αντιστοιχεί στην εργασία $J=(k,m,d)$

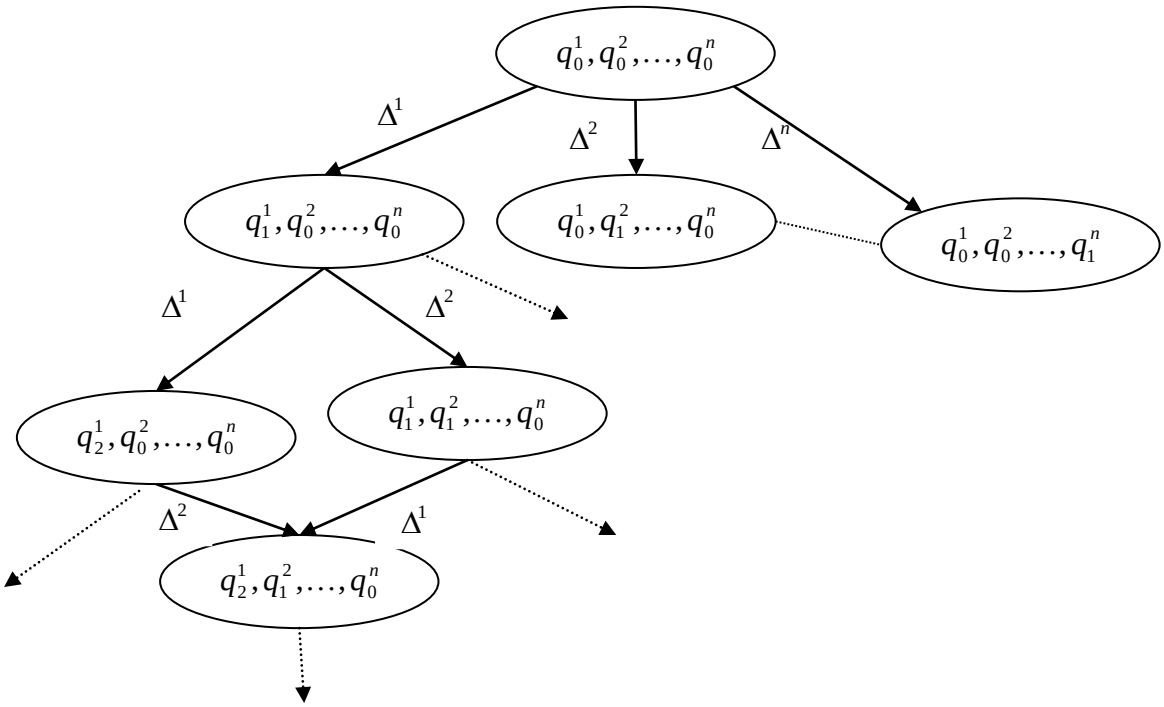
Το παραπάνω αυτόματο μοντελοποιεί μία μοναδική εργασία. Για να μοντελοποιηθεί το όλο πρόβλημα το οποίο περιλαμβάνει n-εργασίες θα πρέπει να συντεθούν τα

αυτόματα όλων των εργασιών. Στη προσπάθεια αυτή το στοιχείο που θα πρέπει να προσεχθεί ιδιαίτερα είναι η τρίτη συνθήκη που αναφέρθηκε στα χαρακτηριστικά του προβλήματος, 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* όπου δύο ή περισσότερα αυτόματα βρίσκονται σε μια συγκεκριμένη κατάσταση στην οποία αντιστοιχεί ο ίδιος πόρος *m*. Σημειώνεται ότι στη θεωρία αυτομάτων, μια κατάσταση $q=(q^1,\ldots,q^n)\in Q^n$ θεωρείται αλληλοσυγκρουόμενη αν περιλαμβάνει δυο στοιχεία q^α και q^β τέτοια ώστε $q^\alpha=q^\beta=m\in M$.

Με δεδομένα τα παραπάνω προκειμένου να συντεθούν τα επιμέρους αυτόματα των εργασιών ώστε να παρασταθεί το πρόβλημα χρονοπρογραμματισμού, θα πρέπει να σχηματιστεί ένα χρονισμένο αυτόματο $A=(Q,C,\Delta,\mathrm{s},\mathrm{f})$ με *n* χρονιστές το οποίο θα περιλαμβάνει τα επιμέρους αυτόματα $A^i=(Q^i,\{c_i\},\Delta^i,s^i,f^i)$ της κάθε εργασίας. Προκειμένου να τηρηθεί ο περιορισμός της αμοιβαίας απόκλισης, στο αυτόματο $A=(Q,C,\Delta,\mathrm{s},\mathrm{f})$, 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.

Στο Σχήμα 35 παρουσιάζεται τμήμα των μεταβάσεων στο ολικό αυτόματο *A*. Όπως φαίνεται κάθε μεμονωμένη μετάβαση στο *A* αντιστοιχεί σε μια μετάβαση σε ένα και μόνο αυτόματο. 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:

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



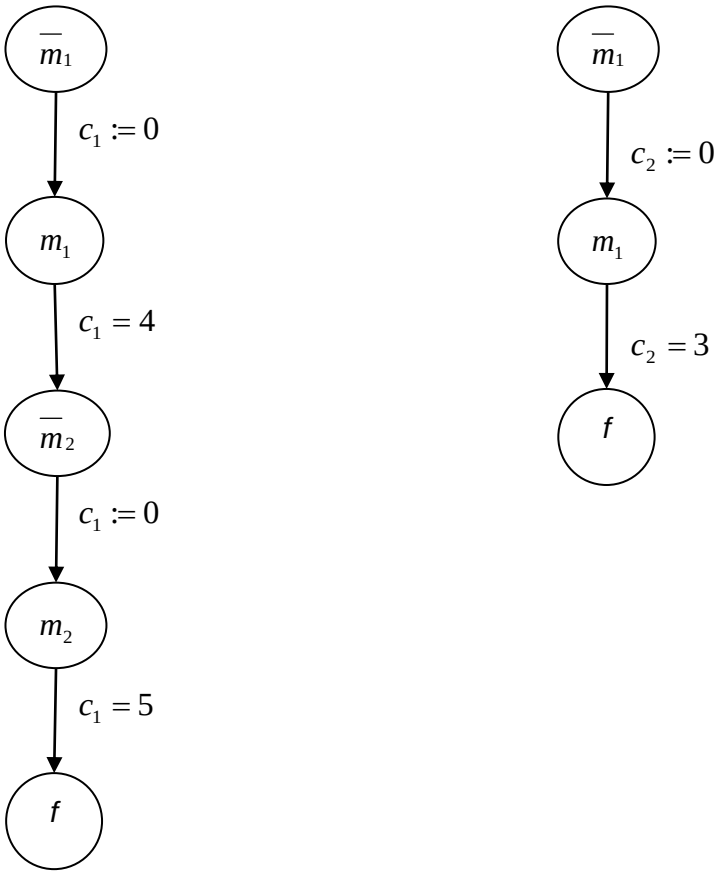
Σχήμα 35: Το χρονισμένο αυτόματο που αντιστοιχεί στα χαρακτηριστικά ενός Job-shop συστήματος $J = \{J^1, \ldots, J^n\}$

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 36 ενώ η σύνθεση και των δύο μαζί παρουσιάζεται στο Σχήμα 37.

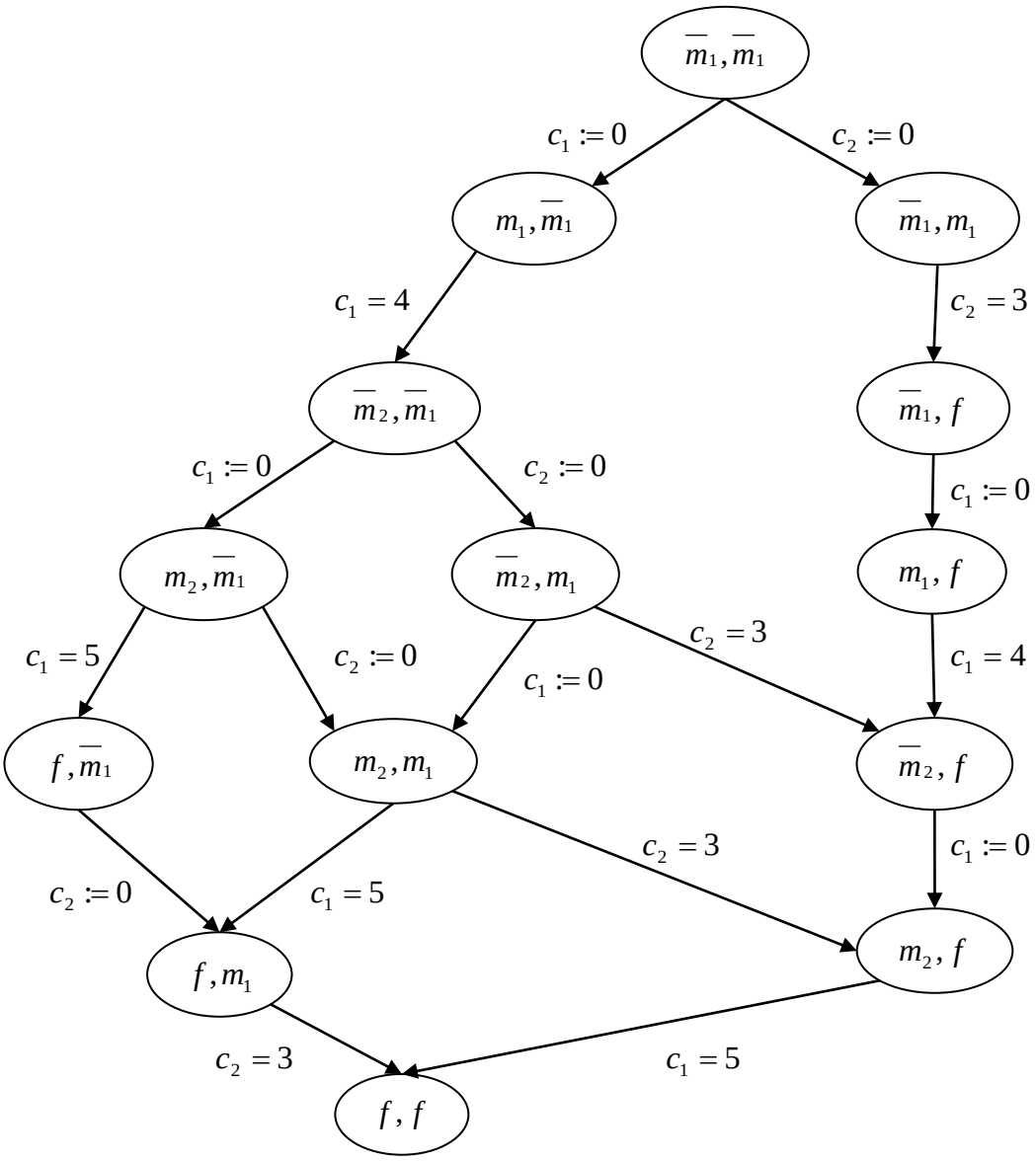
Το ολικό χρονισμένο αυτόματο που προκύπτει είναι μη κυκλικό. Περιλαμβάνει μια αρχική κατάσταση, στην οποία καμία από τις εργασίες δεν έχει ξεκινήσει και μια τελική κατάσταση όπου όλες οι εργασίες έχουν ολοκληρωθεί. Θεωρούμε ως *ολοκληρωμένη (complete)* την εκτέλεση ενός τέτοιου αυτομάτου αν ξεκινάει 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 (με παράλληλη επαναφορά-μηδενισμό του χρονιστή $c_i = 0$).



Σχήμα 36: Τα δύο αυτόματα που αντιστοιχούν στις εργασίες J^1 ***και*** J^2

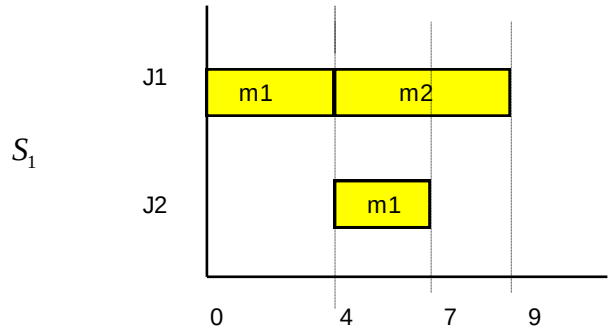
Δύο μεταβάσεις που εξέρχονται από την ίδια κατάσταση δείχνουν ότι ο χρονοπρογραμματιστής πρέπει να επιλέξει ποια από τις δύο θα ακολουθήσει. Συγκεκριμένα, στο παράδειγμά αυτό, οι δύο μεταβάσεις που εξέρχονται από την αρχική κατάσταση $(\overline{m}_1, \overline{m}_1)$ αναπαριστούν την απόφασή μας για 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 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 τον ελεγκτή/ 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.



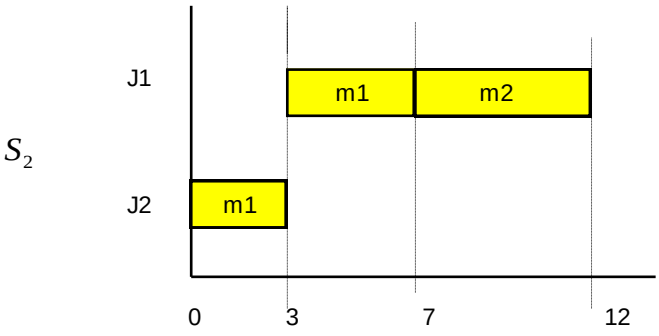
Σχήμα 37: Αναπαράσταση του ολικού αυτομάτου για τις εργασίες J^1 ***και*** J^2

Στα σχήματα που ακολουθούν παρουσιάζονται δύο τυχαίες εκτελέσεις (διαδρομές) του παραπάνω αυτομάτου (ξ_1 και ξ_2), καθώς και τα αντίστοιχα χρονοπρογράμματα που προκύπτουν βάσει αυτών (S_1 και S_2).

$$\xi_1: \begin{array}{l}(\overline{m}_1,\overline{m}_1,\perp,\perp)\xrightarrow{0}(m_1,\overline{m}_1,0,\perp)\xrightarrow{4}(m_1,\overline{m}_1,4,0)\xrightarrow{0}(\overline{m}_2,\overline{m}_1,\perp,\perp)\xrightarrow{0}\\ (m_2,\overline{m}_1,0,\perp)\xrightarrow{0}(m_2,m_1,0,0)\xrightarrow{3}(m_2,m_1,3,3)\xrightarrow{0}(m_2,f,3,\perp)\xrightarrow{2}\\ (m_2,f,5,\perp)\xrightarrow{0}(f,f,\perp,\perp)\end{array}$$



$$\xi_2: \begin{array}{l}(\overline{m}_1,\overline{m}_1,\perp,\perp)\xrightarrow{0}(\overline{m}_1,m_1,\perp,0)\xrightarrow{3}(\overline{m}_1,m_1,\perp,3)\xrightarrow{0}(m_1,f,\perp,\perp)\xrightarrow{0}\\ (m_1,f,0,\perp)\xrightarrow{4}(m_1,f,4,\perp)\xrightarrow{0}(\overline{m}_2,f,\perp,\perp)\xrightarrow{0}(m_2,f,0,\perp)\xrightarrow{5}\\ (m_2,f,,5,\perp)\xrightarrow{0}(f,f,\perp,\perp)\end{array}$$



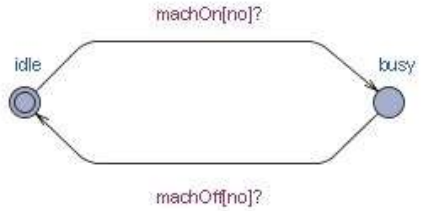
Σχήμα 38: Οι εκτελέσεις ξ_1 και ξ_2 και τα προκύπτοντα χρονοπρογράμματα

6.2.2. Υλοποίηση και έλεγχος μοντέλου

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, για το σχεδιασμό και τον έλεγχο των μοντέλων χρονισμένων αυτομάτων αξιοποιείται στην παρούσα διατριβή το εργαλείο UPPAAL. Σημειώνεται βέβαια ότι, πέρα από μικρές αλλαγές στη σύνταξη των εντολών και των περιορισμών, η μοντελοποίηση δε θα άλλαζε αν χρησιμοποιούνταν οποιοδήποτε άλλο από τα εργαλεία που υποστηρίζουν χρονισμένα αυτόματα. Εν τούτοις επιλέχθηκε το UPPAAL ως ένα εργαλείο το οποίο έχει αναπτυχθεί από δύο πανεπιστήμια και διατίθεται δωρεάν για ακαδημαϊκή χρήση, αλλά παράλληλα τα

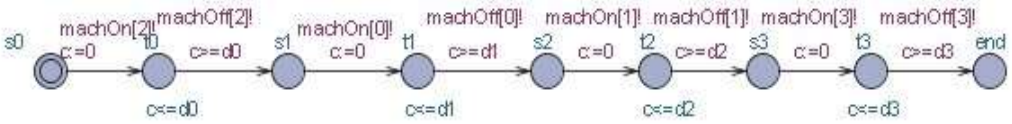
τελευταία 2 έτη έχει κυκλοφορήσει και δοκιμαστεί και σε εμπορικές/ επαγγελματικές εφαρμογές (UPPAAL commercial version).

Το βασικό μοντέλο που αναπτύχθηκε νωρίτερα μοντελοποιείται στο UPPAAL μέσω δύο προτύπων: του προτύπου που αναπαριστά τις μηχανές και αυτού που αναπαριστά τις εργασίες προς υλοποίηση.



Σχήμα 39: Βασικό πρότυπο μηχανής (Machine template)

Το πρότυπο της μηχανής έχει δύο καταστάσεις: την κατάσταση λειτουργίας και την κατάσταση αναμονής. Η μετάβαση από τη μία κατάσταση στην άλλη ελέγχεται μέσω δύο καναλιών συγχρονισμού (machOn και machOff) τα οποία ουσιαστικά συνθέτουν και συγχρονίζουν τα επιμέρους αυτόματα των εργασιών (τα οποία βασίζονται στο πρότυπο του σχήματος) με τα επιμέρους αυτόματα των μηχανών (τα οποία βασίζονται στο πρότυπο του σχήματος)



Σχήμα 40: Βασικό πρότυπο εργασίας (Job template)

Όπως φαίνεται και στο σχήμα του προτύπου, το αυτόματο κάθε εργασίας αποτελείται από ζευγάρια θέσεων (s_i, t_i). Οι θέσεις s_i υποδηλώνουν καταστάσεις στις οποίες η εργασία είναι αδρανής. Οι θέσεις αυτές δεν φέρουν κάποια σταθερά (invariant), δηλαδή δεν υπάρχει κανένας χρονικός ή άλλος περιορισμός για το χρόνο αναμονής πριν η εργασία προχωρήσει στην εκτέλεση του επόμενου βήματος (διεργασίας). Το χρονικό διάστημα που το αυτόματο θα παραμείνει σε αυτές εξαρτάται από τον ελεγκτή μοντέλου (model checker) μέσω του οποίου στο UPPAAL μπορούν να εφαρμοστούν αλγόριθμοι προσπελασιμότητας (όπως οι BFS και DFS που αναφέρθηκαν νωρίτερα). Προκειμένου ο ελεγκτής να πραγματοποιήσει την

μετάβαση από το s_i στο t_i , πρέπει να εξασφαλίσει πως ο συγχρονισμός μεταξύ των αυτομάτων που φέρουν την επιγραφή machOn[k], k∈0…n-1, όπου n είναι ο αριθμός των μηχανών, θα πραγματοποιηθεί. Πρόκειται για τα αυτόματα της εργασίας (job) και της μηχανής (machine). Όπως προαναφέραμε η κάθε πράξη συγχρονισμού προϋποθέτει και οι δύο εμπλεκόμενες ακμές να είναι ενεργοποιημένες. Αυτό συμβαίνει μόνο όταν το αυτόματο βρίσκεται σε θέση idle, οπότε η μηχανή είναι ανενεργή. Όταν τελικά πραγματοποιείται η μετάβαση, το εσωτερικό ρολόι c μηδενίζεται (reset) για να μετρήσει τον χρόνο παραμονής στη κατάσταση t_i. Όταν παρέλθει ο χρόνος παραμονής, το αυτόματο θα προχωρήσει με τη μετάβαση. Και αυτή επίσης φέρει επιγραφή συγχρονισμού (machOff[k]) που σημαίνει την απελευθέρωση της μηχανής από την τρέχουσα διεργασία. Ο συγχρονισμός μεταξύ των δύο αυτομάτων θα πραγματοποιηθεί, διότι η μηχανή δεν μπορεί να φύγει από τη κατάσταση busy με κάποιο άλλο τρόπο.

Με την παραπάνω μοντελοποίηση διασφαλίζονται και οι τρεις συνθήκες του γενικού προβλήματος που αναλύθηκαν στην παράγραφο 6.1.2:

- Διάταξη, δηλαδή τα βήματα της ίδιας εργασίας εκτελούνται με τη σειρά. Ισχύει διότι το αυτόματο της εργασίας θα διέλθει από τη θέση t_{i+1} μόνο αν έχει διέλθει προηγουμένως από τη t_i.
- Επικάλυψη και μη αντικατάσταση, δηλαδή κάθε διεργασία εκτελείται συνεχώς μέχρι την ολοκλήρωσή της. Ισχύει διότι δεν είναι δυνατόν το αυτόματο να εξέλθει από μια θέση t_i προτού περάσει το απαιτούμενο χρονικό διάστημα d_i. Για το ίδιο διάστημα η μηχανή είναι αφιερωμένη στη συγκεκριμένη εργασία και προφανώς δεν μπορεί να την αναλάβει κάποια άλλη.
- Αμοιβαίος αποκλεισμός, δηλαδή δύο διεργασίες διαφορετικών εργασιών που εκτελούνται την ίδια χρονική στιγμή δεν χρησιμοποιούν την ίδια μηχανή. Η συνθήκη ισχύει γιατί κανένα αυτόματο δε μπορεί να χρησιμοποιήσει τη μηχανή εφόσον είναι απασχολημένη (το αυτόματο της βρίσκεται στη θέση busy). Κάτι τέτοιο θα απαιτούσε να συγχρονιστούν τα αυτόματα της εργασίας και της μηχανής μέσω του καναλιού machOn[k], πράγμα που δε μπορεί να γίνει γιατί στη θέση busy η ακμή που φέρει την επιγραφή machOn[k] δεν είναι ενεργοποιημένη.

Έχοντας υλοποιήσει το παραπάνω μοντέλο και εισάγοντας κάποιες τυχαίες συνταγές εργασιών προκειμένου να γίνει η δοκιμή, επόμενο βήμα είναι ο έλεγχος ορθότητας του μοντέλου.

Το UPPAAL, όπως και κάθε αντίστοιχο λογισμικό ελέγχου μοντέλων, διαθέτει ένα ενσωματωμένο εργαλείο ελέγχου της ορθότητας το οποίο χρησιμοποιείται για να επιβεβαιώσει ότι η εκτέλεση του ολικού αυτομάτου είναι εφικτή και ότι είναι δυνατό να μεταβεί το σύστημα από την αρχική του κατάσταση (όπου καμία εργασία δεν έχει ξεκινήσει) σε μια τελική κατάσταση η οποία τίθεται ως κατάσταση-στόχος (όπου όλες οι εργασίες έχουν ολοκληρωθεί). Τα εργαλεία ελέγχου της ορθότητας των μοντέλων αυτό που κάνουν είναι να εφαρμόζουν κάποιο αλγόριθμο προσπελασιμότητας και μέσω αυτού να αναζητούν κάποια διαδρομή που να καταλήγει στην τελική κατάσταση στόχου. Όπως αναφέρθηκε στα προηγούμενα κεφάλαια οι δύο πιο δοκιμασμένοι και αποτελεσματικοί αλγόριθμοι αναζήτησης είναι ο αλγόριθμος BFS (Breadth First Search) και ο αλγόριθμος DFS (Depth First Search). Η διαφορά των δύο αυτών αλγορίθμων έγκειται στον τρόπο που κινούνται μέσα στο γράφο του ολικού αυτομάτου, δηλαδή στο ποια κατάσταση επιλέγουν να εξετάσουν ως επόμενη σε κάθε βήμα. Εν τούτοις και οι δύο αλγόριθμοι έχουν την ίδια λογική: ξεκινάνε από την αρχική κατάσταση και κινούνται μέσα στο δέντρο των προσπελάσιμων καταστάσεων εξετάζοντας διαδοχικά τις επόμενες καταστάσεις στις οποίες μπορεί να βρεθεί το σύστημα μέχρι να εντοπίσουν μια διαδρομή (σειρά καταστάσεων) που οδηγεί από την αρχική στην τελική κατάσταση. Καθώς στο συγκεκριμένο μοντέλο δεν έχουν τεθεί, μέχρι το σημείο αυτό, οποιοιδήποτε περιορισμοί που θα μπορούσαν να περιορίσουν τις μεταβάσεις από την αρχική στην τελική κατάσταση, ο έλεγχος ορθότητας αποβαίνει θετικός ανεξάρτητα από τις τιμές που έχουν τεθεί στο πρόβλημα και ανεξάρτητα από το ποιον από τους δύο αλγόριθμους εφαρμόζουμε.

Στο σημείο αυτό τονίζεται ότι οι αλγόριθμοί BFS και DFS δε δίνουν την πιο σύντομη διαδρομή. Απλά εντοπίζουν μία εφικτή διαδρομή (δηλαδή ένα εφικτό χρονοπρόγραμμα) το οποίο μπορεί να είναι ή να μην είναι το συντομότερο. Έτσι παρ' ότι η εφαρμογή τους έχει νόημα για τον έλεγχο ενός μοντέλου και το κατά πόσον είναι εφικτό να ολοκληρωθεί, δεν μπορούμε εφαρμόζοντάς τους να εντοπίσουμε το συντομότερο (υπό το πρίσμα του ταχύτερου χρόνου ολοκλήρωσης όλων των εργασιών) χρονοπρόγραμμα. Στην πράξη η χρησιμότητα τις εφαρμογής των αλγορίθμων αυτών στο συγκεκριμένο μοντέλο υφίσταται μόνο εφόσον τεθεί ένα όριο για το συνολικό χρόνο ολοκλήρωσης. Αν για παράδειγμα, επιστρέφοντας στο απλό

πρόβλημα της προηγούμενης παραγράφου υποθέσουμε ότι μας ενδιαφέρει να ελέγξουμε το κατά πόσον είναι εφικτό να ολοκληρώσουμε όλες τις εργασίες το πολύ σε 8 μονάδες χρόνου, αυτό μπορεί πολύ εύκολα να γίνει με την εφαρμογή των ανωτέρω αλγορίθμων αρκεί να προστεθεί στο μοντέλο η μεταβλητή του ολικού χρόνου ολοκλήρωσης.

Σε επίπεδο μοντέλου, αν στο χρονισμένο αυτόματο *A* προσθέσουμε ένα χρονιστή/ρολόι ***glc*** *(**gl**obal **c**lock)* που μετράει τον συνολικό χρόνο τότε θα προκύψει ένα 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:

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

και

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

Η τιμή του επιπρόσθετου χρονιστή *glc* 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.

Στο UPPAAL ο χρονιστής glc μπορεί να ενσωματωθεί σε κάθε ολικό αυτόματο με τον ορισμό του ως global clock. Προκειμένου να μπορέσουμε να αξιοποιήσουμε τους αλγορίθμους προσπελασιμότητας για τον έλεγχο της δυνατότητας ολοκλήρωσης πριν από κάποιο συγκεκριμένο χρονικό σημείο (για την περίπτωση αυτή οι 10 μονάδες χρόνου) αρκεί να ενσωματώσουμε στην τελική κατάσταση αναζήτησης την οριακή αυτή τιμή για την επιπλέον μεταβλητή του χρόνου. Έτσι ο αλγόριθμος που θα εφαρμοστεί θα έχει ως τελική κατάσταση την κατάσταση $(f,f,\perp,\perp,10)$. Εκτελώντας με τη βοήθεια του model checker του UPPAAL τόσο τον αλγόριθμο BFS όσο και τον DFS, παίρνουμε ως απάντηση ότι η ζητούμενη συνθήκη ικανοποιείται. Αυτό όμως δε συμβαίνει αν για παράδειγμα θέσουμε ως όριο χρόνου τις 7 μονάδες. Στην

περίπτωση αυτή κανείς αλγόριθμος δε μπορεί να βρει οποιαδήποτε διαδρομή που να οδηγεί από την αρχική κατάσταση $(\overline{m}_1, \overline{m}_1, \perp, \perp, 0)$ στην ζητούμενη τελική κατάσταση $(f, f, \perp, \perp, 7)$ καθώς εκ των πραγμάτων δεν υπάρχει τέτοιο χρονοπρόγραμμα. Στο σημείο αυτό εντοπίζεται μια **αρχική δυνατότητα αξιοποίησης της προτεινόμενης προσέγγισης σε ένα πραγματικό παραγωγικό περιβάλλον**. Με ένα πολύ απλό και γρήγορο τρόπο μπορεί να ελεγχθεί το κατά πόσον είναι εφικτό να ολοκληρωθεί μια σειρά εργασιών πριν από κάποιο χρονικό όριο που θέτει ένας υποψήφιος πελάτης. Έτσι η επιχείρηση μπορεί άμεσα να γνωρίζει κατά πόσον μπορεί να δεσμευτεί έναντι κάθε πελάτη της για την τήρηση της ζητούμενης ημερομηνίας παράδοσης, χωρίς καν να χρειάζεται υπολογισμός του συνολικού προγράμματος παραγωγής ο οποίος σε περιπτώσεις σύνθετων παραγωγικών συστημάτων μπορεί να είναι ιδιαίτερα χρονοβόρος.

6.2.3. Εύρεση βέλτιστου χρονοπρογράμματος

Όπως έχει αναλυθεί νωρίτερα, η λογική της μοντελοποίησης ενός προβλήματος χρονοπρογραμματισμού με τη χρήση χρονισμένων αυτομάτων επικεντρώνεται στη δημιουργία ενός μοντέλου το οποίο αναπαριστά τις πιθανές καταστάσεις του προβλήματος και στη συνέχεια στην εφαρμογή αλγορίθμων προσπέλασης στην κατεύθυνση της εξεύρεσης βέλτιστων (ή σε κάποιες περιπτώσεις υπο-βέλτιστων) λύσεων. Πριν προχωρήσουμε στον εμπλουτισμό του μοντέλου με στοιχεία που θα του επιτρέψουν να αναπαραστήσει σύνθετα προβλήματα χρονοπρογραμματισμού, θα δείξουμε βάσει της απλής περίπτωσης που αναλύθηκε στην προηγούμενη παράγραφο πώς, εκτός του να ελέγξουμε αν μπορεί να επιτευχθεί κάποιος χρονικός στόχος, μπορούμε να βρούμε τη συντομότερη διαδρομή του αυτομάτου η οποία ουσιαστικά παρέχει τη βέλτιστη λύση του προβλήματος για την περίπτωση που ο στόχος είναι η ελαχιστοποίηση του συνολικού χρόνου ολοκλήρωσης των εργασιών. Στις επόμενες παραγράφους το μοντέλο εμπλουτίζεται για την κάλυψη άλλων στόχων βελτιστοποίησης (εναλλακτικών αντικειμενικών συναρτήσεων).

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 μη κυκλικά χρονισμένα αυτόματα που παρουσιάστηκε στο Κεφάλαιο 3.

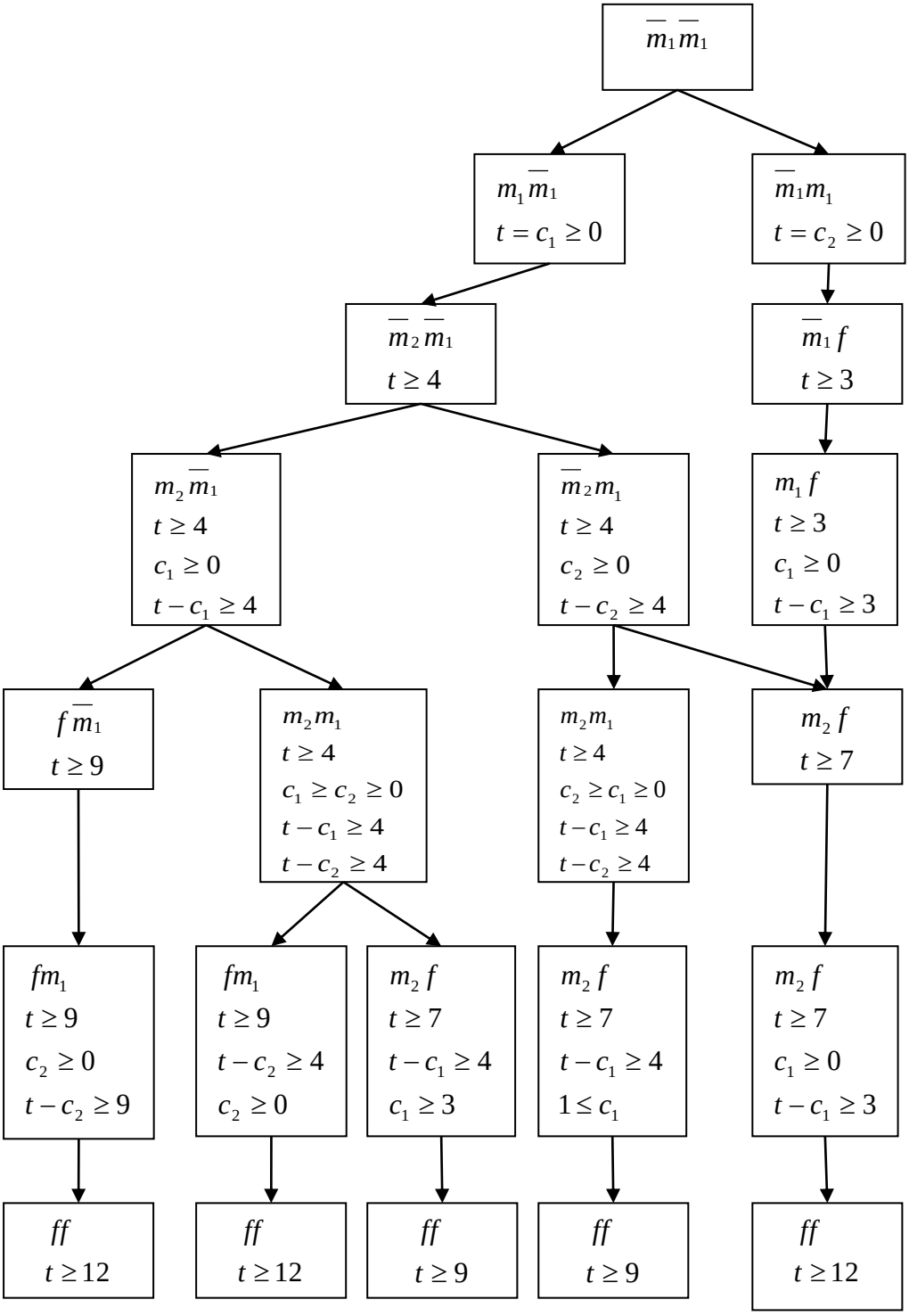
Στο Σχήμα 41 παρουσιάζεται το γράφημα προσομοίωσης του διευρυμένου αυτομάτου του Σχήματος 37. Από κάθε τελική συμβολική κατάσταση (f, Z) στο γράφημα προσομοίωσης μπορούμε να εξάγουμε τη συνάρτηση $G(f, Z)$ που δείχνει τη διάρκεια των ελαχίστων χρονικά εκτελέσεων δια μέσου όλων των εκτελέσεων που μοιράζονται 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^*, ενώ για την εύρεσή του απαιτείται πρακτικά να ελεγχθούν όλες οι διαδρομές που οδηγούν σε εφικτά χρονοπρογράμματα και εξ αυτών να επιλεγεί αυτή ή αυτές που δίνουν το ελάχιστο συνολικό χρόνο. Η παραπάνω διαδικασία μπορεί να πραγματοποιηθεί με τη χρήση του απλού αλγορίθμου ευθείας προσπέλασης που παρουσιάστηκε στο κεφάλαιο 3, ο οποίος εντοπίζει όλες τις δυνατές σειρές μεταβάσεων από την αρχική στην τελική κατάσταση και τελικά εξάγει τη διαδρομή με τον ελάχιστο συνολικό χρόνο ολοκλήρωσης. Σημειώνεται ότι ο συγκεκριμένος αλγόριθμος είναι ενσωματωμένος στο εργαλείο ελέγχου του UPPAAL, κάτι που επιτρέπει με ιδιαίτερη ευκολία να εντοπίσουμε τη διαδρομή με τον ελάχιστο συνολικό χρόνο (δηλ βάσει της μοντελοποίησης στο UPPALL με την ελάχιστη τιμή του χρονιστή/ρολογιού glc). Εν τούτοις, ενώ για μικρά προβλήματα (λίγων μηχανών και εργασιών) ο εντοπισμός όλων των δυνατών καταστάσεων είναι εφικτός, για μεγαλύτερα προβλήματα ο αριθμός αυτών αυξάνει εκθετικά με αποτέλεσμα ο υπολογιστικός χρόνος που απαιτείται να είναι μη αποδεκτός. Για το λόγο αυτό στη συνέχεια το αρχικό μοντέλο εξελίσσεται προκειμένου η αποτελεσματικότητα του αλγορίθμου αναζήτησης των εφικτών μεταβάσεων να βελτιωθεί.



Σχήμα 41: Το γράφημα προσομοίωσης του ολικού χρονισμένου αυτομάτου του Σχήματος 37

6.3. Προσθήκη και έλεγχος άνω ορίου

Όπως αναφέρθηκε προηγούμενα, το βασικό μοντέλο που αναπτύχθηκε υπολογίζει όλες τις πιθανές διαδρομές μέχρι το τέλος τους. Συγκεκριμένα ως τέλος του συστήματος στη προκειμένη περίπτωση θεωρείται η κατάσταση όπου όλα τα αυτόματα των εργασιών έχουν φτάσει στις θέσεις end (βλ. Σχήμα 40). Προκειμένου να μειωθεί ο υπολογιστικός φόρτος και να είναι εφικτή η εφαρμογή της προτεινόμενης προσέγγισης σε μεγάλα προβλήματα, υπάρχουν δύο ζητήματα:

- Πώς διακρίνουμε τις διαδρομές που μπορούμε εκ των προτέρων να ξέρουμε ότι δεν είναι βέλτιστες.
- Πώς οδηγούμε τον αλγόριθμο αναζήτησης να αναγνωρίσει τις διαδρομές αυτές και άρα να σταματήσει νωρίτερα την εξερεύνησή τους χωρίς να φτάσει μέχρι τέλους.

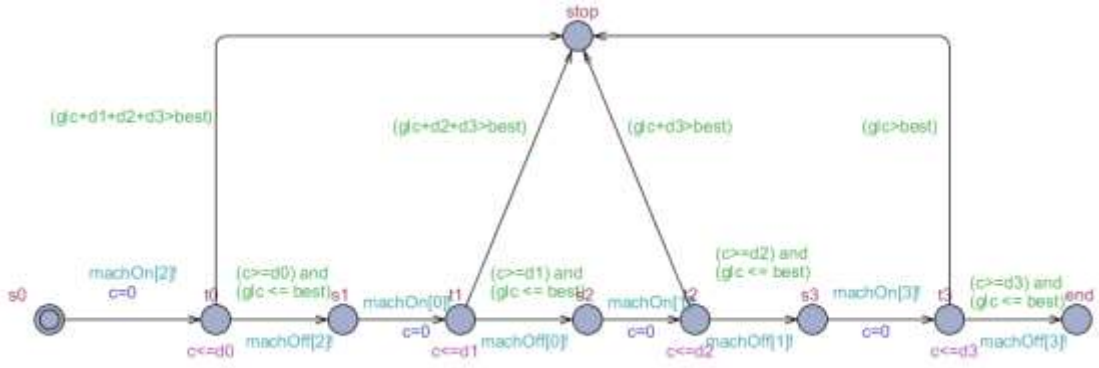
Μια πρώτη προσέγγιση βελτίωσης του βασικού μοντέλου είναι ο καθορισμός ενός άνω ορίου για το συνολικό χρόνο ολοκλήρωσης των εργασιών. Το άνω όριο μπορεί να προκύπτει είναι εμπειρικά, είτε να δίνεται από το ίδιο το πρόβλημα (πχ αν ο χρονοπρογραμματισμός αναφέρεται σε μεσοπρόθεσμο προγραμματισμό επιπέδου μίας εβδομάδας, τότε κάθε χρονοπρόγραμμα το οποίο υπερβαίνει το διάστημα αυτό απορρίπτεται), είτε να έχει προκύψει μέσα από την εφαρμογή κάποιου αλγορίθμου προσπέλασης, όπως οι BFS, DFS που έχουν αναλυθεί νωρίτερα κατά τη διάρκεια του ελέγχου ορθότητας του συστήματος. Σε κάθε περίπτωση θα πρέπει το μοντέλο να προσαρμοστεί προκειμένου ο αλγόριθμός να απορρίπτει απευθείας την τρέχουσα διαδρομή την πρώτη στιγμή που η διάρκεια της υπερβαίνει το καθορισμένο όριο. Με άξονα την παραπάνω ιδέα καταλήξαμε σε ένα δεύτερο μοντέλο, βελτιώνοντας το πρώτο με την προσθήκη συνθηκών ελέγχου και τερματισμού στο αυτόματο των εργασιών. Συγκεκριμένα στο πρότυπο των εργασιών και σε κάθε κόμβο στον οποίο εισέρχεται το αυτόματο μετά την ολοκλήρωση μιας διεργασίας προστέθηκε ο εξής έλεγχος:

Αν οποιαδήποτε στιγμή το άθροισμα της τιμής του ολικού χρονιστή (glc) και των διαρκειών των διεργασιών της εργασίας που δεν έχουν ολοκληρωθεί υπερβαίνει το άνω όριο το οποίο έχει καθοριστεί, τότε η ελεγχόμενη διαδρομή απορρίπτεται απευθείας και ο έλεγχός της σταματάει άμεσα.

Προκειμένου να επιτευχθεί αυτό, προστέθηκε η κατάσταση *stop* στην οποία μεταβαίνει το αυτόματο εφόσον κάποια στιγμή ισχύσουν οι παραπάνω

προϋποθέσεις. Από την κατάσταση αυτή, όπως είναι προφανές στο νέο μοντέλο, το αυτόματο είναι αδύνατο να φύγει και άρα να μεταβεί στην τελική κατάσταση *end*. Έτσι ο αλγόριθμος θεωρεί ότι η τρέχουσα διαδρομή είναι μη εφικτή και απευθείας προχωράει στον έλεγχο άλλων εναλλακτικών επιτυγχάνοντας σημαντική μείωση του συνολικού αριθμού ελεγχόμενων καταστάσεων.

Προκειμένου να γίνει σαφής η λογική του προτεινόμενου ελέγχου, θεωρούμε το αυτόματο μιας εργασίας όπως φαίνεται στο Σχήμα 42. Σε σχέση με το αντίστοιχο αυτόματο του πρώτου μοντέλου έχει προστεθεί η θέση *stop* και οι έλεγχοι που σχετίζονται με τη μετάβαση του αυτομάτου στη θέση αυτή.



Σχήμα 42: Αυτόματο εργασίας με ελέγχους άνω ορίου

Δεδομένου ότι κάθε εργασία αποτελείται από μια σειρά διαδοχικών διεργασιών, κάθε δεδομένη στιγμή ο χρόνος που απομένει μέχρι την ολοκλήρωση της εργασίας είναι κατ' ελάχιστο ίσος με το άθροισμα των διαρκειών των διεργασιών που έπονται της τρέχουσας. Για παράδειγμα μετά την ολοκλήρωση της διεργασίας 1, η εργασία, στη ιδανική περίπτωση κατά την οποία δεν υπάρχουν καθόλου αναμονές, θα χρειάζεται πρόσθετο χρόνο ολοκλήρωσης ίσο με το άθροισμα των διαρκειών των διεργασιών 2 και 3. Με τον έλεγχο που προστέθηκε, το σύστημα ελέγχοντας το άθροισμα της τιμής του καθολικού χρονιστή glc (που δείχνει πόσος χρόνος έχει περάσει από την αρχή των εργασιών) και των διαρκειών των εναπομενουσών διεργασιών, μπορεί και ξέρει εκ των προτέρων αν η συγκεκριμένη διαδρομή θα ξεπεράσει σε χρόνο το άνω όριο το οποίο έχει τεθεί (ή υπολογιστεί). Εφόσον αυτό συμβεί, κάποιο εκ των αυτομάτων των εργασιών θα μεταβεί στη θέση stop από την οποία η περαιτέρω μετάβαση στη θέση end είναι αδύνατη. Έτσι ο αλγόριθμος θα θεωρήσει τη διαδρομή μη αποδεκτή και χωρίς να σπαταληθεί υπολογιστικός χρόνος θα απορριφθεί άμεσα. Με τον τρόπο

αυτό μειώνεται δραστικά ο αριθμός των ελεγχόμενων καταστάσεων του αλγορίθμου ευθείας προσπέλασης και επιταχύνεται σημαντικά η εύρεση λύσης.

6.4. Αφαίρεση αδρανών χρονοπρογραμμάτων

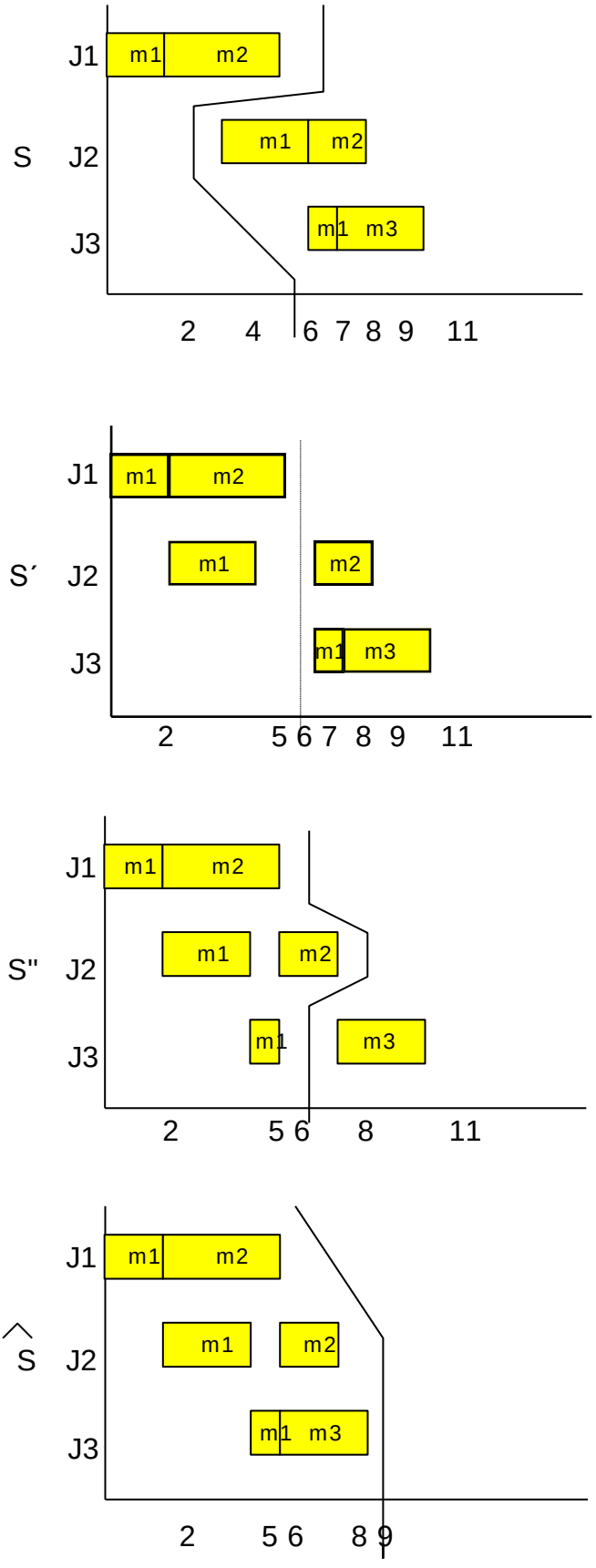
6.4.1. Αδρανή και άμεσα χρονοπρογράμματα

Έστω *S* είναι ένα χρονοπρόγραμμα, *i* είναι μια εργασία και *j* είναι μια διεργασία αυτής της εργασίας με $\mu^i(j) = m$. Ένα το χρονοπρόγραμμα *S* χαρακτηρίζεται ως αδρανές στο (*i*,*j*) αν υπάρχει στο *S* κάποιο διάστημα $[t,(i,j)]$ στο οποίο το (*i*,*j*) να μπορεί να εκτελεστεί.

Η αδρανοποίηση θεωρείται ουσιαστικά μια αχρείαστη αναμονή που σημαίνει ότι το βήμα μιας εργασίας που θα μπορούσε να εκτελεστεί δεν εκτελείται αλλά βρίσκεται σε κατάσταση αναμονής χωρίς όμως κάποια άλλη εργασία να επωφελείται από αυτή την αναμονή.

Στο Σχήμα 43 παρουσιάζεται ένα χρονοπρόγραμμα S όπου όπως φαίνεται υπάρχει αδρανοποίηση στο βήμα (2,1) δηλαδή της εκτέλεσης της εργασίας 2 στη μηχανή 1.

Για την αποφυγή της αδράνειας πρέπει να μετακινήσουμε τη διεργασία (2,1) ώστε να εκτελεστεί νωρίτερα. Έτσι λαμβάνουμε χρονοπρόγραμμα S' στο οποίο παρατηρούμε ότι προκύπτει αχρείαστη καθυστέρηση για τις διεργασίες (2,2) και (3,1). Αντίστοιχα μετακινώντας τις διεργασίες (2,2) και (3,1) νωρίτερα παίρνουμε το S'', στο οποίο επίσης προκύπτει περιττή αναμονή στη διεργασία (3,2). Η μετακίνηση και αυτής της εργασίας νωρίτερα μας οδηγεί στο χρονοπρόγραμμα $\widehat{S}$ το οποίο είναι μη-αδρανές.



Σχήμα 43: Μετακίνηση της αδρανοποίησης από το χρονοπρόγραμμα

Τα μοντέλα αυτομάτων που παρουσιάστηκαν νωρίτερα, παρά το γεγονός ότι μοντελοποιούν εύστοχα το πρόβλημα του χρονοπρογραμματισμού οδηγούν τον αλγόριθμο προσπέλασης στον έλεγχο πολλών διαφορετικών καταστάσεων. Ένας βασικός λόγος είναι ότι τόσο τα μοντέλα όσο και ο αλγόριθμος δεν αποκλείουν τα αδρανή χρονοπρογράμματα πάρα το γεγονός οι βέλτιστες λύσεις δε μπορεί παρά να αφορούν μόνο μη-αδρανή χρονοπρογράμματα. Αυτό συμβαίνει γιατί σύμφωνα με τη θεωρία αυτομάτων κάθε χρονοπρόγραμμα με αδράνεια μετασχηματίζεται σε ένα νέο μη αδρανές χρονοπρόγραμμα (με απλή μετακίνηση των αδρανών εργασιών) το οποίο έχει ίσο ή μικρότερο χρόνο ολοκλήρωσης από το αρχικό. Για το λόγο αυτό ένα βέλτιστο χρονοπρόγραμμα είναι πάντοτε μη-αδρανές. Αυτό σημαίνει ότι θα πρέπει να εστιάσουμε την προσοχή μας στις εκτελέσεις του αυτομάτου στις οποίες αντιστοιχούν χρονοπρογράμματα τα οποία είναι μη αδρανή. Στο σημείο αυτό θα πρέπει να ορίσουμε την έννοια των άμεσων εκτελέσεων, διότι όπως θα δούμε, οι μη-αδρανής εκτελέσεις είναι παράλληλα και άμεσες εκτελέσεις. Μια εκτέλεση θεωρείται ως άμεση όταν όλες οι μεταβάσεις πραγματοποιούνται το συντομότερο που θα μπορούσαν να πραγματοποιηθούν. Μια μη άμεση εκτέλεση περιέχει ένα «θραύσμα» (fragment):

$$(q,v) \xrightarrow{t} (q,v+t) \xrightarrow{0} (q',v')$$

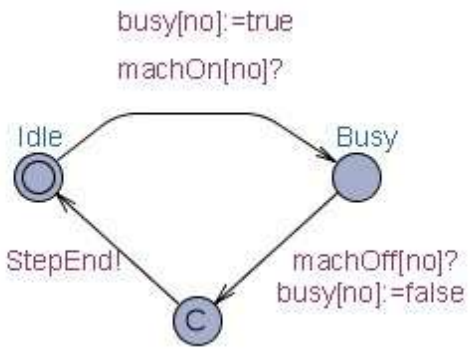
όπου η μετάβαση που λαμβάνεται στη $(q,v+t)$ θα μπορούσε να ληφθεί νωρίτερα στη $(q,v+t')$ με $t' < t$.

Το συμπέρασμα που βγαίνει είναι ότι το χρονοπρόγραμμα που προκύπτει από μια μη άμεση εκτέλεση επιδεικνύει αδρανοποίηση. Από τα παραπάνω προκύπτει το συμπέρασμα ότι για την εύρεση του βέλτιστου χρονοπρογράμματος αρκεί να εξετάζεται μόνο το (πεπερασμένο) σύνολο των άμεσων εκτελέσεων και όχι όλες οι δυνατές εκτελέσεις.

Στην κατεύθυνση αυτή το μοντέλο των αυτομάτων το οποίο αναπτύχθηκε νωρίτερα μπορεί να μετασχηματιστεί σε ένα μοντέλο ακόμα μεγαλύτερης αποτελεσματικότητας το οποίο δε θα επιτρέπει στον αλγόριθμο αναζήτησης να εξετάσει μη άμεσα χρονοπρογράμματα.

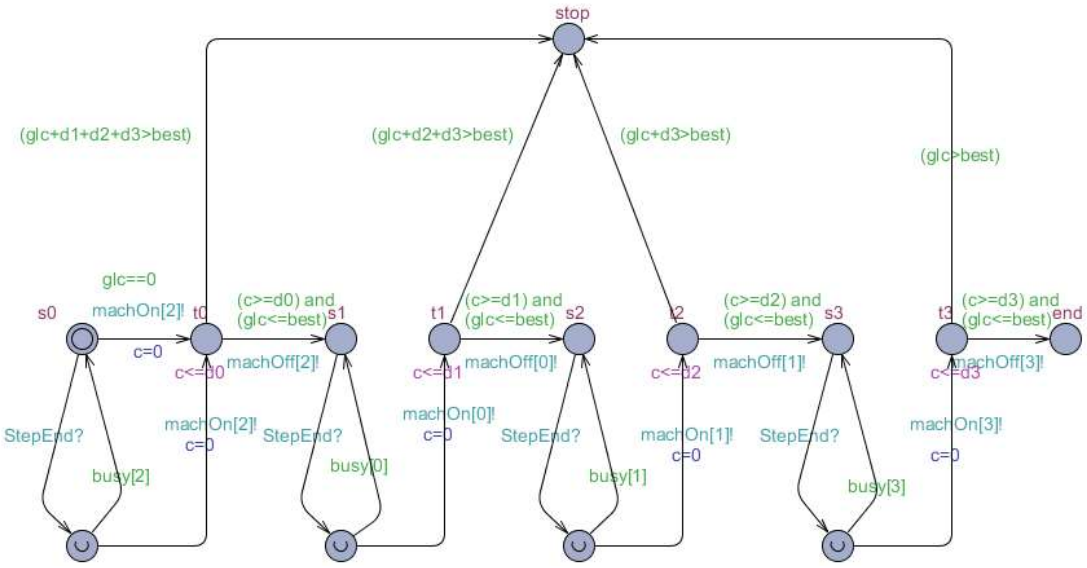
Όπως αναλύθηκε παραπάνω, οι άμεσες εκτελέσεις απαλλάσσουν τον προγραμματισμό από «νεκρά» διαστήματα που ενδεχομένως μεσολαβούν μεταξύ δύο διακεκριμένων θέσεων του συνολικού αυτομάτου. Αυτό σημαίνει ότι μια εργασία επιτρέπεται να συνεχίσει με την επόμενη διεργασία της την χρονική στιγμή του τερματισμού μιας άλλης διεργασίας και όχι απαραίτητα της ίδιας εργασίας. Βεβαίως επιτρέπονται τα κενά διαστήματα μεταξύ διεργασιών της ίδιας εργασίας, αλλά όχι αυτά όπου όλες οι εργασίες (και προφανώς οι μηχανές) είναι ανενεργές. Στην κατεύθυνση αυτή κατασκευάστηκε ένα νέο μοντέλο, ως επέκταση του προηγούμενου, με τις εξής διαφοροποιήσεις:

i. **Στο αυτόματο των μηχανών:** εδώ επιθυμούμε κάθε μηχανή να ειδοποιεί τα αυτόματα όλων των εργασιών πως έχει τελειώσει με την διεργασία που ανέλαβε, οπότε εκείνες, αν είναι ανενεργές, μπορούν να ξεκινήσουν τις επόμενες διεργασίες τους (ή φυσικά να παραμείνουν ανενεργές). Όταν πραγματοποιείται ο συγχρονισμός machOff μεταξύ μηχανής και εργασίας, θέλουμε να επιτυγχάνεται και συγχρονισμός μεταξύ αυτής της μηχανής και των αδρανών εργασιών. Επειδή μια μετάβαση δε μπορεί να φέρει πολλαπλές επιγραφές συγχρονισμού, προστέθηκε μια θέση, ενδιάμεση της busy και της idle. Η θέση δηλώθηκε committed έτσι ώστε μόλις το αυτόματο φτάσει σ' αυτή να ενεργοποιηθεί αμέσως και χωρίς καθυστέρηση η μετάβαση που εξέρχεται απ' αυτή και καταλήγει στην idle. Δηλαδή οι δύο μεταβάσεις εκτελούνται διαδοχικά την ίδια χρονική στιγμή. Το νέο αυτόματο της μηχανής απεικονίζεται στο παρακάτω σχήμα:



Σχήμα 44 - Αυτόματο μηχανής

ii. **Στο αυτόματο των εργασιών:** Μια εργασία αποχωρεί από τη θέση s_i όταν γίνει συγχρονισμός με μια μηχανή που επιστρέφει στη θέση idle. Μεταβαίνει σε μια θέση στην οποία ο model checker έχει δύο επιλογές. Είτε επιστρέφει στη θέση s_i όπου και περιμένει μέχρι τον επόμενο τερματισμό διεργασίας είτε προχωρά στην επόμενη διεργασία. Η θέση δηλώθηκε urgent, έτσι ώστε το αυτόματο να μη καθυστερήσει στη λήψη της απόφασης και να αποχωρήσει την ίδια χρονική στιγμή. Έχει τη δυνατότητα να μεταβεί στην επόμενη διεργασία εφ' όσον, όπως και προηγουμένως, είναι δυνατός ο συγχρονισμός του με την αντίστοιχη μηχανή, ενώ η μετάβαση θα πραγματοποιηθεί σίγουρα αν η επικείμενη διεργασία είναι η τελευταία που πρόκειται να αναλάβει η μηχανή ή καμία άλλη εργασία δεν είναι ενεργή. Στη πρώτη περίπτωση δεν υπάρχει λόγος η εργασία να περιμένει, ενώ η δεύτερη είναι κλασική περίπτωση μη άμεσης εκτέλεσης. Επίσης το αυτόματο ξεκινά την πρώτη του διεργασία είτε τη χρονική στιγμή 0 (για να αποφύγουμε μια άσκοπη χρονική καθυστέρηση) είτε όταν τελειώσει κάποια διεργασία (περίπτωση όπου τη χρονική στιγμή 0 η μηχανή της πρώτης διεργασίας καταλήφθηκε από άλλη εργασία). Στο ακόλουθο σχήμα παρουσιάζεται το επικαιροποιημένο αυτόματο μιας εργασίας:

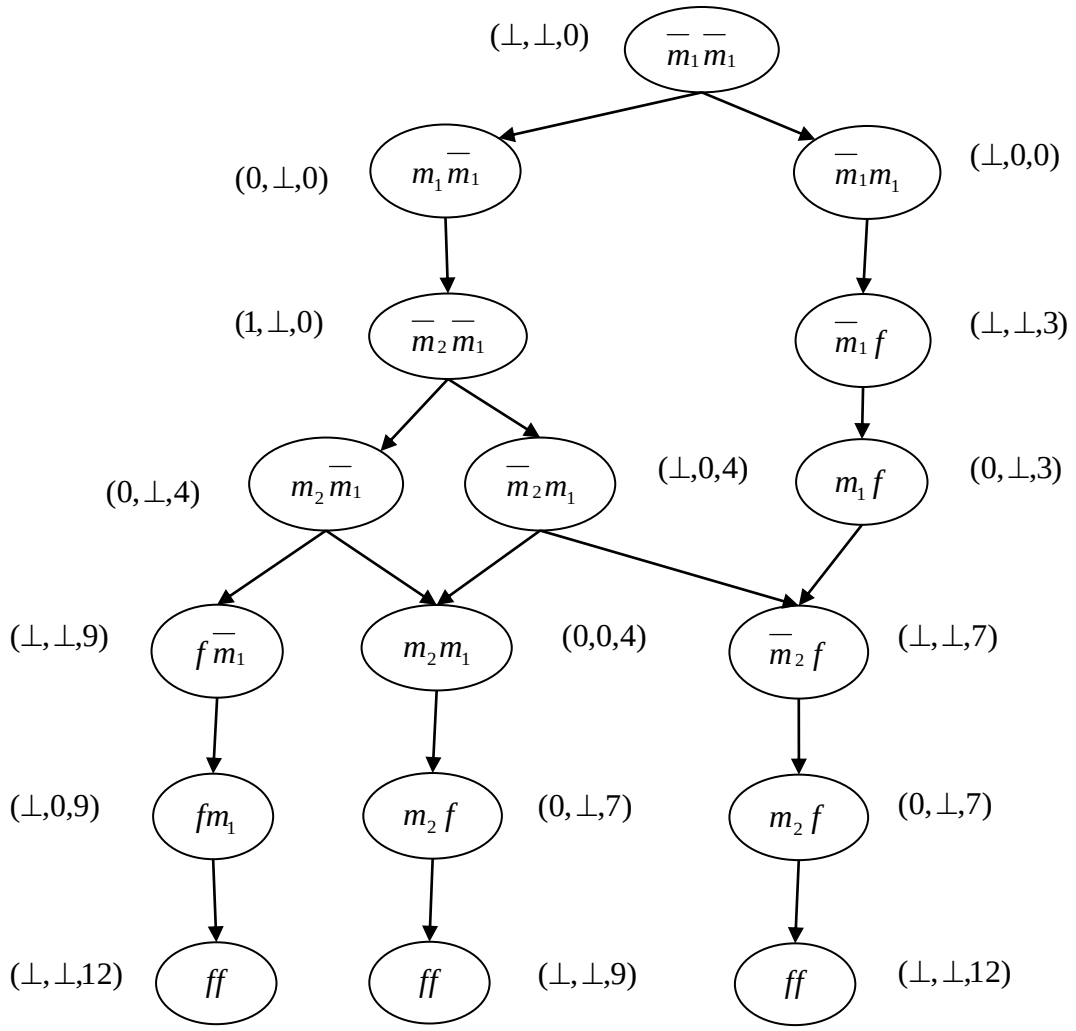


Σχήμα 45 - Αυτόματο εργασίας

iii. **Στις δηλώσεις των μεταβλητών**: Προστέθηκαν δηλώσεις μεταβλητών γενικής εμβέλειας. Για την ειδοποίηση των εργασιών σχετικά με το τέλος μιας διεργασίας, δηλώθηκε το κανάλι StepEnd ως broadcast διότι οι

αποδέκτες του συγχρονισμού είναι δυνητικά πολλοί. Ο boolean πίνακας busy περιέχει μεταβλητές που υποδηλώνουν αν μια μηχανή είναι απασχολημένη.

Προκειμένου να γίνει κατανοητό το όφελος από την αφαίρεση της αδράνειας, ακολούθως παρουσιάζονται για το αυτόματο του Σχήματος 37 μόνο οι άμεσες εκτελέσεις του:



Σχήμα 46: Οι άμεσες εκτελέσεις του χρονισμένου αυτομάτου του Σχήματος 37

6.5. Διαδικασία επίλυσης

6.5.1. Περιορισμός εύρους αναζήτησης

Στις προηγούμενες παραγράφους αναπτύχθηκε το βασικό μοντέλο επίλυσης προβλημάτων χρονικού προγραμματισμού με αυτόματα το οποίο στη συνέχεια θα εμπλουτιστεί και θα επεκταθεί προκειμένου να καλύψει ένα μεγάλο φάσμα προβλημάτων του πραγματικού κόσμου. Συγκεκριμένα αναπτύχθηκαν τρία μοντέλα, ξεκινώντας από το απλό μοντέλο το οποίο προέκυψε μέσα από ανάλυση της σχετικής βιβλιογραφίας και προηγούμενων εργασιών που έχουν δημοσιευθεί σχετικά με την αξιοποίηση των χρονισμένων αυτομάτων στον προγραμματισμό διαφόρων καταστάσεων. Παρά το γεγονός ότι το αρχικό μοντέλο είναι ορθό και μπορεί να δώσει αποτέλεσμα, η δομή του είναι τέτοια που ουσιαστικά αναγκάζει τον αλγόριθμο αναζήτησης λύσεων να εξετάσει το σύνολο των πιθανών διαδρομών και εξ αυτών να εντοπίσει αυτή με τον ταχύτερο χρόνο ολοκλήρωσης.

Ενώ, όμως, η παραπάνω διαδικασία είναι εφικτή για περιπτώσεις «μικρών» προβλημάτων όπου ο αριθμός των πιθανών λύσεων είναι μετρήσιμος, το μοντέλο αυτό αδυνατεί να δώσει αποτέλεσμα ακόμα και σε μεσαίου μεγέθους προβλήματα (όπως προβλήματα διαστάσεων 5x5, 6x6 και 5x10 που συχνά συναντώνται στη βιβλιογραφία). Αυτό είναι κάτι αναμενόμενο καθώς προβλήματα χρονοπρογραμματισμού είναι από τη φύση τους μη πολυωνυμικα (NP-Hard) δηλαδή πρακτικά δε μπορούν να λυθούν σε αποδεκτό χρόνο μέσω πλήρους απαρίθμηση όλων των εναλλακτικών χρονοπρογραμμάτων.

Με βάση τα παραπάνω, προχωρήσαμε στην εξέλιξη του αρχικού μοντέλου περιορίζοντας τον αριθμό των καταστάσεων προς εξερεύνηση. Έτσι σχηματίσαμε ένα δεύτερο μοντέλο θέτοντας άνω όρια στους χρόνους των εργασιών και απορρίπτοντας νωρίς τις καταστάσεις που οδηγούν σε υπέρβαση των ορίων αυτών. Όπως φαίνεται από τα πειραματικά αποτελέσματα που παρουσιάζονται παρακάτω, υπήρξε με τον τρόπο αυτό σχετικός περιορισμός του αριθμού των ελεγχόμενων καταστάσεων από τον αλγόριθμο αναζήτησης και αντίστοιχη μείωση του χρόνου υπολογισμού της ελάχιστης διαδρομής. Εν τούτοις και πάλι ο αριθμός των καταστάσεων που ελέγχονταν ήταν ιδιαίτερα μεγάλος με αποτέλεσμα ναι μεν να παράγεται αποτέλεσμα γρηγορότερα για «μικρά» προβλήματα αλλά και πάλι το

σύστημα να αδυνατεί να δώσει λύση σε πολυωνυμικό χρόνο σε πιο μεγάλα προβλήματα.

Μετά από μελέτη του στοιχείου που αυξάνει τον αριθμό των ελεγχόμενων καταστάσεων, διαπιστώθηκε ότι αυτές θα μπορούσαν να περιοριστούν δραστικά με το να αποκλειστούν όλες οι μη-αμεσες μεταβάσεις προκειμένου να αποκλειστεί η πλειονότητα των αδρανών περιπτώσεων. Βάσει αυτής της λογικής καταλήξαμε σε ένα τρίτο μοντέλο το οποίο στη συνέχεια της εργασίας θα χρησιμοποιηθεί ως το βασικό μοντέλο πάνω στο οποίο θα χτιστούν οι προσεγγίσεις αντιμετώπισης σύνθετων προβλημάτων χρονοπρογραμματισμού. Οι πειραματικές δοκιμές απέδειξαν ότι το μοντέλο αυτό περιορίζει περαιτέρω το χώρο των λύσεων και άρα τον αριθμό των εξεταζόμενων καταστάσεων καθιστώντας το ικανό να επιλύσει προβλήματα με μεγαλύτερη αποτελεσματικότητα.

Έχοντας ως δεδομένο τον περιορισμό του εύρους των ελεγχόμενων καταστάσεων που προκύπτει από την υιοθέτηση του τελευταίου μοντέλου, εξετάστηκε στη συνέχεια η περίπτωση της αξιοποίησης κάποιου αλγορίθμου αναζήτησης με μεγαλύτερη αποτελεσματικότητα από ότι ο αλγόριθμος ευθείας προσπέλασης.

Δεδομένου άλλωστε ότι η ανάγκη της υιοθέτησης πιο «έξυπνων» προσεγγίσεων οι οποίες δεν εξετάζουν όλο το εύρος των πιθανών λύσεων αλλά κινούνται μέσα στο χώρο λύσεων με συγκεκριμένο τρόπο είναι κοινός τόπος για όλα τα NP-Hard προβλήματα, το πρόβλημα του χρονοπρογραμματισμού των εργασιών, το οποίο από τη φύση του έχει μεγάλη πολυπλοκότητα, αποτελεί χαρακτηριστική τέτοια περίπτωση.

Στο πλαίσιο αυτό εξετάστηκε μια βασική παραλλαγή του αλγορίθμου ευθείας προσπέλασης η οποία έχει την εξής διαφορά: αφού βρεθεί μια πρώτη λύση του προβλήματος, η τιμή του συνολικού χρόνου αποθηκεύεται σε μια μεταβλητή «best». Καθώς ο αλγόριθμος εκτελείται ελέγχεται σε κάθε κατάσταση αν ο χρόνος που έχει διέλθει από την αρχική κατάσταση μέχρι την τρέχουσα είναι μεγαλύτερος του best. Αν είναι τότε όλες οι καταστάσεις που πηγάζουν από την πρώτη απορρίπτονται (δεν εξετάζονται). Αν από την άλλη ο χρόνος είναι μικρότερος, τότε ο αλγόριθμος προχωράει σε βάθος στις επόμενες καταστάσεις, κάνοντας κάθε φορά τον παραπάνω έλεγχο. Σε περίπτωση που βρεθεί κάποια διαδρομή που έχει συνολικό χρόνο μικρότερο του προηγούμενου βέλτιστου, τότε η μεταβλητή best παίρνει αυτή την τιμή και ο αλγόριθμος συνεχίζεται μέχρι να εξεταστούν ή να απορριφθούν όλες οι

δυνατές καταστάσεις. Σημειώνεται ότι αυτός ο «βελτιστοποιημένος» αλγόριθμος ευθείας προσπέλασης έχει το πλεονέκτημα ότι τα τμήματα του εύρους καταστάσεων τα οποία δεν εξετάζει (μειώνοντας έτσι το χρόνο αναζήτησης) είναι 100% βέβαιο ότι δεν περιλαμβάνουν τη βέλτιστη λύση. Συμβολικά ο βελτιστοποιημένος αλγόριθμος μπορεί να αναπαρασταθεί ως εξής:

Βελτιστοποιημένος αλγόριθμος ευθείας προσπέλασης για εύρεση ελάχιστου χρόνου

$Waiting := \{Succ(s,v,t)\};$
$Best := \infty$
$Pick(q,v,t) \in Waiting;$
while $Waiting \neq \varnothing$
do
  $For\ every\ (q',v',t') \in Succ(q,v,t);$
    if $q' = f$ **and** $t' < Best$ **then** $Best = t';$
    if $t < Best$ **then** $Insert\ (q',v',t')\ into\ Waiting;$
$Remove\ (q,v,t)\ from\ Waiting;$
end

Όπως είναι προφανές ο βελτιστοποιημένος αλγόριθμος διατηρεί το πλεονέκτημα ελέγχου όλου του εύρους λύσεων (με άμεσο ή έμμεστο τρόπο) και άρα δίνει σε κάθε περίπτωση το σωστό αποτέλεσμα – αποκλείεται δηλαδή να καταλήξει σε υπο-βέλτιστη λύση. Από την άλλη δίνει τη δυνατότητα να ελεγχθεί το εύρος λύσεων για προβλήματα που με απαρίθμηση όλων των δυνατών καταστάσεων χωρίς αποφυγή (απόρριψη) τμημάτων του εύρους λύσεων θα ήταν αδύνατο να υπολογιστούν σε αποδεκτό χρόνο.

Παρά το γεγονός ότι ο παραπάνω αλγόριθμος βελτιώνει το χρόνο αναζήτησης χωρίς χρήση ευρετικών μεθόδων και επιτρέπει να λυθούν προβλήματα μεσαίου μεγέθους, εξακολουθεί και αυτός σε μεγάλα προβλήματα να αδυνατεί να δώσει γρήγορη λύση σε αποδεκτό χρόνο καθώς ο αριθμός των κλάδων στους οποίους προχωράει την αναζήτηση εξακολουθεί να είναι μεγάλος. Για το λόγο αυτό και προκειμένου να μπορέσουμε να αντιμετωπίσουμε πιο σύνθετα προβλήματα αναζητήθηκε μια προσέγγιση που θα περιόριζε περαιτέρω το εύρος των λύσεων. Ανατρέχοντας στις εναλλακτικές επιλογές των μεθόδων βελτιστοποίησης που θα μπορούσαν να εφαρμοστούν στην περίπτωση των χρονισμένων αυτομάτων, η υιοθέτηση κάποιας τεχνικής διακλάδωσης και οριοθέτησης (branch & bound) διαφαίνεται ως η πλέον

αποτελεσματική προσέγγιση περιορισμού των κλάδων αναζήτησης, όπως έχει εφαρμοστεί σε πλήθος NP-Hard προβλημάτων. Τόσο στο κεφάλαιο αυτό όσο και στη συνέχεια της διατριβής η αναζήτηση των βέλτιστων λύσεων θα βασιστεί στη λογική της διακλάδωσης και οριοθέτησης με κατάλληλη προσαρμογή και βελτιστοποίηση της προκειμένου τόσο να είναι αποτελεσματική η αναζήτηση των λύσεων όσο και να μπορούν να αντιμετωπισθούν σύνθετα προβλήματα εκτός των τυποποιημένων προβλημάτων (jobshop, flowshop κλπ) που συνήθως εξετάζονται σε ερευνητικό επίπεδο. Στη συνέχεια αναλύουμε τη λογική στην οποία βασίζεται η προτεινόμενη αξιοποίησή της τεχνικής διακλάδωσης και οριοθέτησης καθώς και τα σημεία στην εξειδίκευση και βελτιστοποίηση των οποίων βασίζεται η αποτελεσματικότητα της μεθόδου για την αναλυόμενη περίπτωση.

Όπως έχει αναλυθεί, στην περίπτωση της μοντελοποίησης προβλημάτων χρονοπρογραμματισμού με χρονισμένα αυτόματα, ο προφανής τρόπος εντοπισμού της βέλτιστης λύσης είναι η αναζήτηση όλων των διαδρομών που οδηγούν από την αρχική στην τελική κατάσταση (όπου όλες οι εργασίες θα έχουν περατωθεί) και η επιλογή αυτής που βελτιστοποιεί την αντικειμενική συνάρτηση – στόχο. Υιοθετώντας μια τεχνική διακλάδωσης και οριοθέτησης μπορεί να επιτευχθεί η απαλοιφή ολόκληρων τμημάτων (ή κλάδων) του χώρου αναζήτησης λύσεων χωρίς εντός των κλάδων αυτών να προσπελασθούν οι πιθανές λύσεις και να συγκριθούν μεταξύ τους. Μεταφέροντας τη λογική του αλγορίθμου B&B στο πρόβλημα χρονοπρογραμματισμού το οποίο αναλύεται στο παρόν κεφάλαιο, εφαρμόζονται τα εξής:

- Αρχικά υπολογίζεται μια οποιαδήποτε λύση του προβλήματος, δηλαδή ένα οποιοδήποτε εφικτό χρονοπρόγραμμα. Η λύση αυτή ορίζεται αρχικά ως λύση αναφοράς.
- Ξεκινάμε την αναζήτηση επιλέγοντας ένα κλάδο (δηλ μια σειρά καταστάσεων) στον οποίο μπορεί να φτάσει το ολικό αυτόματο ξεκινώντας από την αρχική κατάσταση (κατά την οποία καμία εργασία δεν έχει ξεκινήσει).
- Για τον κλάδο αυτό εντοπίζονται όλοι οι υπο-κλάδοι του ή, αλλιώς, οι πιθανές επόμενες καταστάσεις στις οποίες μπορεί να μεταπέσει.
- Για κάθε υπο-κλάδο και χρησιμοποιώντας κάποια επιλεγμένη τεχνική (η οποία εξαρτάται από το πρόβλημα) υπολογίζεται ένα κάτω όριο (lower bound)

το οποίο παρέχει ένα ελάχιστο του χρόνου που απαιτείται για τη μετάβαση από την κατάσταση αυτή στην τελική.

- Στη συνέχεια ο κάθε υπο-κλάδος ελέγχεται ως προς το εξής: Αν ο χρόνος που έχει περάσει από την αρχή μέχρι την ελεγχόμενη κατάσταση, αθροιζόμενος με το υπολογιζόμενο κάτω όριο του υπο-κλάδου είναι μεγαλύτερος από τη λύση αναφοράς, τότε απευθείας όλος ο κλάδος (δηλ όλη η σειρά καταστάσεων που έπονται της ελεγχόμενης) αποκόπτεται και δεν εξετάζεται περαιτέρω για την πιθανότητα η βέλτιστη λύση να προέρχεται από αυτόν. Αν από την άλλη το άθροισμα προκύπτει μικρότερο από την αρχική λύση τότε ο αλγόριθμος προχωράει στην επόμενη κατάσταση και πραγματοποιεί τον ίδιο έλεγχο για τους απογόνους αυτής.
- Εφόσον τελικά με τον τρόπο αυτό προσπελαστεί η τελευταία κατάσταση του ολικού αυτομάτου υπολογίζεται ο συνολικός χρόνος και αν το προκύπτον χρονοπρόγραμμα είναι συντομότερο αυτού της αρχικής λύσης, τότε αυτό παίρνει τη θέση της λύσης – αναφοράς.
- Ο αλγόριθμος συνεχίζεται με τους επόμενους κλάδους και υπο-κλάδους, πραγματοποιώντας κάθε φορά έλεγχο αν ένας κλάδος πρέπει να εξεταστεί ή να αποκοπεί απευθείας αξιοποιώντας τον υπολογισμό του κάτω ορίου (lower bound) για κάθε κατάσταση. Όπως είναι σαφές όσο πιο καλή είναι η λύση αναφοράς τόσο πιο εύκολα αποκόπτονται στη συνέχεια κλάδοι και επιταχύνεται η αναζήτηση

Η παραπάνω προσέγγιση της τεχνικής διακλάδωσης και οριοθέτησης εφαρμοζόμενη σε προβλήματα χρονοπρογραμματισμού εργασιών και εφόσον γίνει κατάλληλη προσαρμογή της ανάλογα με τη φύση του προβλήματος εγγυάται τη δραστική μείωση του πλήθους των εξεταζόμενων λύσεων (μέσω της αποκοπής ολόκληρων τμημάτων/ κλάδων του χώρου λύσεων) και άρα του χρόνου που απαιτείται για να βρεθεί η βέλτιστη λύση του προβλήματος. Καθώς όμως τα σύνθετα προβλήματα χρονοπρογραμματισμού έχουν πολύ μεγάλο αριθμό πιθανών λύσεων, στη πράξη υπάρχουν τρία κρίσιμα στοιχεία τα οποία πρέπει να ερευνηθούν και να απαντηθούν προκειμένου να είναι η παραπάνω προσέγγιση πρακτικά αποτελεσματική. Τα στοιχεία αυτά είναι:

i. **Ο εντοπισμός της Αρχικής Λύσης**: Η εύρεση μιας αρχικής (όχι βέλτιστης) λύσης του προβλήματος αποτελεί προϋπόθεση για την

εφαρμογή μιας τεχνικής B&B. Η λύση αυτή μπορεί να εντοπισθεί με οποιοδήποτε τρόπο στην περίπτωση του χρονοπρογραμματισμού με αυτόματα, όπως για παράδειγμα εφαρμόζοντας ένα οποιοδήποτε αλγόριθμο προσπέλασης (πχ BFS, DFS κλπ) όπως περιγράφηκε παραπάνω. Εν τούτοις πρέπει να τονιστεί ότι όσο καλύτερη (δηλ πιο κοντά στη βέλτιστη) είναι η αρχική αυτή λύση τόσο γρηγορότερα θα μπορεί να γίνει η απαλοιφή μεγάλων κλάδων του χώρου λύσεων, επιτυγχάνοντας έτσι μείωση του συνολικού υπολογιστικού χρόνου. Έτσι παρά το γεγονός ότι με απλούς αλγορίθμους μπορεί να υπολογιστούν διάφορες λύσεις ενός προβλήματος, η εκκίνηση από μια όσο καλύτερη γίνεται αρχική λύση αποτελεί κρίσιμη παράμετρο.

ii. **Η οριοθέτηση, ή αλλιώς ο υπολογισμός του Κάτω Ορίου (Lower Bound)**: Ή εκτίμηση (ή υπολογισμός) του κάτω ορίου τόσο σε προβλήματα χρονοπρογραμματισμού όσο και σε άλλα προβλήματα που προσεγγίζονται με αυτόματα είναι ιδιαίτερα κρίσιμος. Αν ο τρόπος υπολογισμού είναι πολύ συντηρητικός τότε πολύ λίγοι κλάδοι τελικά θα αποκοπούν και άρα η αποτελεσματικότητα του αλγορίθμου θα είναι ιδιαίτερα χαμηλή. Από την άλλη αν ο τρόπος υπολογισμού οδηγεί σε υψηλές τιμές για το κάτω όριο, τότε αυξάνονται σημαντικά οι πιθανότητες κάποιος κλάδος να αποκοπεί χωρίς να ελεγχθεί, ενώ τελικά η βέλτιστη λύση να προέρχεται από τον κλάδο αυτόν.

iii. **Ο Τρόπος Αναζήτησης ή αλλιώς ο τρόπος με τον οποίο κάθε φορά επιλέγεται η επόμενη προς εξέταση κατάσταση**: Μια απλή προσέγγιση του τρόπου αναζήτησης είναι να καταφύγουμε στους γνωστούς αλγορίθμους αναζήτησης κατά πλάτος ή κατά βάθος (BFS και DFS αντίστοιχα). Εν τούτοις μια τέτοια στρατηγική αυξάνει τελικά την τυχαιότητα και παρ' ότι είναι λειτουργική δε βελτιστοποιεί την αποτελεσματικότητα της προσέγγισης, κάτι που θα μπορούσε να συμβεί μέσα από την επιλογή μιας πιο προσαρμοσμένης στο πρόβλημα στρατηγικής αναζήτησης λύσεων.

6.5.2. Πειραματικά αποτελέσματα

Καθώς η επίλυση σύνθετων προβλημάτων χρονοπρογραμματισμού με χρήση αυτομάτων τελικά διέρχεται από την εφαρμογή εξειδικευμένων προσεγγίσεων B&B, η ανάπτυξη πρωτότυπων λύσεων αντιμετώπισης των αναφερθέντων ζητημάτων αποτέλεσε βασικό στόχο της παρούσας διατριβής, όπως αναλύεται στα επόμενα κεφάλαια. Δεδομένου όμως ότι προκειμένου να λυθεί το ζήτημα του αλγορίθμου εύρεσης λύσεων, απαραίτητη προϋπόθεση αποτελεί να έχει μοντελοποιηθεί το πρόβλημα με τρόπο τέτοιο ώστε να μπορεί αποτελεσματικά να επιλυθεί, στη φάση αυτή χρησιμοποιούνται ορισμένες παραδοχές προκειμένου από τη μία να δοκιμαστούν και να συγκριθούν μεταξύ τους τα μοντέλα που αναπτύχθηκαν στις προηγούμενες παραγράφους και από την άλλη να δοκιμαστεί η εφαρμογή μια βασικής προσέγγισης διακλάδωσης και οριοθέτησης και η αποτελεσματικότητά της έναντι του βελτιστοποιημένου αλγορίθμου ευθείας προσπέλασης. Η εξέταση του απλού αλγορίθμου ευθείας προσπέλασης είναι φυσικά άνευ νοήματος καθώς ο βελτιστοποιημένος αλγόριθμος εγγυάται το αυτό αποτέλεσμα σε σαφώς μικρότερο χρόνο.

Στο παραπάνω πλαίσιο, για τα τρία μοντέλα χρονισμένων αυτομάτων που αναπτύχθηκαν προχωρήσαμε στην επίλυση μιας σειράς δοκιμαστικών προβλημάτων, αξιοποιώντας το βασικό αλγόριθμο διακλάδωσης και οριοθέτησης με τις εξής συνθήκες:

- Η αρχική λύση υπολογίστηκε σε κάθε περίπτωση με χρήση του αλγορίθμου DFS ο οποίος καθώς κινείται σε βάθος μπορεί να παρέχει γρήγορα μια τυχαία λύση του προβλήματος
- Ο υπολογισμός του κάτω ορίου έγινε κατ’ αντιστοιχία με την προσέγγιση για το άνω όριο κατά την ανάπτυξη του δεύτερου μοντέλου. Έτσι ως κάτω όριο ορίστηκε για κάθε κατάσταση το μέγιστο άθροισμα των διαρκειών των εναπομενουσών διεργασιών κάθε εργασίας. Το συγκεκριμένο όριο είναι φυσικά αρκετά συντηρητικό καθώς υποθέτει ότι οι εργασίες εκτελούνται όλες χωρίς καμία καθυστέρηση. Από την άλλη αποτελεί ένα όριο το οποίο σε καμία περίπτωση δεν αποκλείει εκ παραδρομής κλάδους στους οποίους πιθανόν περιλαμβάνεται η βέλτιστη λύση. Έτσι παρέχεται η δυνατότητα αντικειμενικής σύγκρισης και αξιολόγησης των μοντέλων που αναπτύχθηκαν.

- Ο τρόπος αναζήτησης εναλλάσσεται ανάμεσα σε BFS και DFS. Η επιλογή των αλγορίθμων αυτών αποτελεί μονόδρομο αλλά και αποτελεσματική προσέγγιση για τα δεδομένα των παραπάνω μοντέλων. Όπως θα φανεί στη συνέχεια, μέσω της προσθήκης στοιχείων όπως το κόστος στα αυτόματα, η επιλογή ή ανάπτυξη ενός πιο αποτελεσματικού τρόπου αναζήτησης είναι κρίσιμη. Εν τούτοις στη φάση αυτή και για τα εξεταζόμενα προβλήματα δεν αποτελεί προϋπόθεση.

Στη συνέχεια παρουσιάζονται τα αποτελέσματα της εφαρμογής των τριών μοντέλων που αναπτύχθηκαν στην προσπάθεια επίλυσης μιας σειράς δοκιμαστικών προβλημάτων. Αξιοποιώντας ανοικτές βιβλιοθήκες jobshop προβλημάτων [51], [76], [34] εξετάστηκαν 28 προβλήματα διαστάσεων από 3x3 μέχρι 7x7 με γνωστές βέλτιστες λύσεις. Για κάθε μοντέλο και για κάθε πρόβλημα εξετάστηκε τόσο η εφαρμογή του αλγορίθμου ευθείας προσπέλασης όσο και η εφαρμογή της τεχνικής διακλάδωσης και οριοθέτησης με τις συνθήκες που περιγράφηκαν παραπάνω. Καθώς, ειδικά για την πρώτη περίπτωση, ο αριθμός των καταστάσεων και αντίστοιχα ο χρόνος ελέγχου τους είναι μη-πολυωνυμικός για τα μεγαλύτερα προβλήματα, θέσαμε ένα άνω όριο στο χρόνο εκτέλεσης ίσο με 3.600 sec. Για τις περιπτώσεις που η μηχανή ελέγχου έφτανε να υπερβεί το όριο, επιλέξαμε να σταματάει η εκτέλεση του. Σημειώνεται ότι δεδομένου πως καθώς η πολυπλοκότητα ακόμα και μικρού μεγέθους προβλημάτων jobshop (της τάξεως των 4 - 5 εργασιών/μηχανών) θεωρείται μεγάλη και καθώς ο στόχος μας στο σημείο αυτό ήταν η σύγκριση τόσο των τριών μοντέλων μεταξύ τους, όσο και των δύο προσεγγίσεων επίλυσης τους, επιλέξαμε να περιοριστούμε σε προβλήματα όχι μεγαλύτερα, προκειμένου να αυξηθούν οι πιθανότητες ολοκλήρωσης της διαδικασίας επίλυσης εντός του χρονικού ορίου που τέθηκε.

Τα αποτελέσματα στα εξεταζόμενα προβλήματα παρουσιάζονται στους πίνακες που ακολουθούν. Σημειώνεται ότι η βέλτιστη λύση **σε όλα τα προβλήματα που λύθηκαν** με κάθε μοντέλο να ταυτίζεται με τη λύση της βιβλιογραφίας. Ως εκ τούτου η λύση παρέχεται καθαρά για λόγους πληρότητας καθώς η σημασία της πειραματικής αυτής διαδικασίας αφορά κατά βάση στην αποτελεσματικότητα των μοντέλων που αναπτύχθηκαν και των δύο προτεινόμενων προσεγγίσεων επίλυσης.

Μέγεθος (Εργασίες/Μηχανές)		3/3	3/4	3/5	3/6
Βέλτιστη Λύση		167	246	293	299
Μοντέλο 1 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	48.325	91.107	357.732	583.250
	Χρόνος(ms)	947,55	1.822,13	7.300,65	12.151,03
Μοντέλο 2 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	21.966	35.041	119.244	171.544
	Χρόνος(ms)	448,29	730,02	2.537,11	3.729,22
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	12.552	31.380	51.777	40.794
	Χρόνος(ms)	258,80	660,63	1.113,48	896,57
Μοντέλο 3 (Διακλ. & Οριοθ.)	Καταστάσεις	9.937	17.259	40.271	37.656
	Χρόνος(ms)	207,02	367,21	875,46	836,80

Μέγεθος (Εργασίες/Μηχανές)		3/7	4/2	4/3	4/4
Βέλτιστη Λύση		338	169	232	264
Μοντέλο 1 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	461.077	65.898	408.986	930.940
	Χρόνος(ms)	9.810,14	1.432,57	9.088,58	21.157,73
Μοντέλο 2 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	121.336	15.690	88.910	186.188
	Χρόνος(ms)	2.696,36	356,59	2.067,67	4.433,05
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	46.547	51.254	132.842	180.435
	Χρόνος(ms)	1.046,00	1.178,25	3.125,69	4.347,83
Μοντέλο 3 (Διακλ. & Οριοθ.)	Καταστάσεις	44.455	10.983	39.748	66.944
	Χρόνος(ms)	1.010,34	255,42	946,38	1.632,78

Μέγεθος (Εργασίες/Μηχανές)		4/5	4/6	5/2	5/3
Βέλτιστη Λύση		302	307	175	219
Μοντέλο 1 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	1.310.429	2.375.152	1.494.839	2.885.705
	Χρόνος(ms)	30.475,09	56.551,24	36.459,48	72.142,62
Μοντέλο 2 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	242.672	409.509	241.103	437.228
	Χρόνος(ms)	5.918,83	10.237,73	6.182,13	11.506,00
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	71.651	55.438	19.351	99.893
	Χρόνος(ms)	1.769,16	1.403,49	502,62	2.663,81
Μοντέλο 3 (Διακλ. & Οριοθ.)	Καταστάσεις	54.392	50.731	15.690	48.116
	Χρόνος(ms)	1.359,80	1.300,79	412,89	1.300,43

Μέγεθος (Εργασίες/Μηχανές)		5/4	5/5	5/6	5/7
Βέλτιστη Λύση		267	291	379	435
Μοντέλο 1 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	2.031.855	9.195.595	7.444.905	82.491.221
	Χρόνος(ms)	52.098,85	241.989,35	201.213,65	2.291.422,81
Μοντέλο 2 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	290.265	1.242.648	954.475	10.059.905
	Χρόνος(ms)	7.845,00	34.518,00	27.270,71	295.879,56
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	189.326	100.416	358.778	1.317.960
	Χρόνος(ms)	5.187,01	2.828,62	10.399,36	39.342,09
Μοντέλο 3 (Διακλ. & Οριοθ.)	Καταστάσεις	51.777	65.375	64.852	174.682
	Χρόνος(ms)	1.438,25	1.867,86	1.907,41	5.293,39

Μέγεθος (Εργασίες/Μηχανές)		6/2	6/3	6/4	6/5
Βέλτιστη Λύση		258	274	339	367
Μοντέλο 1 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	10.245.988	18.201.969	17.128.041	52.904.379
	Χρόνος(ms)	292.742,53	535.352,03	519.031,54	1.653.261,84
Μοντέλο 2 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	1.191.394	2.022.441	1.822.132	5.398.406
	Χρόνος(ms)	36.102,85	63.201,28	58.778,45	179.946,87
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	647.474	976.441	5.953.832	1.143.278
	Χρόνος(ms)	19.922,28	30.998,13	195.207,61	38.755,19
Μοντέλο 3 (Διακλ. & Οριοθ.)	Καταστάσεις	19.351	32.426	362.439	69.036
	Χρόνος(ms)	604,72	1.046,00	12.081,30	2.380,55

Μέγεθος (Εργασίες/Μηχανές)		6/6	6/7	7/2	7/3
Βέλτιστη Λύση		433	407	303	318
Μοντέλο 1 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	-	-	41.893.346	2.677.028
	Χρόνος(ms)	-	-	1.444.598,14	95.608,14
Μοντέλο 2 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	86.124.502	56.193.735	3.808.486	234.827
	Χρόνος(ms)	2.969.810,41	2.006.919,11	141.055,04	9.031,81
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	28.545.863	4.420.396	31.903	204.723
	Χρόνος(ms)	1.001.609,23	160.741,67	1.203,89	8.028,76
Μοντέλο 3 (Διακλ. & Οριοθ.)	Καταστάσεις	1.956.543	501.034	30.334	112.968
	Χρόνος(ms)	69.876,54	18.556,81	1.166,69	4.518,72

Μέγεθος (Εργασίες/Μηχανές)		7/4	7/5	7/6	7/7
Βέλτιστη Λύση		336	407	412	426
Μοντέλο 1 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	-	-	-	-
	Χρόνος(ms)	-	-	-	-
Μοντέλο 2 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	10.202.684	-	-	-
	Χρόνος(ms)	408.107,36	-	-	-
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	Καταστάσεις	3.700.493	2.105.999	16.841.929	18.424.767
	Χρόνος(ms)	151.227,14	91.910,45	992.974,62	856.965,91
Μοντέλο 3 (Διακλ. & Οριοθ.)	Καταστάσεις	623.416	1.999.952	3.526.066	706.573
	Χρόνος(ms)	25.975,67	86.954,43	160.275,73	33.646,33

Σύγκριση μοντέλων και αλγορίθμων για τα προβλήματα που λύθηκαν σε όλες τις περιπτώσεις

Μοντέλο (Αλγόριθμος)	Αριθμός Προβλημάτων που επιλύθηκαν εντός χρονικού ορίου	Μ.Ο. Αριθμού Καταστάσεων (22 κοινά προβλήματα)	Μ.Ο. Χρόνου Εκτέλεσης (ms) (22 κοινά προβλήματα)
Μοντέλο 1 (Βελτ.Ευθ.Προσπ)	22	11.601.262	344.848,08
Μοντέλο 2 (Βελτ.Ευθ.Προσπ)	25	1.323.428	41.112,39
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	28	532.638	16.856,33
Μοντέλο 3 (Διακλ. & Οριοθ.)	28	64.519	1.945,92

Σύγκριση αλγορίθμων για το σύνολο των προβλημάτων με τη χρήση του μοντέλου 3

Μοντέλο (Αλγόριθμος)	Μ.Ο. Αριθμού Καταστάσεων	Μ.Ο. Χρόνου Εκτέλεσης (ms)
Μοντέλο 3 (Βελτ.Ευθ.Προσπ)	3.062.768	129.509,58
Μοντέλο 3 (Διακλ. & Οριοθ.)	383.322	15.646,28

Η ανάλυση των αποτελεσμάτων καταλήγει σε ορισμένα βασικά συμπεράσματα σχετικά με τις προτεινομενες προσεγγίσεις. Τα συμπεράσματα αυτά ήταν πλήρως αναμενόμενα βάσει της ποιοτικής ανάλυσης που πραγματοποιήθηκε νωρίτερα, εν τούτοις ποσοτικοποιήθηκαν και επιβεβαιώθηκαν μέσα από τα πειραματικά αποτελέσματα.

- Όλα τα μοντέλα ήταν ικανά να λύσουν το σύνολο των προβλημάτων και κατάφεραν να εντοπίσουν τις βέλτιστες λύσεις όπως αυτές παρουσιάζονται στη βιβλιογραφία.
- Λόγω του ορίου των 3.600 δευτερολέπτων που τέθηκε, τα Μοντέλα 1 και 2 δεν κατάφεραν να λύσουν ορισμένα από τα προβλήματα και ο αλγόριθμος σταμάτησε στο όριο του χρόνου.
- Το μοντέλο 3 παρουσιάζει σημαντική μείωση του αριθμού των ελεγχόμενων καταστάσεων σε σχέση με τα 1 και 2.
- Η προσέγγιση διακλάδωσης και οριοθέτησης βελτίωσε σημαντικά το χρόνο επίλυσης των περισσοτέρων προβλημάτων καθώς και τον αριθμό των εξεταζόμενων καταστάσεων αντίστοιχα.

Με βάση τα παραπάνω, το 3^{ο} μοντέλο που αναπτύχθηκε έχει τα εχέγγυα για να αξιοποιηθεί ως το βασικό μοντέλο πάνω στο οποίο θα χτιστούν επεκτάσεις στη συνέχεια της διατριβής. Παράλληλα, δεδομένης της πολύ καλής απόδοσης της τεχνικής της διακλάδωσης και οριοθέτησης θα γίνει προσπάθεια να προσεγγιστούν και τα πιο σύνθετα προβλήματα με τη μέθοδο αυτή. Σημειώνεται φυσικά ότι η τεχνική προϋποθέτει τον υπολογισμό ή την εκτίμηση συγκεκριμένων παραμέτρων προκειμένου να είναι αποδοτική και ενώ οι παράμετροι αυτές είναι εύκολο σχετικά να υπολογιστούν για το απλό πρόβλημα jobshop, η εύρεση (ή εκτίμηση) τους απαιτεί σύνθετες μεθόδους αν το ίδιο το πρόβλημα είναι μεγαλύτερης πολυπλοκότητας.

Κεφάλαιο 7:
Χρονοπρογραμματισμός με διακοπτόμενες εργασίες

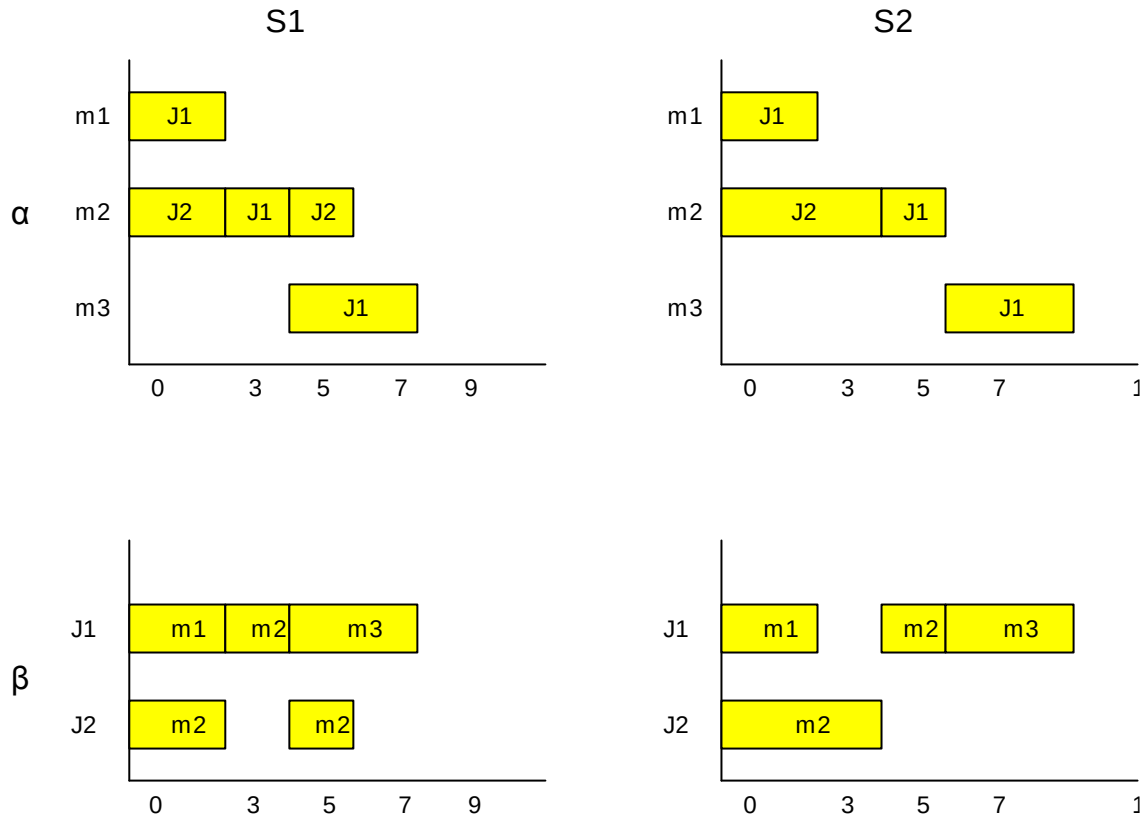
7.1. Προεκτοπισμός εργασιών

Στο παρόν κεφάλαιο θα επεκτείνουμε το πρόβλημα του χρονοπρογραμματισμού ντετερμινιστικών Job –shop συστημάτων θεωρώντας ότι οι εργασίες μπορούν να προεκτοπιστούν. Αυτό πρακτικά σημαίνει ότι μια εργασία μπορεί να χρησιμοποιήσει μια μηχανή για κάποιο χρονικό διάστημα, στη συνέχεια να διακοπεί η επεξεργασία της από τη μηχανή και κάποια στιγμή αργότερα να συνεχιστεί εκ νέου η επεξεργασία από το σημείο στο οποίο σταμάτησε. Τέτοιες καταστάσεις συνήθως συμβαίνουν όταν οι μηχανές είναι ηλεκτρονικοί υπολογιστές.

Έστω για παράδειγμα ότι έχουμε ένα σύνολο πόρων που περιλαμβάνει τρεις μηχανές M={m_1, m_2, m_3} και ένα σύνολο εργασιών J που περιλαμβάνει δύο εργασίες $J = \left\{J^1, J^2\right\}$. Τα χαρακτηριστικά των εργασιών στις μηχανές έχουν ως εξής:

$$J^1 = (m_1,3),(m_2,2),(m_3,4) \text{ και } J^2 = (m_2,5)$$

Με βάση τα παραπάνω μπορούμε να εξάγουμε δύο χρονοπρογράμματα S_1 και S_2 όπως φαίνεται και στο Σχήμα 47.



Σχήμα 47: Τα δύο προεκτοπιστικά χρονοπρογράμματα και με βάση τις συναρτήσεις κατανομής των μηχανών (α) και προόδου των εργασιών (β)

Όπως μπορεί να παρατηρήσει κανείς στο χρονοπρόγραμμα S_1 η εργασία J^2 προεκτοπίζεται στο χρόνο *t*=3 στη μηχανή m_2 και η οποία από τη χρονική αυτή στιγμή και μετά αρχίζει να απασχολείται από την εργασία J^1. Ακολούθως αφού η J^1 ολοκληρωθεί στη m_2, η J^2 επανέρχεται στην συγκεκριμένη μηχανή για να συνεχιστεί και τελικά ολοκληρωθεί η επεξεργασία της. Με αυτό τον τρόπο εκτέλεσης των εργασιών η διάρκεια του S_1 είναι 9 που στην ουσία είναι και το βέλτιστο χρονοπρόγραμμα.

Οι ορισμοί που δόθηκαν στο προηγούμενο κεφάλαιο σχετικά με τα χαρακτηριστικά ενός Job –shop συστήματος (Ορισμός 8) και του εφικτού χρονοπρογράμματος (Ορισμός 9) παραμένουν οι ίδιοι και σε αυτό το κεφάλαιο. Αυτό όμως που αλλάζει εδώ είναι ο περιορισμός που αναφερόταν στη τρίτη συνθήκη, της

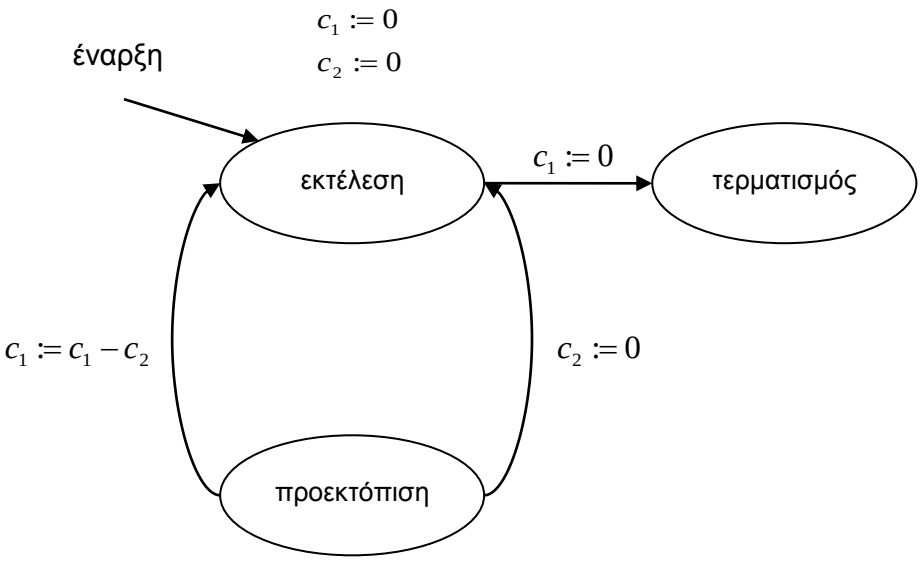
μη προεκτόπισης, που δόθηκε στον Ορισμό 9 του προηγουμένου κεφαλαίου. Αυτό σημαίνει πως για κάθε βήμα (i, j) το σύνολο $\{t,(i, j,t)\in S\}$ είναι μια ένωση από διαστήματα τέτοια ώστε το άθροισμα του μήκους κάθε ενός να είναι ίσο με τη διάρκεια του βήματος.

7.1.1. Μοντελοποίηση με αυτόματα διακοπτόμενων χρόνων

Το είδος του αυτομάτου που έχει παρουσιαστεί μέχρι τώρα δεν μας ικανοποιεί πλέον αφού μέσω αυτού δεν είναι δυνατόν να παρουσιαστεί η προεκτόπιση ενός βήματος. Συγκεκριμένα, όπως έχει αναφερθεί, μια εργασία σε ένα αυτόματο μπορεί να αφήσει μια κατάσταση εκτέλεσης στην οποία βρίσκεται μόνο αν το βήμα έχει ολοκληρωθεί. Στη περίπτωση που εξετάζουμε εδώ η μετακίνηση από τη κατάσταση εκτέλεσης μπορεί να πραγματοποιηθεί ακόμα και αν η εκτέλεση δεν έχει ολοκληρωθεί πλήρως.

Για να μοντελοποιήσουμε λοιπόν την διαδικασία της προεκτόπισης πρέπει να προσθέσουμε μια επιπλέον κατάσταση που την ονομάζουμε «προεκτόπιση». Μέσω αυτής της κατάστασης το αυτόματο μπορεί να μετακινείται μπροστά ή πίσω μεταξύ των καταστάσεων «εκτέλεση» και «προεκτόπιση», όπως φαίνεται και στο Σχήμα 48.

Στη περίπτωση της μοντελοποίησης με το προεκτοπιστικό μοντέλο δεν μπορούμε να χρησιμοποιήσουμε το χρονιστή c_1 ως βοηθό μετάβασης (guard) για τη μετάβαση στην κατάσταση ολοκλήρωσης του βήματος. Μια από τις πιθανές λύσης που θα μπορούσε να εφαρμοστεί για την αντιμετώπιση αυτού του προβλήματος είναι η πρόσθεση ενός επιπλέον χρονιστή c_2, ο ρόλος του οποίου θα είναι να μετράει τον χρόνο που διαρκεί η προεκτόπιση και να αναβαθμίζει (χρονικά) το χρονιστή c_1. Έτσι όταν η εκτέλεση του βήματος επανέρχεται για να συνεχιστεί από εκεί που σταμάτησε η τιμή του χρονιστή θα είναι c_1-c_2. Η λειτουργία αυτή είναι πέραν της λειτουργίας της απλής επαναφοράς που συναντήσαμε στο προηγούμενο κεφάλαιο.



Σχήμα 48: Διαδικασία προεκτόπισης

Μια εναλλακτική λύση αναφορικά με τους χρονιστές, θα ήταν να επεκτείνουμε το μοντέλο των χρονισμένων αυτομάτων με χρονιστές οι οποίοι θα «πάγωναν» σε συγκεκριμένες καταστάσεις, δηλαδή χρονιστές με παράγωγο το μηδέν. Τα αυτόματα που προκύπτουν από αυτόν τον τρόπο ονομάζονται Γράφοι Ενοποίησης (Integration Graphs) [97]. Ανάλογα αυτόματα έχουν ερευνηθεί σε διάφορες εργασίες [117], [37] στις οποίες περιγράφονται η εφαρμογή προσεγγιστικών αλγορίθμων επαλήθευσης.

Ορισμός 19 (Αυτόματα Διακοπτόμενων Χρόνων): Ένα αυτόματο διακοπτόμενου χρόνου είναι μια πλειάδα της μορφής *A*=(*Q*,*C*,*s*,*f*,*u*,Δ) όπου *Q* είναι ένα πεπερασμένο σύνολο καταστάσεων, *C* είναι ένα πεπερασμένο σύνολο χρονιστών, το $u : Q \to \{0,1\}^n$ προσδιορίζει τη σταθερή κλίση σε κάθε κατάσταση και το Δ είναι η σχέση μετάβασης η οποία περιλαμβάνει στοιχεία της μορφής (q, ϕ, p, q') όπου *q* και *q'* είναι καταστάσεις, $p \subseteq C$ και ϕ (ο βοηθός μετάβασης) είναι ένας συνδυασμός της άλγεβρας Boole της μορφής ($c \in I$) για κάποιο χρονιστή *c* και κάποιο διάστημα εύρους ακεραίων *I*. Οι καταστάσεις *s* και *f* είναι η αρχική και τελική κατάσταση αντίστοιχα.

7.1.2. Μοντελοποίηση εργασιών

Η διαδικασία μοντελοποίησης των εργασιών στο προεκτοπιστικό μοντέλο που παρουσιάζεται εδώ έχει αρκετά κοινά στοιχεία με τη διαδικασία που ακολουθήθηκε στο ντετερμινιστικό μοντέλο του προηγουμένου κεφαλαίου. Συγκεκριμένα για κάθε εργασία $J=(k,\mu,d)$ κατασκευάζουμε ένα αυτόματο διακοπτόμενου χρόνου με ένα χρονιστή τέτοιο ώστε για κάθε βήμα j με $\mu(j)=m$ υπάρχουν τριών ειδών διαφορετικές καταστάσεις:

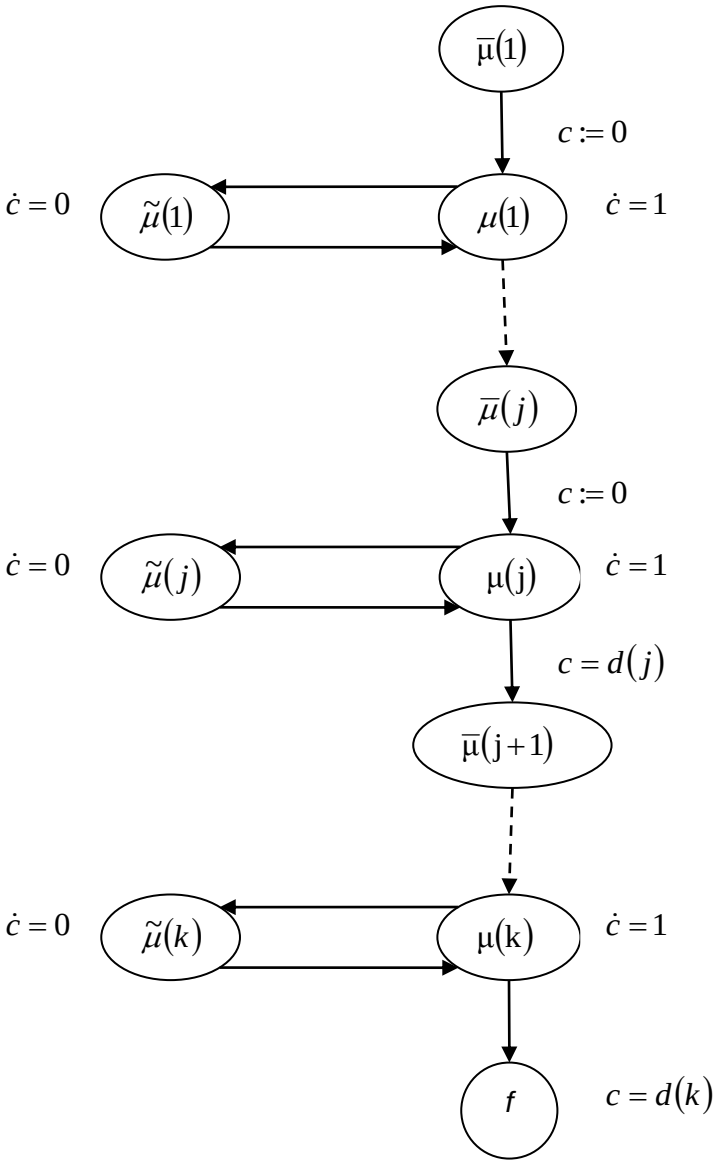
- Μια κατάσταση αναμονής $\overline{m}$
- Μια ενεργή κατάσταση m όπου δείχνει ότι η συγκεκριμένη εργασία εκτελείται στη μηχανή και
- Μια κατάσταση $\tilde{m}$ που δείχνει ότι η εργασία έχει προεκτοπιστεί μετά από κάποιο χρονικό διάστημα αφότου έχει ξεκινήσει.

Εισερχόμενοι στην κατάσταση m ο χρονιστής μηδενίζεται και αμέσως ξεκινάει να μετράει το χρόνο που περνάει μέσα στην (ενεργή) κατάσταση m. Οι διαδικασίες της προεκτόπισης και της επανέναρξης της εκτέλεσης της εργασίας από τη μηχανή, μοντελοποιούνται μέσω μεταβάσεων προς και από την κατάσταση $\tilde{m}$ στην οποία ο χρόνος δεν προοδεύει. Όταν ο χρονιστής φτάσει στην τιμή $d(j)$ τότε το αυτόματο αφήνει την κατάσταση m και προχωράει στην επόμενη κατάσταση αναμονής $\overline{m}$. Έστω $\overline{M}=\{\overline{m}: m\in M\}$, $\tilde{M}=\{\tilde{m}: m\in M\}$ και έστω $\tilde{\mu}: K\to\tilde{M}$ είναι μια βοηθητική συνάρτηση τέτοια ώστε $\overline{\mu}(j)=\overline{m}$ και $\tilde{\mu}(j)=\tilde{m}$ οποτεδήποτε $\mu(j)=m$.

Αυτόματο Διακοπτόμενου Χρόνου για την εργασία: Έστω $J=(k,\mu,d)$ είναι μια εργασία. Το αυτόματο της δεδομένης εργασίας είναι $A=(Q,\{c\},u,\Delta,s,f)$ με $Q=P\cup\overline{P}\cup\tilde{P}\cup\{f\}$ όπου $P=\{\mu(1),\mu(2),\ldots,\mu(n)\}$, $\overline{P}=\{\overline{\mu}(1),\overline{\mu}(2),\ldots,\overline{\mu}(n)\}$ και $\tilde{P}=\{\tilde{\mu}(1),\tilde{\mu}(2),\ldots,\tilde{\mu}(n)\}$. Η κλίση ορίζεται ως $u_q=1$ όπου $q\in P$ ενώ διαφορετικά $u_q=0$. Η σχέση μετάβασης περιλαμβάνει τις ακόλουθες πλειάδες της μορφής:

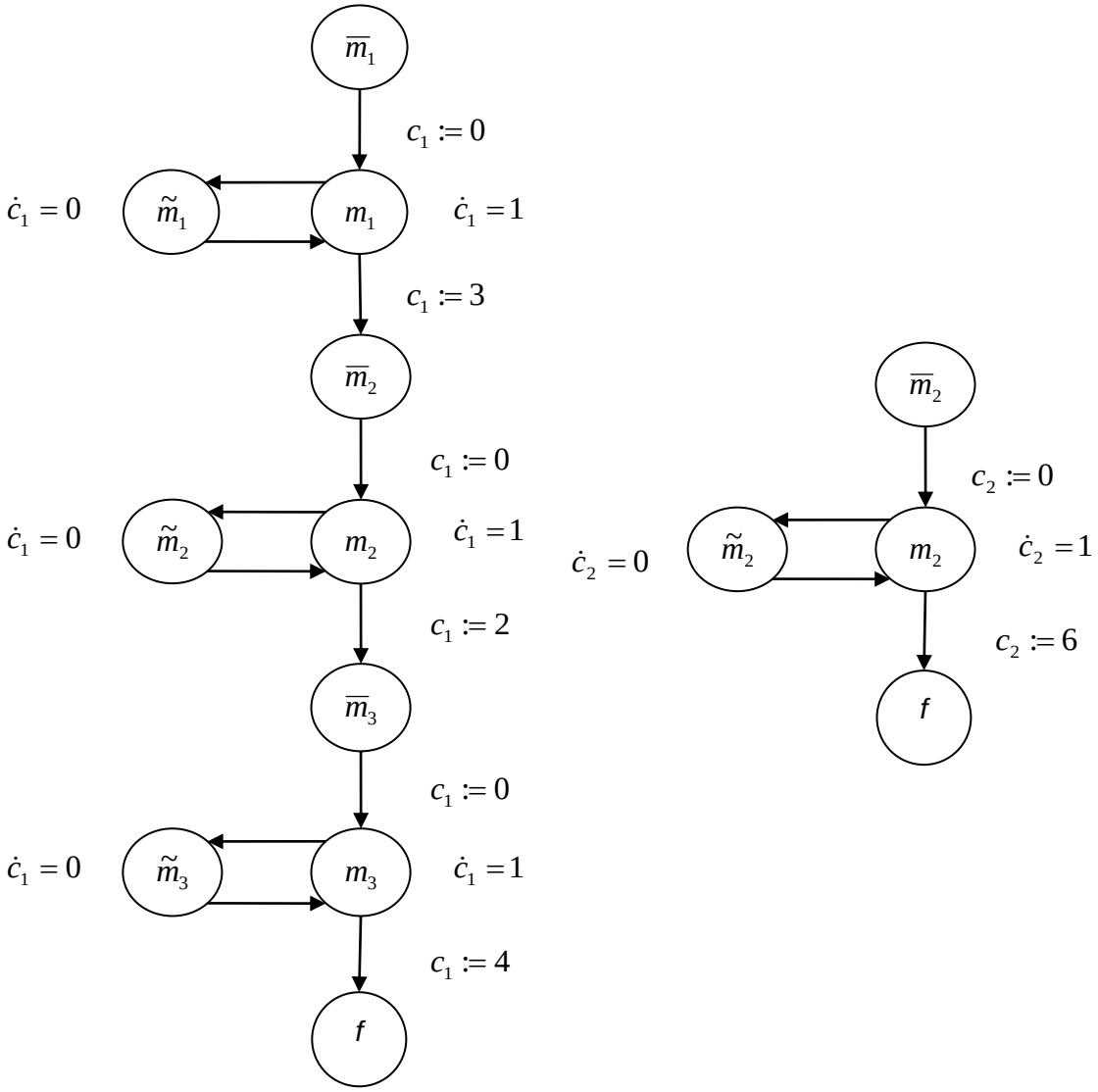
τύπος	q	ϕ	ρ	q'	
1) έναρξη	$\overline{\mu}(j)$	true	$\{c\}$	$\mu(j)$	$j=1\ldots k$
2) παύση	$\mu(j)$	true	$\varnothing$	$\tilde{\mu}(j)$	$j=1\ldots k$
3)επανέναρξη	$\tilde{\mu}(j)$	true	$\varnothing$	$\mu(j)$	$j=1\ldots k$
4)τέλος	$\mu(j)$	$c=d(j)$	$\varnothing$	$\overline{\mu}(j+1)$	$j=1\ldots k-1$
τέλος	$\mu(k)$	$c=d(k)$	$\varnothing$	f	

Η αρχική κατάσταση είναι $\bar{\mu}(1)$.



Σχήμα 49: Το γενικό αυτόματο διακοπτόμενου χρόνου για μια συγκεκριμένη εργασία

Αντίστοιχα με το παραπάνω Σχήμα 49 το αυτόματο διακοπτόμενου χρόνου που αντιστοιχεί στις δύο εργασίες του προαναφερθέντος παραδείγματος παρουσιάζεται στο Σχήμα 50.

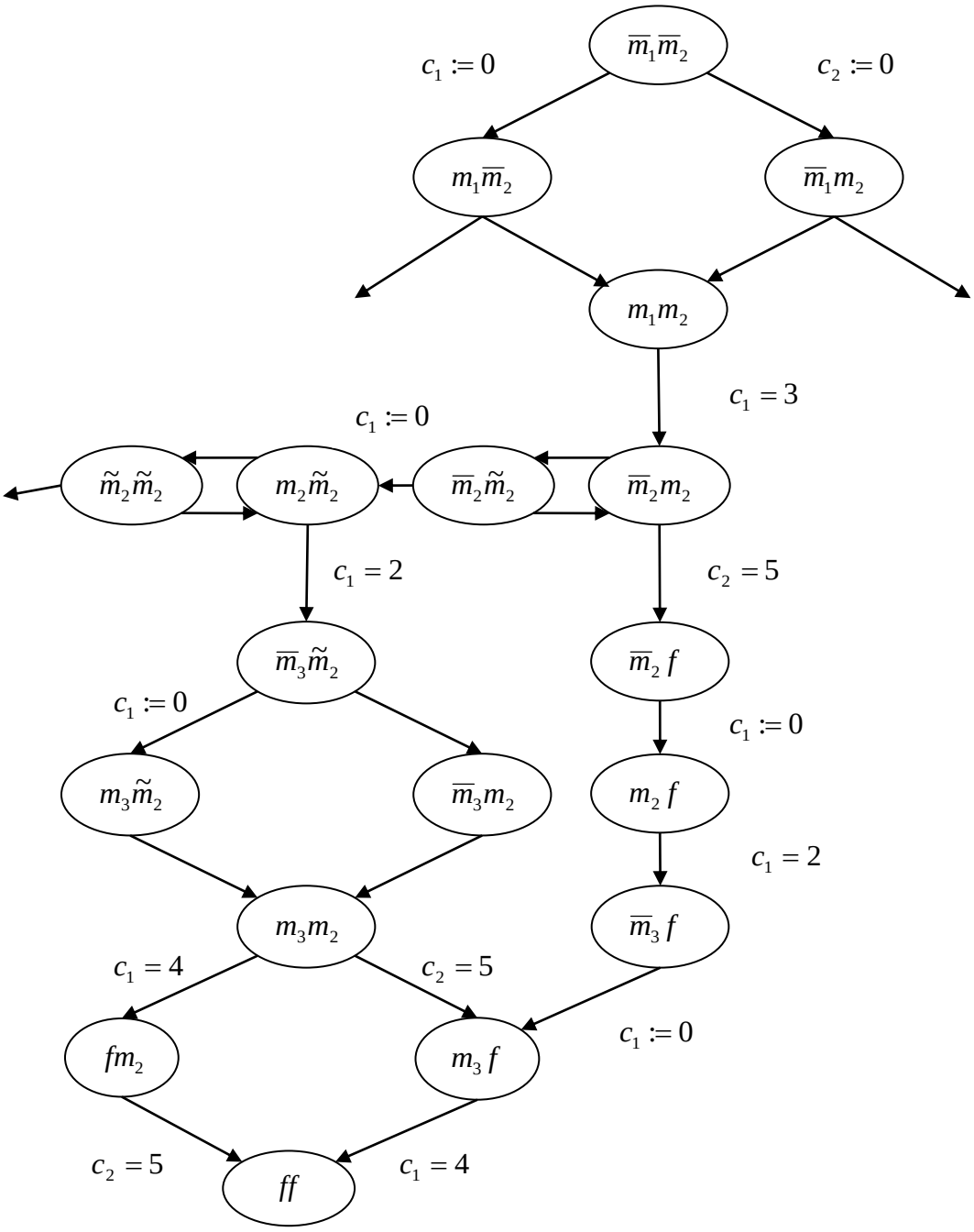


Σχήμα 50: Τα αυτόματα διακοπτόμενων χρόνων που αντιστοιχούν στις εργασίες $J^1 = (m_1,3),(m_2,2),(m_3,4)$ ***και*** $J^2 = (m_2,5)$

7.1.3. Το συνολικό αυτόματο

Συνθέτοντας τα αυτόματα των δύο εργασιών μπορούμε να αποκτήσουμε το συνολικό αυτόματο που ουσιαστικά επιτυγχάνει τη μοντελοποίηση του προεκτοπιστικού Job –shop προβλήματος. Βέβαια είναι προφανές ότι πρέπει να ισχύει και για τη περίπτωση που εξετάζουμε εδώ η συνθήκη σύνθεσης αμοιβαίας απόκλισης (mutual exclusion composition) η οποία αναφέρθηκε αναλυτικότερα στο προηγούμενο κεφάλαιο.

Στο Σχήμα 51 παρουσιάζεται μέρος του αυτομάτου το οποίο προκύπτει αν συνθέσουμε τα δύο αυτόματα του Σχήματος 50. Στο συγκεκριμένο αυτόματο έχουν παραληφθεί χάριν απλούστευσης οι μεταβάσεις προεκτόπισης και επανέναρξης για τις μηχανές m_1 και m_3 όπως επίσης και για μερικές άλλες που οδηγούν σε μη σημαντικές διαδρομές. Παρατηρούμε πως σε αντίθεση με το ντετερμινιστικό αυτόματο το προεκτοπιστικό είναι κυκλικό. Αυτό οφείλεται στο γεγονός ότι η προεκτόπιση και επανέναρξη ενός οποιουδήποτε βήματος μπορεί να συμβεί οποιαδήποτε χρονική στιγμή.



Σχήμα 51: Μέρος του συνολικού αυτομάτου διακοπτόμενου χρόνου των δύο εργασιών

7.1.4. Εκτελέσεις και χρονοπρογράμματα

Η αντιστοιχία μεταξύ εκτελέσεων και εφικτών χρονοπρογραμμάτων είναι ανάλογη με αυτή του ντετερμινιστικού προβλήματος.

Έστω A είναι ένα αυτόματο διακοπτόμενου χρόνου ενός προεκτοπιστικού Job – shop συστήματος με χαρακτηριστικά J. Κάθε ολοκληρωμένη εκτέλεση του A αντιστοιχεί σε ένα εφικτό χρονοπρόγραμμα με μήκος ίσο με το μετρικό μήκος της εκτέλεσης.

Τα δύο χρονοπρογράμματα του Σχήματος 47 αντιστοιχούν στις ακόλουθες δύο εκτελέσεις:

$$\xi_1: \begin{array}{l}(\overline{m}_1,\overline{m}_2,\perp,\perp)\xrightarrow{0}(m_1,\overline{m}_2,0,\perp)\xrightarrow{0}(m_1,m_2,0,0)\xrightarrow{3}(m_1,m_2,3,3)\xrightarrow{0}\\(\overline{m}_2,m_2,\perp,3)\xrightarrow{0}(\overline{m}_2,\tilde{m}_2,\perp,3)\xrightarrow{0}(m_2,\tilde{m}_2,0,3)\xrightarrow{2}(m_2,\tilde{m}_2,2,3)\xrightarrow{0}\\(\overline{m}_3,\tilde{m}_2,\perp,3)\xrightarrow{0}(\overline{m}_3,m_2,\perp,3)\xrightarrow{0}(m_3,m_2,0,3)\xrightarrow{2}(m_3,m_2,2,5)\xrightarrow{0}\\(m_3,f,2,\perp)\xrightarrow{2}(m_3,f,4,\perp)\xrightarrow{0}(f,f,\perp,\perp)\end{array}$$

$$\xi_2: \begin{array}{l}(\overline{m}_1,\overline{m}_2,\perp,\perp)\xrightarrow{0}(m_1,\overline{m}_2,0,\perp)\xrightarrow{0}(m_1,m_2,0,0)\xrightarrow{3}(m_1,m_2,3,3)\xrightarrow{0}\\(\overline{m}_2,m_2,\perp,3)\xrightarrow{2}(\overline{m}_2,m_2,\perp,5)\xrightarrow{0}(m_2,f,\perp,\perp)\xrightarrow{0}(m_2,f,0,\perp)\xrightarrow{0}\\(m_2,f,2,\perp)\xrightarrow{0}(\overline{m}_3,f,\perp,\perp)\xrightarrow{0}(m_3,f,0,\perp)\xrightarrow{4}(m_3,f,4,\perp)\xrightarrow{0}\\(f,f,\perp,\perp)\end{array}$$

Το πρόβλημα του βέλτιστου χρονοπρογράμματος ενός προεκτοπιστικού Job –shop συστήματος μπορεί να περιοριστεί σε ένα πρόβλημα εύρεσης της συντομότερης διαδρομής σε ένα αυτόματο διακοπτόμενου χρόνου.

Στη προσπάθεια αυτή για την εύρεση της συντομότερης διαδρομής σε ένα αυτόματο που θα μας οδηγήσει και στην εύρεση του βέλτιστου χρονοπρογράμματος προκύπτουν δύο προβλήματα που πρέπει να λάβουμε υπόψη μας:

1. Τα προβλήματα προσπέλασης στα αυτόματα διακοπτόμενων χρόνων είναι γνωστό πως είναι προβλήματα δυσκολίας λήψης αποφάσεων (undecidable).
2. Σε αντίθεση με το ντετερμινιστικό μοντέλο το συνολικό αυτόματο διακοπτόμενου χρόνου στο προεκτοπιστικό είναι κυκλικό και συνεπώς έχουμε άπειρο αριθμό εκτελέσεων.

Στη συνέχεια θα παρουσιαστεί η μεθοδολογία με την οποία μπορούν να ξεπεραστούν τα πιο πάνω δύο προβλήματα αναφορικά με τη βελτιστοποίηση προεκτοπιστικών χρονοπρογραμμάτων.

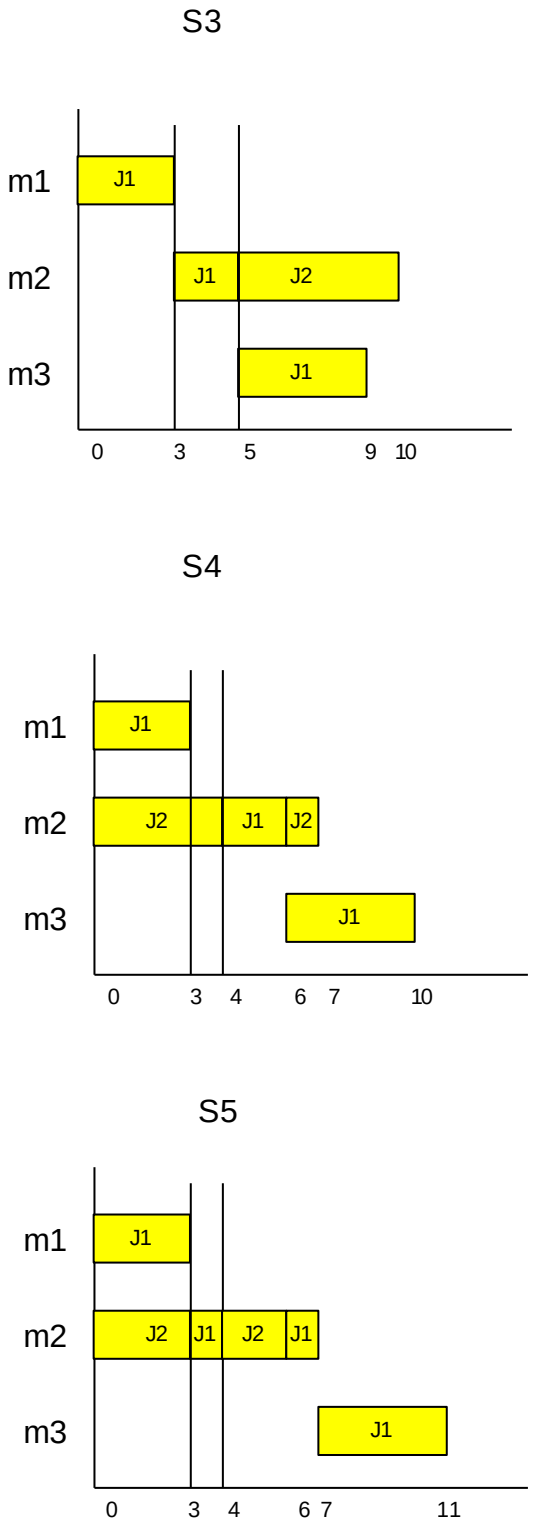
7.2. Αποτελεσματικά χρονοπρογράμματα

Ένα χρονοπρόγραμμα S θεωρείται αποτελεσματικό αν για κάθε εργασία i και βήμα j τέτοιο ώστε $\mu^i(j)=m$, η εργασία i χρησιμοποιεί την μηχανή m κατά τη διάρκεια όλων των χρονικών διαστημάτων E_j^i εκτός όμως εκείνων των χρονικών διαστημάτων όπου μια άλλη εργασία i' τέτοια ώστε $i' \prec_m i$ (η i' έχει προτεραιότητα της i στη μηχανή m) να χρησιμοποιεί την ίδια μηχανή m.

Η έννοια της αποτελεσματικότητας σημαίνει ότι κάθε βήμα (i, j) απασχολεί τη μηχανή πχ. m κατά τη διάρκεια όλων των χρονικών διαστημάτων E_j^i εκτός εκείνων όπου η μηχανή χρησιμοποιείται για την εκτέλεση ενός βήματος το οποίο ολοκληρώνεται νωρίτερα από το (i, j).

Τα χρονοπρογράμματα του Σχήματος 52 αναφέρονται στο παράδειγμα που παρουσιάζουμε και είναι εφικτά χρονοπρογράμματα. Στη συνέχεια θα δείξουμε για ποιο λόγο τα χρονοπρογράμματα αυτά αν και είναι εφικτά εντούτοις είναι μη αποτελεσματικά. Σύμφωνα με τον ορισμό που δόθηκε πιο πάνω και με βάση το ίδιο παράδειγμα μπορούμε να εξάγουμε τη σχέση προτεραιότητας που υπάρχει μεταξύ των εργασιών J^1 και J^2 σχετικά με τη μηχανή m_2 και για τα χρονικά διαστήματα E_2^1 και E_1^2.

- $S_3 : J^1 \prec_{m_2} J^2,\ \mathrm{E}_2^1 = [3,5],\ \mathrm{E}_1^2 = (0,10]$
- $S_4 : J^1 \prec_{m_2} J^2,\ \mathrm{E}_2^1 = [3,6],\ \mathrm{E}_1^2 = (0,7]$
- $S_5 : J^2 \prec_{m_2} J^1,\ \mathrm{E}_2^1 = [3,7],\ \mathrm{E}_1^2 = (0,6]$



Σχήμα 52: Τρία αναποτελεσματικά χρονοπρογράμματα. Το χρονικό διάστημα που δείχνει την αναποτελεσματικότητα φαίνεται από τη διακεκομμένη γραμμή

Στο χρονοπρόγραμμα S_3 η εργασία J^2 μπορεί να κάνει χρήση της μηχανής m_2 στο διάστημα [0,10]. Παρόλα αυτά ξεκινάει σε χρόνο *t*=5 αν και καμία άλλη εργασία δεν χρησιμοποιεί τη προαναφερθείσα μηχανή στο διάστημα [0,3].

Στο χρονοπρόγραμμα S_4 η εργασία J^1 μπορεί να χρησιμοποιήσει τη μηχανή m_2 κατά το χρονικό διάστημα [3,6]. Παρόλα αυτά όμως για το διάστημα [3,4] η εργασία J^2 απασχολεί την μηχανή m_2, ενώ κανονικά η εργασία J^1 έχει προτεραιότητα έναντι της J^2 πάντα για την ίδια μηχανή ($J^1 \prec_{m_2} J^2$).

Στο χρονοπρόγραμμα S_5 η εργασία J^2 μπορεί να κάνει χρήση της μηχανής m_2 κατά το χρονικό διάστημα [0,6]. Στο χρονικό όμως διάστημα [3,4] η εργασία J^1 απασχολεί τη μηχανή ενώ κανονικά ισχύει $J^2 \prec_{m_2} J^1$.

Στη συνέχεια θα δείξουμε πως κάθε πρόβλημα μπορεί να έχει μια βέλτιστη λύση που να προκύπτει μέσα από συγκεκριμένες και καθορισμένες σχέσεις προτεραιότητας μεταξύ των εργασιών και των μηχανών. Οι σχέσεις προτεραιότητας πρέπει να είναι τέτοιες ώστε:

- Το βήμα μιας εργασίας θα πρέπει να εκτελείται σε μια μηχανή το συντομότερο δυνατό εξαιρουμένων των χρονικών διαστημάτων εκείνων όπου έρχεται σε σύγκρουση με κάποια άλλη εργασία με βάση πάντα τις προκαθορισμένες σχέσεις προτεραιότητας.
- Η προεκτόπιση λαμβάνει χώρα όταν ένα βήμα που έχει μια υψηλότερη προτεραιότητα σε σχέση με ένα βήμα που ήδη εκτελείται, μπορεί να μπει σε διαδικασία εκτέλεσης. Το αποτέλεσμα είναι ότι ακολουθώντας αυτούς τους κανόνες και τις διαδικασίες ο αριθμός των προεκτοπίσεων είναι πλέον πεπερασμένος.

7.3. Αναζήτηση αποτελεσματικών εκτελέσεων

Στη συνέχεια θα χρησιμοποιήσουμε την μοντελοποίηση που αναφέρθηκε νωρίτερα για να βρούμε τη συντομότερη διαδρομή σε ένα αυτόματο διακοπτόμενου χρόνου. Σκοπός είναι να επικεντρώσουμε τη προσοχή μας στις εκτελέσεις εκείνες που αντιστοιχούν σε αποτελεσματικά χρονοπρογράμματα.

Μια εκτέλεση σε ένα αυτόματο διακοπτόμενου χρόνου είναι αποτελεσματική αν όλες οι διακριτές μεταβάσεις λαμβάνονται το συντομότερο δυνατό που μπορούν να ληφθούν, ενώ και όλες οι συγκρούσεις επιλύονται σύμφωνα με τις καθορισμένες σχέσεις προτεραιότητας.

Μια αποτελεσματική εκτέλεση συμβάλλει έτσι ώστε:

1. Να περιορίζει την αναζήτηση μόνο στις άμεσες εκτελέσεις
2. Να περιορίζει τον αριθμό των ποιοτικών διαδρομών (qualitative paths) έτσι ώστε να είναι πεπερασμένος και συνεπώς να αποφεύγονται βρόγχοι καθώς και άλλες περιττές για το μοντέλο προεκτοπίσεις και επανενάρξεις.

Έστω για παράδειγμα ότι έχουμε δύο εργασίες, J^1 και J^2, οι οποίες έρχονται σε σύγκρουση σχετικά με τη μηχανή m και η J^1 έχει υψηλότερη προτεραιότητα από τη J^2 $\left(J^1 \preceq_m J^2\right)$ σχετικά με τη μηχανή m. Ο παρακάτω πίνακας δείχνει όλες τις πιθανές καταστάσεις σύγκρουσης μεταξύ των εργασιών και το πως αυτές οι συγκρούσεις επιλύονται.

Επίλυση Συγκρούσεων όταν $J^1 \preceq_m J^2$				
	Κατάσταση	Ενέργεια	Νέα Κατάσταση	Παρατήρηση
1	$(\overline{m}, \overline{m})$	έναρξη της 1	$(m, \overline{m})$	
2	$(\overline{m}, \widetilde{m})$	έναρξη της 1	$(m, \widetilde{m})$	
3	$(\overline{m}, m)$	προεκτόπιση της 2	$(\overline{m}, \widetilde{m})$	
4	$(\widetilde{m}, \overline{m})$	επανέναρξη της 1	$(m, \overline{m})$	
5	$(\widetilde{m}, \widetilde{m})$	επανέναρξη της 1	$(m, \widetilde{m})$	
6	$(\widetilde{m}, m)$			Αδύνατο
7	$(m, \overline{m})$	συνέχιση επεξεργασίας	$(m, \overline{m})$	
8	$(m, \widetilde{m})$	συνέχιση επεξεργασίας	$(m, \widetilde{m})$	
9	(m, m)			Αδύνατο

Ο πίνακας ουσιαστικά εξετάζει όλες τις πιθανές καταστάσεις που μπορούν να υπάρξουν μεταξύ δύο εργασιών και με βάση μια δεδομένη σχέση προτεραιότητας. Συγκεκριμένα οι καταστάσεις 1 και 2 μας λένε ότι η μηχανή m δεν απασχολείται και συνεπώς η εργασία J^1 η οποία βρίσκεται σε αναμονή θα πρέπει να ξεκινήσει την επεξεργασίας της στη m, αφού έχει προτεραιότητα έναντι της J^2. Το ίδιο περίπου συμβαίνει και στις καταστάσεις 4 και 5. Στις περιπτώσεις αυτές η J^1 αναμένει διότι έχει ήδη προεκτοπιστεί. Συνεπώς με βάση πάντα τη σχέση προτεραιότητας πρέπει να γίνει επανέναρξη της εργασίας. Όλες οι καταστάσεις -1,2,4 και 5- μπορούν να

συμβούν είτε στη περίπτωση που μια τρίτη εργασία πχ J^3 με υψηλότερη σχέση προτεραιότητας απελευθερώσει τη μηχανή m είτε στη περίπτωση που η J^1 έχει ολοκληρώσει κάποιο προηγούμενο βήμα της και είναι έτοιμη να μπει στη κατάσταση m. Στη κατάσταση 3 η m επεξεργάζεται την εργασία J^2 και συνεπώς πρέπει να προεκτοπιστεί για να αρχίσει η εκτέλεση της J^1. Η συνέχεια της κατάστασης 3 είναι η κατάσταση 2. Η κατάσταση 6 είναι αδύνατη διότι, λόγω της σχέσης προτεραιότητας δεν είναι δυνατόν να έχει προεκτοπιστεί η J^1 και να εκτελείται η J^2. Οι καταστάσεις 7 και 8 (εκτελείται η J^1) είναι σύμφωνες με τη σχέση προτεραιότητας και συνεπώς δεν χρειάζεται να γίνει καμία προεκτόπιση. Η κατάσταση 9 είναι και αυτή όπως και η 6 αδύνατη διότι θεωρεί ότι οι δύο εργασίες εκτελούνται ταυτόχρονα στην ίδια μηχανή γεγονός που έρχεται σε αντίθεση με τη συνθήκη της αμοιβαίας απόκλισης.

Έστω A είναι ένα αυτόματο διακοπτόμενου χρόνου ενός προεκτοπιστικού Job –shop συστήματος J. Κάθε ολοκληρωμένη αποτελεσματική εκτέλεση του A αντιστοιχεί σε ένα εφικτό αποτελεσματικό χρονοπρόγραμμα με διάρκεια ίση με τη μετρική διάρκεια της εκτέλεσης.

Το πρόβλημα της βελτιστοποίησης ενός προεκτοπιστικού Job –shop χρονοπρογράμματος μπορεί να περιοριστεί σε ένα πρόβλημα εύρεσης της συντομότερης αποτελεσματικής εκτέλεσης σε ένα αυτόματο διακοπτόμενου χρόνου.

Ο περιορισμός και γενικότερα η εστίαση μόνο στις αποτελεσματικές εκτελέσεις καθιστά το πρόβλημα εύρεσης της συντομότερης διαδρομής τέτοιο ώστε να μπορεί να ληφθεί σαφής απόφαση που να οδηγεί στην επίλυσή του. Συγκεκριμένα μπορούν να απαριθμηθούν όλες οι σχέσεις προτεραιότητας και από τη στιγμή που γίνει η απαρίθμηση αυτή μπορεί να ελεγχθεί για κάθε μια πως επηρεάζεται το μήκος της αποτελεσματικής εκτέλεσης

Όπως και στο ντετερμινιστικό μοντέλο έτσι και εδώ εφαρμόζεται ένας αλγόριθμος αναζήτησης στο ανάπτυγμα του αυτομάτου ο οποίος δημιουργεί σχέσεις προτεραιότητας ***εν κινήσει* (*on the* fly)** και αυτό συμβαίνει όποτε δύο εργασίες έρχονται σε σύγκρουση. Επανερχόμενοι στο παράδειγμα του Σχήματος 51 παρατηρούμε ότι μια πρώτη σύγκρουση μεταξύ δύο εργασιών έρχεται στην επιφάνεια στη κατάσταση $(\overline{m}_2, m_2)$. Στη περίπτωση αυτή καλούμαστε να επιλέξουμε μεταξύ δύο εναλλακτικών. Ή αφήνουμε το χρόνο να περάσει και οδηγούμαστε στη κατάσταση $(\overline{m}_2, f)$ όπου η εργασία J^2 ολοκληρώνει την επεξεργασία της στη μηχανή m_2, ή προεκτοπίζουμε την εργασία J^2 και οδηγούμαστε στη κατάσταση

$(\overline{m}_2, \tilde{m}_2)$. Στη πρώτη περίπτωση όπου βέβαια δεν υπάρχει προεκτόπιση η σχέση προτεραιότητας που ισχύει και με βάση αυτή ενεργούμε είναι $J^2 \prec_{m_2} J^1$. Στη δεύτερη περίπτωση (μετακίνηση στη κατάσταση $(\overline{m}_2, \tilde{m}_2)$) η σχέση προτεραιότητας που ισχύει είναι $J^1 \prec_{m_2} J^2$, ενώ η μετακίνηση προς τα πίσω στη κατάσταση $(\overline{m}_2, m_2)$ απαγορεύεται. Από εκεί και μετά η συνέχεια είναι να μετακινηθούμε στη κατάσταση $(m_2, \tilde{m}_2)$ (έναρξη επεξεργασίας της εργασίας J^1 στη m_2) και να αφήσουμε το χρόνο να περάσει μέχρι η J^1 να ολοκληρωθεί και να απελευθερώσει τη m_2.

Για να τυποποιήσουμε όλη τη παραπάνω διαδικασία ορίζουμε και χρησιμοποιούμε σχέσεις ***ακολούθων*** που είναι πλειάδες της μορφής (q, v, t, Π) όπου (q, v, t) είναι ο γενικός σχηματισμός του διευρυμένου αυτομάτου και Π είναι μια μερική σχέση προτεραιότητας. Όταν δεν υπάρχουν αρχικές μεταβάσεις στην κατάσταση (q, v, t) έχουμε:

$$Succ(q, v, t, \Pi) = \{(Succ^t(q, v, t), \Pi)\}$$

όπου $Succ^t(q, v, t)$ είναι η χρονικά ακόλουθη σχέση όπως αυτή ορίζεται στη μη – προεκτοπιστική περίπτωση.

Όταν υπάρχουν αρχικές μεταβάσεις στη (q, v, t) τότε έχουμε:

$$Succ(q, x, \Pi, \theta) = L_1 \cup L_2 \cup L_3$$

όπου

$$L_1 = \{(q', x', \Pi, \theta) : (q, x) \xrightarrow{\delta} (q', x')\}$$

για κάθε άμεση μετάβαση δ τέτοια ώστε η δ να μην έρχεται σε σύγκρουση ή να ανήκει σε εργασία η ο οποία έχει υψηλότερη προτεραιότητα σε μια μηχανή από όλες εκείνες που ανήκουν σε ανταγωνιστικές εργασίες. Αν υπάρξει μια σύγκρουση αναφορικά με μια μηχανή m που εμπλέκει μια εργασία i της οποίας η προτεραιότητα συγκρίνεται με μιας άλλης εργασίας i^* αλλά η υψηλότερη προτεραιότητα δεν έχει ακόμα προσδιοριστεί θα έχουμε:

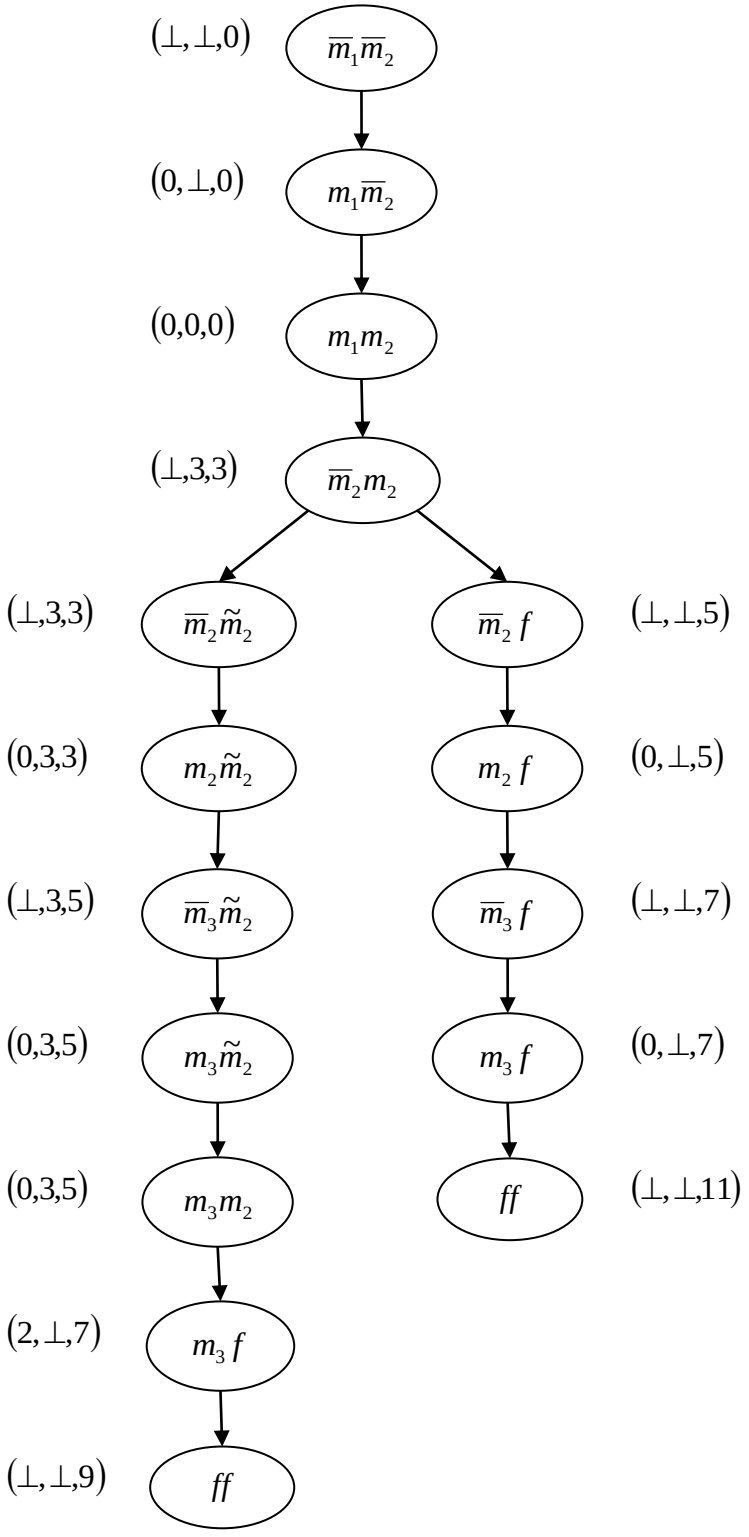
$$L_2 = \left\{\left(q, x, \Pi \cup \left\{i^* \prec i\right\}, \theta\right)\right\}$$

και

$$L_3 = \left\{\left(q, x, \Pi \cup \bigcup_{\{i':i'\|_m i\}} \left\{i \prec i'\right\}, \theta\right)\right\}$$

Η ακόλουθη της L_2 αναπαριστά την επιλογή να προτιμηθεί η i^* έναντι της i. Η σχέση προτεραιότητας της i σχετικά με εργασίες που είναι είτε σε αναμονή είτε είναι ήδη προεκτοπισμένες θα προσδιοριστεί μόνο όταν η i^* ολοκληρωθεί. Η L_3 αναπαριστά την επιλογή να προτιμηθεί η i έναντι όλων των άλλων εργασιών.

Στο Σχήμα 53 παρουσιάζονται οι αποτελεσματικές εκτελέσεις του χρονισμένου αυτομάτου του Σχήματος 51 που δημιουργούνται με βάση τον αλγόριθμο που παρουσιάστηκε παραπάνω.



Σχήμα 53: Οι αποτελεσματικές εκτελέσεις του χρονισμένου αυτομάτου του Σχήματος 51

Κεφάλαιο 8: Χρονοπρογραμματισμός με βάση το κόστος

8.1. Προσθήκη κόστους στα αυτόματα

Τα αυτόματα που θα κατασκευαστούν έχουν ως σκοπό να αντιμετωπίσουν το Job – shop πρόβλημα. Σε αυτά τα αυτόματα το πρόβλημα της προσπέλασης που θα παρουσιαστεί, έχει να κάνει με την ύπαρξη πάντα μιας διαδρομής υπό την απουσία όμως περιορισμών όπως για παράδειγμα χρονικών προθεσμιών (deadlines) κλπ. Η διαδρομή αυτή θα ξεκινάει με μια αρχική κατάσταση (καμία εργασία δεν έχει ξεκινήσει) και θα καταλήγει σε μια τελική κατάσταση (έχει ολοκληρωθεί η επεξεργασία όλων των εργασιών). Βέβαια ο αριθμός των διαφορετικών διαδρομών που μπορεί να προκύψουν σε ένα τέτοιο αυτόματο είναι πάρα πολύ μεγάλος διότι αντιστοιχούν σε διαφορετικά εφικτά χρονοπρογράμματα. Η βελτιστοποίηση του προβλήματος όμως έχει να κάνει με την επιλογή εκείνης της διαδρομής με το συντομότερο συνολικό χρόνο ολοκλήρωσης (makespan). Η επιλογή λοιπόν εκείνης με το συντομότερο χρόνο ολοκλήρωσης η οποία και βελτιστοποιεί το πρόβλημα μπορεί να πραγματοποιηθεί με την εφαρμογή διαφόρων αλγορίθμων συντομότερης διαδρομής σε αυτόματα με βάρη.

Τα αυτόματα με βάρη είναι τα αυτόματα εκείνα τα οποία σχετίζουν κάποιο είδος κόστους (πχ. ενέργεια, χρόνο κλπ.) με τις μεταβάσεις. Αντίστοιχα αν οι κόμβοι σε ένα μοντέλο παριστάνουν φυσικές τοποθεσίες, τότε το βάρος σε μια ακμή έχει να κάνει με το μήκος (απόσταση) μεταξύ των κορυφών. Επίσης το βάρος θα μπορούσε να εκφράζει το χρόνο που απαιτείται για να πάμε από τον ένα κόμβο στον άλλο. Το βεβαρημένο αυτόματο συμβολίζεται ως $A=(Q,\delta,\omega)$, όπου $\omega:\delta\to\Re$ είναι μια συνάρτηση που προσδίδει σε κάθε μετάβαση $(q,q')\in\delta$ ένα βάρος $\omega_{q,q'}$. Για κάθε πεπερασμένη εκτέλεση

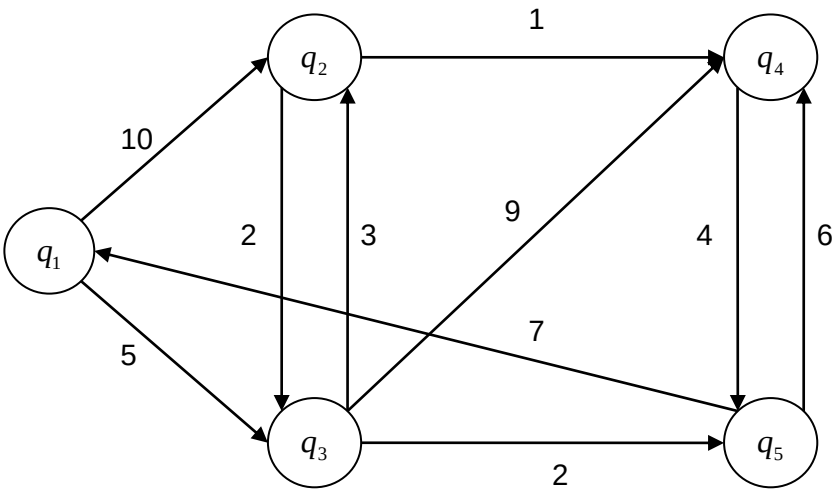
$$q_0 \xrightarrow{\omega_{0,1}} q_1 \xrightarrow{\omega_{1,2}} q_2 \xrightarrow{\omega_{2,3}} \ldots \xrightarrow{\omega_{k-1,k}} q_k$$

σχετίζουμε ένα κόστος το οποίο είναι

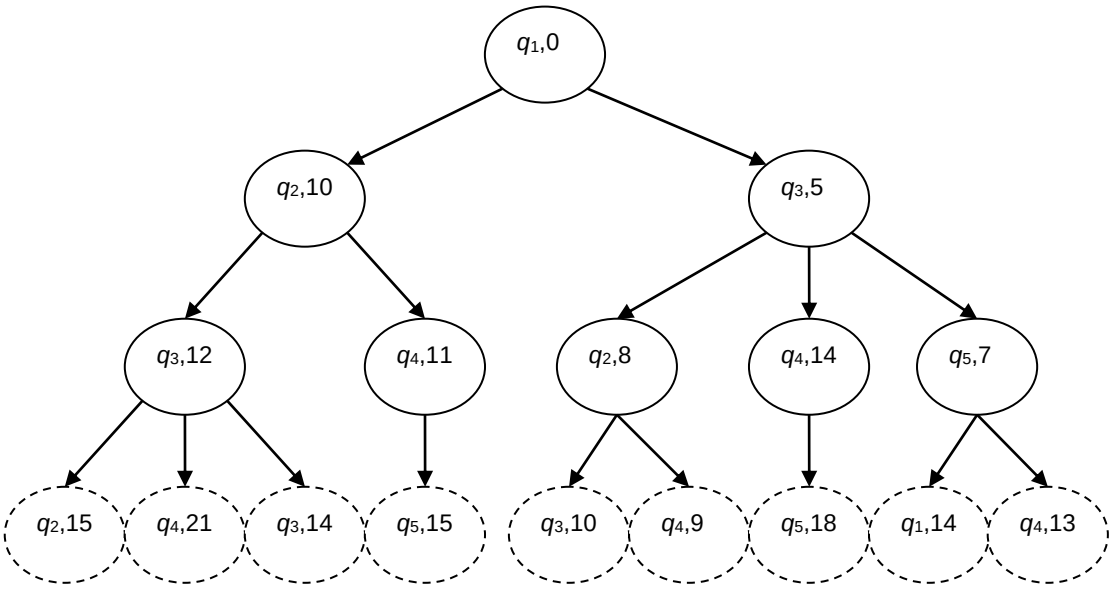
$$C=\omega_{0,1}+\omega_{1,2}+\omega_{2,3}+\ldots+\omega_{k-1,k}$$

Η συντομότερη διαδρομή μεταξύ της αρχικής κατάσταση q και της τελικής κατάστασης q' είναι η εκτέλεση με το ελάχιστο C. Θα υποθέτουμε ότι το βάρος ω είναι πάντα θετικό και συνεπώς η συντομότερη διαδρομή μεταξύ των δύο καταστάσεων του αυτομάτου (αρχικής και τελικής) δεν έχει κύκλους.

Το q-ανάπτυγμα ενός αυτομάτου με βάρη είναι ένα δέντρο το οποίο φέρει ετικέτες που είναι ζεύγη στο $Q\times\Re_+$. Ξεκινάει με την ετικέτα $(q,0)$ και στη συνέχεια δημιουργεί για κάθε κόμβο (p,r) ακολούθους κόμβους της μορφής (p',r') τέτοιους ώστε $(p,p')\in\delta$ και $r'=r'+\omega_{p,p'}$. Για κάθε κόμβο (p,r) στο δέντρο, το r αντιστοιχεί στην απόσταση μεταξύ των q και p στην αντίστοιχη διαδρομή. Ένα τέτοιο αυτόματο και το ανάπτυγμά του όπως περιγράφηκε παραπάνω παρουσιάζονται στο Σχήμα 54 και στο Σχήμα 55 αντίστοιχα. Η εύρεση της συντομότερης διαδρομής μεταξύ των q και q' είναι στην ουσία ισοδύναμη με την εύρεση του κόμβου (q',r) με το μικρότερο r στο q-ανάπτυγμα του αυτομάτου.

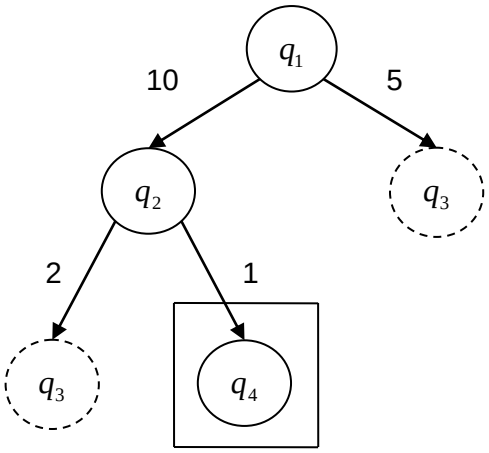


Σχήμα 54: Το αυτόματο $A=(Q,\delta,\omega)$ ***με βάρη***



Σχήμα 55: Το q1-ανάπτυγμα του αυτομάτου με βάρη

Η εφαρμογή του αλγορίθμου BFS θα σταματήσει αφού προσπελαστεί η q' ενώ υπάρχει συντομότερη διαδρομή στη q' με περισσότερες μεταβάσεις. Για παράδειγμα ο BFS μπορεί να προσπελάσει τη q_4 ακολουθώντας τη διαδρομή $q_1 \xrightarrow{10} q_2 \xrightarrow{1} q_4$, ενώ υπάρχει συντομότερη διαδρομή παρά το γεγονός ότι περιέχει περισσότερες μεταβάσεις και είναι η $q_1 \xrightarrow{5} q_3 \xrightarrow{3} q_2 \xrightarrow{1} q_4$.



Σχήμα 56: Η εφαρμογή του BFS αλγορίθμου στο αυτόματο με βάρη

Ένας πολύ γνωστός αλγόριθμος ο οποίος εφαρμόζεται σε γράφους με θετικά βάρη και μας δίνει τη συντομότερη διαδρομή είναι αυτός του Dijkstra. Ο αλγόριθμος του Dijkstra περιλαμβάνει μια σειρά επαναλήψεων μιας συνάρτηση $d: Q \to \Re_+$ τέτοια ώστε κάθε επανάληψη i, $d^i(p)$ είναι το μήκος της μακρύτερης διαδρομής στη κατάσταση p έχοντας το πολύ i μεταβάσεις. Αν η p δεν είναι προσπελάσιμη μέσα σε i βήματα τότε $d^i(p) = \infty$. Όταν ο αλγόριθμος τερματίζει τότε η $d(p)$ κρατά τη διάρκεια της συντομότερης διαδρομής στη κατάσταση p.

Αλγόριθμος 3 (Ο Αλγόριθμος του Dijkstra (*A*,*q*))

$d^0(p) = 0;$
$\forall p \neq q \;\; d^0(p) = \infty;$
$i = 0;$
Repeat
begin
$i := i + 1;$
$\forall p \in Q;$
$d^i(p) = \min\left(\left\{\left[d^{i-1}(p') + W_{pp'}\right] : p' \in \Pr e(p)\right\} \cup \left\{d^i(p)\right\}\right);$
end *Until* $d^i = d^{i+1}$

8.2. Ανάπτυξη μοντέλου στο Uppaal

Στο προηγούμενο κεφάλαιο περιγράφηκαν τρόποι επίλυσης του προβλήματος χρονοπρογραμματισμού, όπως επίσης και βελτιώσεις που λαμβάνουν υπόψη περισσότερα δεδομένα (όπως κόστη, ημερομηνίες), ώστε να υπάρχει μια πιο ρεαλιστική αποτύπωση των διαδικασιών και η προτεινόμενη λύση να αντιπροσωπεύει μια οικονομικότερη λύση.

Στο συγκεκριμένο μοντέλο έχουν ληφθεί υπόψη το κόστος αποθήκευσης σε περίπτωση που οι εργασίες περατωθούν ταχύτερα από το προβλεπόμενο, η ζημιά που προκαλείται (φήμη, ρήτρες, ποιότητα κλπ) εάν καθυστερήσει η περάτωση ή παράδοση κάποιας εργασίας. Για τη μείωση του κόστους, λαμβάνεται επίσης υπόψη το κόστος αδράνειας των μηχανών μεταξύ εργασιών, αλλά και ο συνολικός χρόνος περάτωσης των εργασιών.

8.2.1. Δομικά στοιχεία μοντέλου

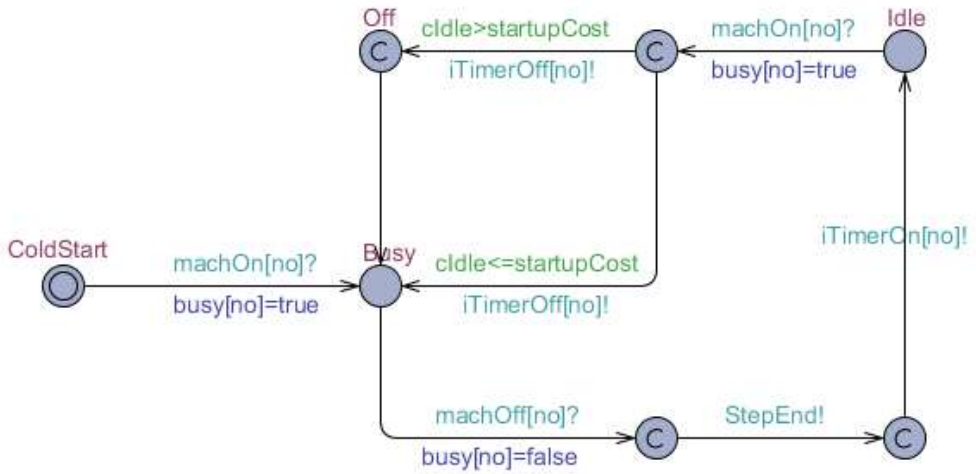
8.2.1.1. Η μηχανή

Για τα δεδομένα του προβλήματος θεωρούμε ότι η μηχανή μπορεί να βρίσκεται ουσιαστικά σε 3 καταστάσεις: Busy (απασχολημένη), Idle (σε ηρεμία) και Off (κλειστή).

Στην μοντελοποίηση βέβαια χρησιμοποιούμε παραπάνω διακριτές καταστάσεις οι οποίες βοηθούν στον καλύτερο έλεγχο της ροής του προγράμματος. Έτσι, ορίζεται επιπλέον η κατάσταση ColdStart η οποία υποδηλώνει την πρώτη εκκίνηση της μηχανής και 3 ακόμα committed καταστάσεις οι οποίες κυρίως βοηθούν να υπάρχει σύγχυση για τη σειρά των πράξεων και των διαδικασιών μέσω των καναλιών συγχρονισμού.

Ανάλογα με τον αριθμό των μηχανών που έχουν δηλωθεί στο System Declarations (Δηλώσεις Συστήματος), το πρόγραμμα πρέπει να δημιουργήσει αντίστοιχο αριθμό εικονικών μηχανών. Αυτό επιτυγχάνεται με την εισαγωγή σταθεράς παραμέτρου (*const int no*) ως αριθμό-ταυτότητα για κάθε μηχανή.

Στο αυτόματο που ακολουθεί, η παράμετρος *no* θα αντικαθίσταται κάθε φορά από τον αριθμό κάθε μηχανής.



Σχήμα 57: Αυτόματο μηχανής

Κάθε μηχανή ξεκινά από την κατάσταση ColdStart.

Σημείωση: Η αρχική κατάσταση συμβολίζεται με δύο ομόκεντρους κύκλους στο Uppaal.

Μόλις δοθεί η εντολή (*machOn[no]!*) να ξεκινήσει η μηχανή, τότε αυτή μεταβαίνει στην κατάσταση Busy, ενημερώνοντας πρώτα τη μεταβλητή *busy[no]* σε *true*.

Σημείωση: Ό,τι απεικονίζεται με γαλάζιο χρώμα αφορά σε κανάλι συχρονισμού. Στα κανάλια (chan) ο συγχρονισμός επιτυγχάνεται από την εντολή με το θαυμαστικό (!) στο τέλος προς αυτή με το ερωτηματικό (?) και όχι αντίστροφα.

Η μεταβλητή *busy[no]* χρησιμεύει στην αναγνώριση, από τις άλλες εργασίες που θέλουν να χρησιμοποιήσουν τη μηχανή, ότι είναι απασχολημένη.

Όσο βρίσκεται στην κατάσταση *Busy* αναμένει πότε θα δοθεί η εντολή *machOff[no]!*, ώστε να συγχρονίσει με την *machOff[no]?* και να οδηγηθεί στην αποδέσμευση από τη συγκεκριμένη εργασία.

Μόλις ενεργοποιηθεί η *machOff[no]?*, τότε ενημερώνεται η μεταβλητή *busy[no]* σε *false* και ενεργοποιείται ακαριαία συγχρονισμός του *StepEnd*. Το *StepEnd* δίνει σήμα σε όλες τις εργασίες ότι κάποια μηχανή σταμάτησε τη λειτουργία οπότε μπορεί να χρησιμοποιηθεί από άλλη εργασία.

Η ενδιάμεση κατάσταση χρησιμοποιείται για να ξεχωρίσει τους δύο συγχρονισμούς που πραγματοποιούνται. Εάν χρησιμοποιούσαμε στο ίδιο βήμα το *machOff[no]?* και

StepEnd! η μηχανή μπορεί να έφευγε από την κατάσταση *Busy* χωρίς να έχει τελειώσει η εργασία, δημιουργώντας πιθανή επικάλυψη (overlapping) μεταξύ των δύο εργασιών.

Σημείωση: Το "C" μέσα στον κύκλο προέρχεται από τη λέξη Committed υποδηλώνοντας ότι η συγκεκριμένη κατάσταση πρέπει να πραγματοποιηθεί ακαριαία, χωρίς να σπαταληθεί χρόνος συστήματος. Εάν δε φύγει από μία Committed κατάσταση, το πρόγραμμα δεν μπορεί να συνεχίσει.

Αμέσως επόμενο βήμα είναι ο συγχρονισμός του καναλιού iTimerOn[no]! που δίνει σήμα ότι η μηχανή δεν είναι πλέον απασχολημένη και μετρά εάν συμφέρει η μηχανή να σβήσει (δεδομένου ότι υπάρχει κόστος έναρξης) ή να παραμείνει σε κατάσταση αναμονής μέχρι να ξαναχρειαστεί να χρησιμοποιηθεί.

Ο έλεγχος γίνεται μέσω ενός ρολογιού (clock cIdle;) το οποίο συγκρίνει πόση ώρα πρόκειται να μείνει ανενεργή η μηχανή με την παράμετρο startupCost, που υποδηλώνει ουσιαστικά το κόστος που απαιτείται για να ξεκινήσει η μηχανή μεταφρασμένο σε χρόνο για να είναι εφικτή η σύγκριση.

Η εντολή για να ξαναχρησιμοποιηθεί η μηχανή δίνεται μέσω του καναλιού machOn[no]? και τίθεται αυτόματα η μεταβλητή busy[no]=true, ώστε να φαίνεται απασχολημένη η μηχανή και να μην κληθεί από άλλη εργασία.

Αν και στην πράξη ο έλεγχος -για το αν η μηχανή πρέπει να σβήσει ή όχι για την εξοικονόμηση χρήματος- εκτελείται πριν ξανανοίξει η μηχανή, στην μοντελοποίηση έχουμε χρησιμοποιήσει τον έλεγχο ακριβώς μετά την έναρξη της μηχανής για άλλη εργασία, συνδέοντάς τον με μια ενδιάμεση κατάσταση committed (ώστε να μην υπάρχει σπατάλη χρόνου).

Αυτό, μας δίνει τη δυνατότητα να δημιουργήσουμε ένα πιο απλό μοντέλο μηχανής, το οποίο όμως δεν υπολείπεται σε λειτουργικότητα.

8.2.1.2. Η εργασία

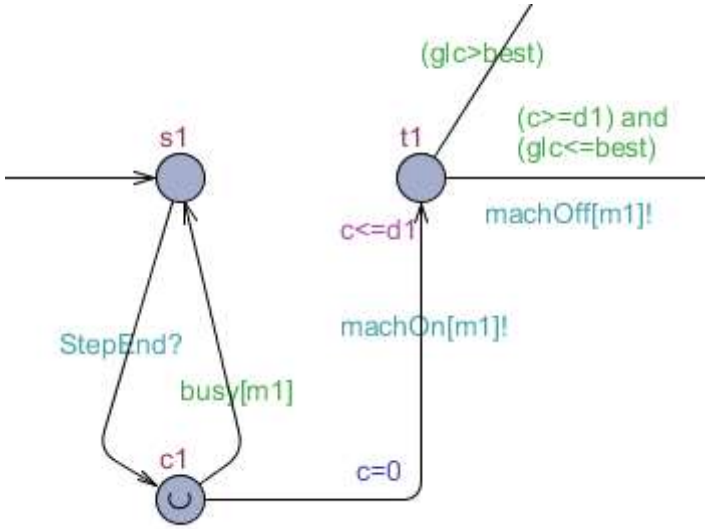
Η κάθε εργασία αποτελεί μια αλληλουχία ενεργειών που πρέπει να γίνουν στις μηχανές, με συγκεκριμένη σειρά ώστε να ολοκληρωθεί.

Για μεγαλύτερη χρηστικότητα και ευκολία κατανόησης και συντήρησης του προγράμματος, επιλέξαμε να χρησιμοποιήσουμε ένα μόνο πρότυπο αυτομάτου για

όλες τις εργασίες, στο οποίο θα μεταβάλλεται η σειρά των μηχανών που πρέπει να χρησιμοποιηθούν, ανάλογα με τις δηλώσεις που γίνονται στο System Declarations.

Για παράδειγμα στην παρακάτω εικόνα, το *m1* δεν αντιπροσωπεύει τη μηχανή 1, αλλά θα αντικατασταθεί με διαφορετικό (πιθανά) αριθμό μηχανής για κάθε εργασία. Υποδηλώνει τη δεύτερη μηχανή κατά σειρά που θα χρησιμοποιήσει η κάθε εργασία.

Σημείωση: Η αρίθμηση ξεκινά από το 0, οπότε το m1 συμβολίζει τη δεύτερη μηχανή και όχι την πρώτη.



Θα εξετάσουμε τη βασική λειτουργία της εργασίας όταν χρησιμοποιεί τη δεύτερη κατά σειρά μηχανή. Με αλληλουχίες της παραπάνω διάταξης μπορούμε να δημιουργήσουμε ένα template με όσες μηχανές επιθυμούμε για κάθε εργασία.

Χρησιμοποιώντας τη μηχανή βλέπουμε ότι υπάρχουν τρεις καταστάσεις για να αναπαραστήσουν την πραγματική λειτουργία.

- Στην κατάσταση s (*s1 στην συγκεκριμένη περίπτωση*) ξεκινά το πρόγραμμα να προσπαθεί να χρησιμοποιήσει την *m1*.
- Στην κατάσταση ελέγχου (control) c ελέγχεται εάν η επιθυμητή μηχανή είναι ελεύθερη για χρήση.
- Τέλος, στην κατάσταση t (time), η εργασία περιμένει όσο χρόνο χρειάζεται για να ολοκληρωθεί η χρήση της *m1* και να αποδεσμευτεί.

Αναλυτικά:

Φτάνοντας στην κατάσταση s1, το πρόγραμμα αναμένει να δοθεί σήμα ότι κάποια μηχανή έχει τελειώσει τη λειτουργία της, μέσω του καναλιού *StepEnd*.

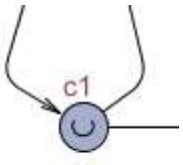
Σημείωση: Έχουμε χρησιμοποιήσει ένα μόνο κανάλι ελέγχου για το τέλος εργασίας των μηχανών, ενώ θα μπορούσε να έχει χρησιμοποιηθεί ξεχωριστό κανάλι για κάθε μηχανή. Με αυτόν τον τρόπο θα μειώναμε τους ελέγχους που πρέπει να πραγματοποιηθούν, αυξάνοντας την ταχύτητα του προγράμματος.

Η υλοποίηση που έχουμε επιλέξει δημιουργεί ένα ευανάγνωστο και λιγότερο περίπλοκο πρόγραμμα, εστιάζοντας στην ουσία του προβλήματος. Στην πράξη οι πόροι που δεσμεύονται είναι ελάχιστοι, χωρίς να επηρεάζουν το τελικό αποτέλεσμα.

Στην κατάσταση c1, ελέγχεται εάν η μεταβλητή *busy[m1]* έχει την τιμή *true*. Εάν ναι, υποδεικνύεται στο πρόγραμμα ότι η μηχανή είναι απασχολημένη και μεταβαίνει πίσω στην κατάσταση s1 αναμένοντας κάποια άλλη μηχανή να τελειώσει τη λειτουργία της.

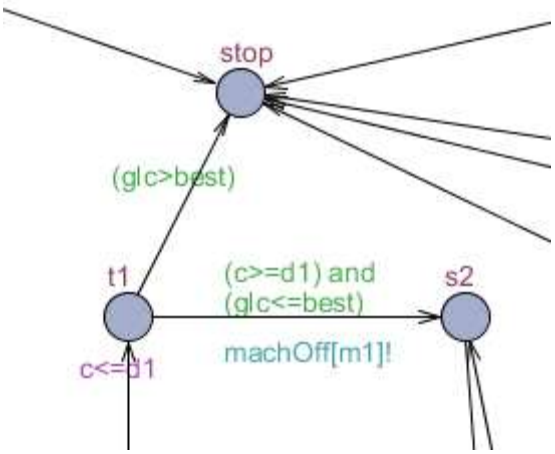
Έχοντας φτάσει ξανά στην κατάσταση c1 και εάν *busy[m1]=false*, το πρόγραμμα μεταβαίνει στην κατάσταση t1, ενημερώνοντας το ρολόι (clock c) με την τιμή 0.

Σημείωση: Το "U" στην κατάσταση c1 υποδηλώνει ότι είναι επείγον (Urgent) να φύγει από την κατάσταση αυτή το πρόγραμμα. Χρησιμοποιείται συνήθως όταν δεν υπάρχουν guards ή εντολές συγχρονισμού που να επιβάλλουν συγκεκριμένη χρονική στιγμή αποχώρησης από την κατάσταση.



Το ρολόι c χρησιμοποιείται ως τοπική μεταβλητή στο template Job για να καταγράφει πόσος χρόνος έχει δαπανηθεί για την ολοκλήρωση των ενεργειών σε κάθε μηχανή.

Παράλληλα ενεργοποιείται συγχρονισμός με το κανάλι machOn[m1] για να ξεκινήσει τη λειτουργία της η μηχανή. Φτάνοντας στην t1 θα παραμείνει σε αυτή την κατάσταση όσο χρόνο έχει ορισθεί από τη σταθερά d1 (const int d1).



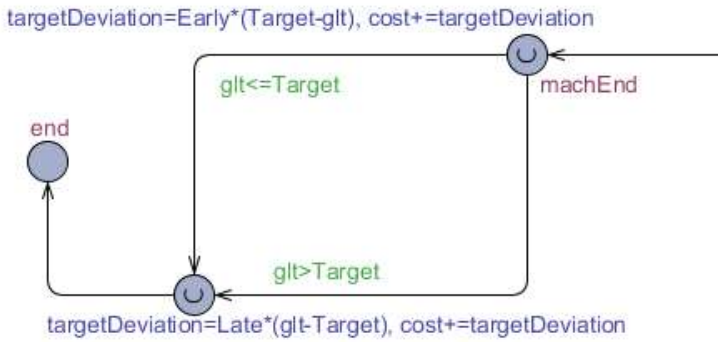
Στο σημείο αυτό πραγματοποιείται έλεγχος εάν ο χρόνος που έχει ήδη δαπανηθεί είναι μεγαλύτερος από ένα άνω όριο που έχει θέσει ο χρήστης (int best;). Το glc (global clock) μετράει τον συνολικό χρόνο από τη στιγμή εκκίνησης, οπότε εάν ξεπεράσει την τιμή best η εργασία οδηγείται στην κατάσταση stop όπου και σταματάει.

Σημείωση: Στην πράξη, για την εύρεση της βέλτιστης λύσης εκτελούμε ένα ερώτημα το οποίο έχει σαν προϋπόθεση όλες οι εργασίες να φτάσουν στην κατάσταση end. Το γεγονός ότι μεταβαίνει στην κατάσταση Stop δίνει τη δυνατότητα στη μηχανή βελτιστοποίησης να καταλάβει ότι η συγκεκριμένη πιθανή λύση δεν πρόκειται να φτάσει στην κατάσταση end, οπότε μεταβαίνει στην επόμενη πιθανή λύση, εξοικονομώντας υπολογιστικό χρόνο.

Εφόσον δε θέλουμε να περιοριστεί το εύρος λύσεων, μπορεί να τεθεί μια πολύ υψηλή τιμή στη μεταβλητή best ώστε να εξεταστούν όλες οι ενδεχόμενες λύσεις και το πρόγραμμα να προτείνει διαδρομή βάσει αυτών.

Δεδομένου λοιπόν ότι έχει παρέλθει ο απαιτούμενος χρόνος d1 και πως ο συνολικός χρόνος glc είναι μικρότερος από αυτόν που έχει ορισθεί από τον χρήστη, δίδεται εντολή στη μηχανή m1 να σβήσει. (machOff[m1]!).

Η ίδια διαδικασία εκτελείται και για τις υπόλοιπες μηχανές, ώσπου η εργασία φτάνει στην κατάσταση machEnd.



Στην κατάσταση αυτή θα γίνει έλεγχος εάν υπάρχει απόκλιση από το χρονικό όριο που είχε τεθεί για την ολοκλήρωση της παραγγελίας (const int Target).

Στην περίπτωση που η εργασία τελειώσει νωρίτερα από το αναμενόμενο, τότε υπάρχει κάποιο κόστος αποθήκευσης ανάλογο της χρονικής απόκλισης από τον στόχο. Οπότε, ορίζουμε μια μεταβλητή targetDeviation στην οποία θα αποθηκεύεται το κόστος της χρονικής απόκλισης.

Δίνονται δύο εντολές:

- targetDeviation=Early*(Target-glt)
- cost+=targetDeviation

όπου Early η παράμετρος που δηλώνει το κόστος αποθήκευσης για κάθε προϊόν ανά μονάδα χρόνου, Target η σταθερά που δηλώνει τον αναμενόμενο χρόνο ολοκλήρωσης και glt (global time) ο συνολικός χρόνος από την έναρξη του προγράμματος.

Σημείωση: Οι μεταβλητές glc και glt έχουν την ίδια τιμή. Η μόνη διαφορά είναι ότι η πρώτη είναι τύπου ρολογιού, ενώ η δεύτερη είναι ακέραια μεταβλητή. Τα ρολόγια μπορούν να χρησιμοποιηθούν για συγκρίσεις μεταξύ τους και με ακεραίους, αλλά μόνο οι ακέραιες μεταβλητές μπορούν να χρησιμοποιηθούν για αριθμητικές πράξεις.

Η πρώτη εντολή αποθηκεύει στην μεταβλητή targetDeviation το γινόμενο της χρονικής απόκλισης με το κόστος αποθήκευσης ανά μονάδα χρόνου, ενώ η δεύτερη εντολή αποθηκεύει το αποτέλεσμα στην μεταβλητή cost.

Σημείωση: Η μεταβλητή cost είναι ενσωματωμένη του προγράμματος Uppaal Cora και χρησιμοποιείται για την εισαγωγή κόστους στα χρονισμένα αυτόματα. Ο λόγος που χρειάζεται ενδιάμεση μεταβλητή (targetDeviation), αντί να χρησιμοποιηθεί κατευθείαν η εντολή cost+=Early(Target-glt), είναι λόγω συντακτικού του Uppaal Cora που δε*

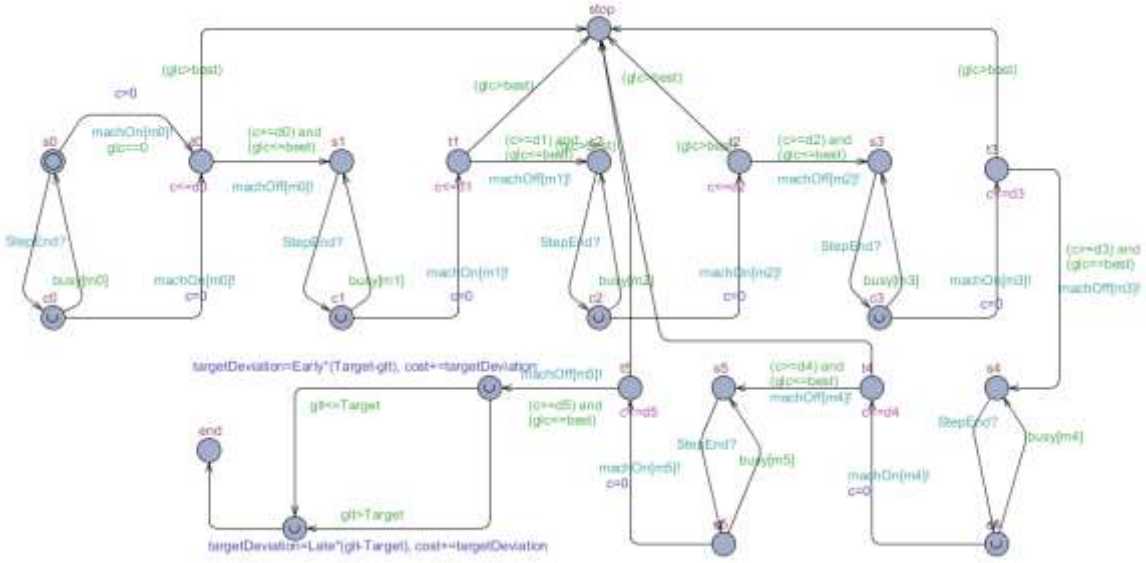
μπορεί να πραγματοποιήσει αριθμητική πράξη και μετά να την καταχωρήσει στην μεταβλητή cost.

Αντίστοιχα στην περίπτωση που η εργασία τελειώσει πια αργά από το αναμενόμενο, τότε υπάρχει κάποια ποινή που επιβάλλεται στο εργοστάσιο. Συνήθως αυτή προβλέπεται από τη σύμβαση με τους προμηθευτές. Στο κόστος αυτό θα μπορούσε να περιληφθεί η πιθανή ζημιά των προϊόντων σε περίπτωση που έχουν μικρή διάρκεια ζωής, ή και ακόμη η ζημιά στη φήμη της επιχείρησης.

Οι μόνες αλλαγές που χρειάζονται στην περίπτωση βραδείας περάτωσης της εργασίας είναι στη μεταβλητή Late, που δηλώνει το κόστος απόκλισης ανά μονάδα χρόνου και στον υπολογισμό χρόνου απόκλισης, ώστε να μη βγαίνει αρνητικός αριθμός.

- targetDeviation=Late*(glt-Target)

Όταν υπολογιστεί και αυτό το κόστος, η εργασία οδηγείται στο τέλος (κατάσταση end), μέσω μιας ενδιάμεσης επείγουσας (urgent) κατάστασης.



Σχήμα 58: Το βασικό template της εργασίας

8.2.1.3. Μετρητές

Χρησιμοποιούμε 3 μετρητές χρόνου/κόστους για να μπορούμε εύκολα να προσαρμόσουμε το μοντέλο ανάλογα με τα δεδομένα. Σε κάθε υλοποίηση, μπορούμε να επιλέξουμε να μη χρησιμοποιήσουμε κάποιον από τους μετρητές, απλά θέτοντας μηδενική τιμή στις μεταβλητές.

Ακόμη και αν δεν έχουμε ακριβείς μετρήσεις για τα κόστη, προσαρμόζοντας τις τιμές των μετρητών είναι σαν να θέτουμε “βάρη” ανάλογα με τη σημαντικότητα. Αυτό, σε συνδυασμό με τις δηλώσεις σταθερών παραμέτρων των μηχανών και των εργασιών μας δίνει πλήρη ευελιξία και ένα ευρύ φάσμα μοντέλων.

Cost Timer

Ο μετρητής κόστους προσθέτει κάποια τιμή στο συνολικό κόστος σε κάθε χρονικό βήμα (5 στο παράδειγμα). Χρησιμοποιείται ώστε να περιοριστεί ο συνολικός χρόνος εκτέλεσης των εργασιών, αφού όταν επιτυγχάνεται βελτίωση ως προς αυτή τη μέτρηση (χωρίς να λαμβάνονται υπόψη άλλοι μετρητές ή παράγοντες που αυξάνουν το κόστος), σημαίνει ότι έχει βρεθεί η βέλτιστη λύση όσον αφορά στον συνολικό χρόνο.

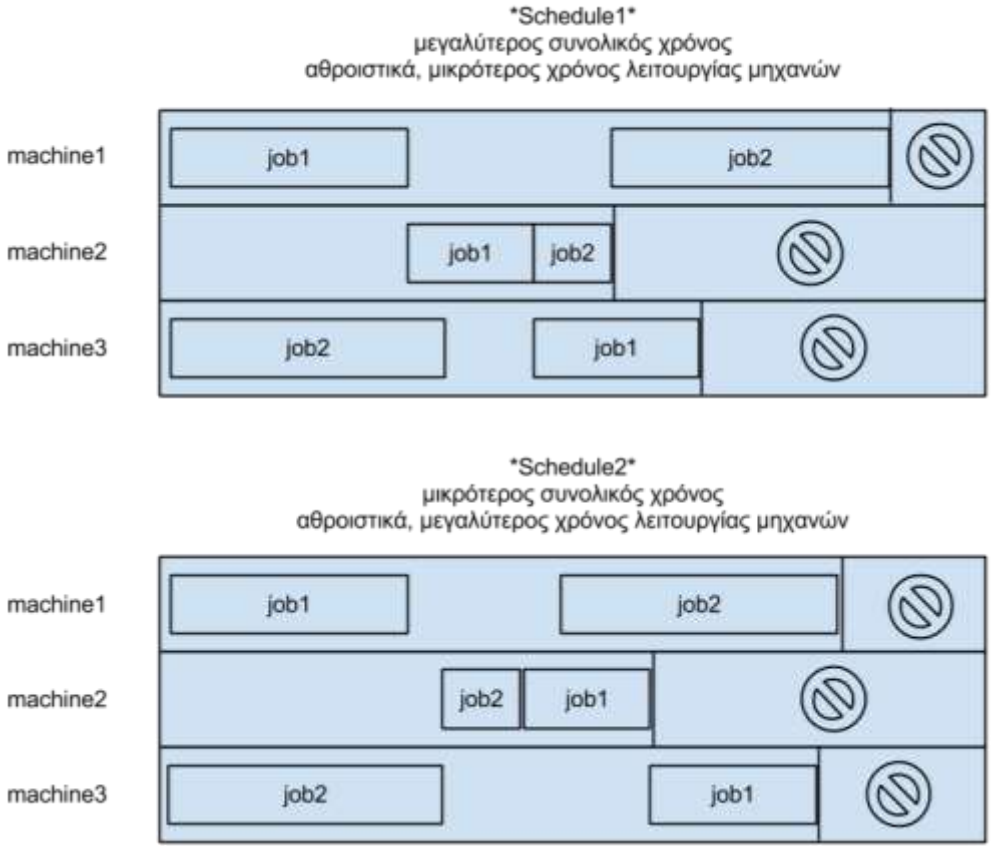
Η τιμή που αυξάνει σε κάθε χρονική μονάδα το κόστος ουσιαστικά δεν έχει σημασία εάν δεν υπάρχουν άλλοι παράγοντες, αφού

*συνολικό κόστος = κόστος ανά χρονική μονάδα * συνολικός χρόνος*

άρα το κόστος ανά μονάδα ως απλός παράγοντας δεν επηρεάζει τον τρόπο βελτίωσης του συνολικού κόστους.

Παρόλα αυτά όμως, αλλάζοντας την τιμή κόστους ανά χρονική μονάδα, όταν υπάρχουν και άλλοι παράγοντες που επηρεάζουν το κόστος, αλλάζει τη βαρύτητα που δίνεται στον συνολικό χρόνο εκτέλεσης έναντι των άλλων παραμέτρων.

Για παράδειγμα, στο παρακάτω σχήμα φαίνεται πώς εάν ο συνολικός χρόνος λειτουργίας των μηχανών είναι σημαντικότερος έναντι του συνολικού χρόνου λειτουργίας, τότε βελτιστοποιώντας μόνο ως προς τον συνολικό χρόνο, μπορεί να απορρίπταμε υλοποιήσεις που θα είχαν ακόμη μικρότερο κόστος.



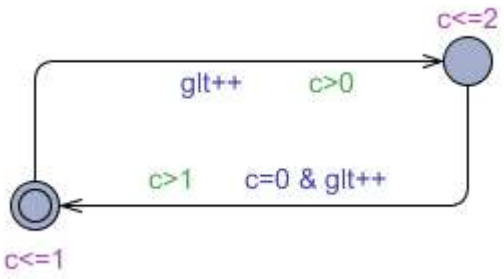
Σχήμα 59: Αναπαράσταση προβλήματος 2χ3

Στο Σχήμα 59 γίνεται ενδεικτική αναπαράσταση προβλήματος 3 μηχανών και 2 εργασιών. Στην πράξη όσο αυξάνονται οι μηχανές και οι εργασίες, τόσο οι διάφορες παράμετροι επηρεάζουν το συνολικό κόστος και άρα η βαρύτητα που δίνεται σε κάθε παράμετρο έχει μεγαλύτερη σημασία και επίδραση στο συνολικό αποτέλεσμα.

Timer

Ο απλός μετρητής χρόνου σχεδιάστηκε για να καλύψει μια έλλειψη του προγράμματος μοντελοποίησης. Το Uppaal διαχωρίζει πλήρως τον χρόνο από το κόστος και δε δίνει τη δυνατότητα να αντιστοιχιστούν χρονικές τιμές (ρολογιών) στη μεταβλητή κόστους. Στην πράξη το κόστος (πχ αποθήκευσης) είναι άρρηκτα συνδεδεμένο με τον χρόνο.

Σχεδιάζοντας έναν μετρητή ο οποίος μετατρέπει κάθε χρονική μονάδα σε μονάδα κόστους, λύνει αυτό το πρόβλημα. Έπειτα ο χρόνος αυτός πολλαπλασιάζεται με το κόστος αποθήκευσης (ή αργοπορίας) ανά μονάδα χρόνου και το αποτέλεσμα είναι το συνολικό κόστος που αναζητούμε.



Σχήμα 60: Timer

Αρχική κατάσταση θεωρείται αυτή με τους δύο ομόκεντρους κύκλους. Το clock c είναι ένα τοπικό ρολόι που χρησιμοποιείται για να εξασφαλίζει ότι για κάθε χρονικό βήμα (του global clock glc) θα αυξάνεται κατά μια μονάδα και το glt.

Σημειώνεται πως το glt είναι καθολική ακέραια μεταβλητή.

Περιγραφή λειτουργίας χρονομετρητή

Το ρολόι c ξεκινάει με την τιμή 0 (προεπιλογή σε όλα τα ρολόγια). Ξεκινώντας από την αρχική κατάσταση υπάρχει περιορισμός c<=1 (invariant). Αυτό σημαίνει ότι δε μπορεί να περιμένει στην κατάσταση αυτή για τιμή μεγαλύτερη του c από 1. Άρα οι πιθανές τιμές αποχώρησης είναι 0, 1.

Χρησιμοποιώντας και την προϋπόθεση (guard) c>0, εξασφαλίζουμε ότι το πρόγραμμα θα περιμένει ακριβώς 1 χρονική μονάδα στην κατάσταση αυτή πριν αποχωρήσει.

Αποχωρώντας από την αρχική κατάσταση αυξάνεται το glt (που είχε αρχική τιμή 0) κατά μια μονάδα.

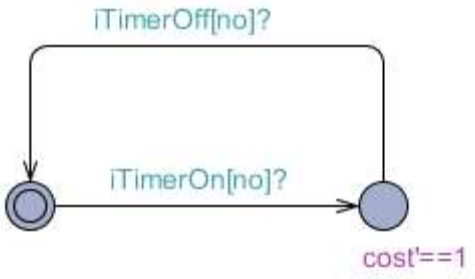
Αντίστοιχα στην επόμενη κατάσταση χρησιμοποιείται invariant c<=2 και guard c>1 ώστε να διασφαλιστεί ότι θα περιμένει στην κατάσταση αυτή ακριβώς μια χρονική μονάδα.

Αποχωρώντας από την κατάσταση αυτή αυξάνεται επίσης κατά μια μονάδα το glt και το τοπικό ρολόι c μηδενίζεται ώστε να ξανακάνει τη διαδικασία αυτή ξανά και ξανά, έως ότου οι εργασίες φτάσουν στην κατάσταση end.

Idle Timer

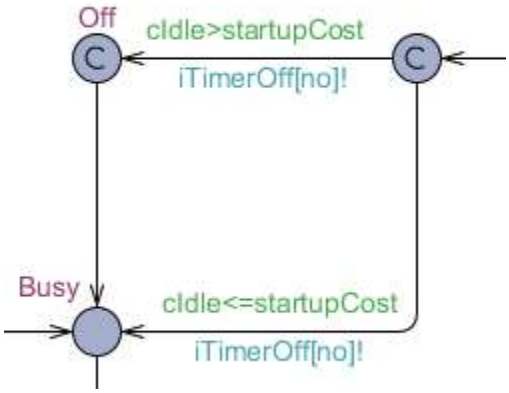
Ο μετρητής αυτός χρησιμοποιείται για να υπολογίζει τον χρόνο που κάθε μηχανή μένει ανενεργή και εάν χρειάζεται αποφασίζει το πρόγραμμα εάν συμφέρει να σβήσει τη μηχανή ή όχι.

Στην ουσία πρόκειται για ένα σύνολο μετρητών, ένα για κάθε μηχανή που χρησιμοποιείται (6 στην συγκεκριμένη περίπτωση).



Σχήμα 61: Idle Timer

Η λειτουργία του είναι σχετικά απλή. Κάθε φορά που δίνεται εντολή από κάποια μηχανή να ενεργοποιηθεί ο Idle Timer, συγχρονίζει το κανάλι iTimerOn[no]? και το πρόγραμμα μεταβαίνει στην επόμενη κατάσταση, όπου το κόστος αυξάνεται κατά μια μονάδα (cost'==1) για κάθε χρονική μονάδα που παραμένει στην κατάσταση.



Αντίστοιχα, όταν δοθεί η εντολή να σβήσει ο Idle Timer τότε μεταβαίνει πάλι στην αρχική κατάσταση σταματώντας να αυξάνει το κόστος.

Είναι προφανές πως όσο περισσότερο παραμένει μια μηχανή σε κατάσταση Standby χωρίς να εκτελεί εργασία, το κόστος αυξάνεται, με μικρότερο βέβαια ρυθμό από ότι αυξάνεται όταν η μηχανή είναι απασχολημένη. Στην πράξη δε χρειάζεται να

υπολογίσουμε και να συγκρίνουμε πόσο χρόνο παραμένει ενεργή κάθε μηχανή, αφού δεδομένων των παραγγελιών όλες οι μηχανές θα είναι απασχολημένες συνολικά για τον ίδιο χρόνο.

Στην περίπτωση που υπάρχουν παράλληλες μηχανές (δηλαδή διαφορετικές μηχανές που μπορούν να εκτελέσουν την ίδια εργασία) τότε θα πρέπει να υπολογίζεται ο συνολικός χρόνος που είναι απασχολημένη κάθε μηχανή, αφού λόγω ηλικίας, τεχνολογίας ή άλλων συνθηκών είναι πιθανό να εκτελεί ταχύτερα ή πιο οικονομικά κάποια εργασία.

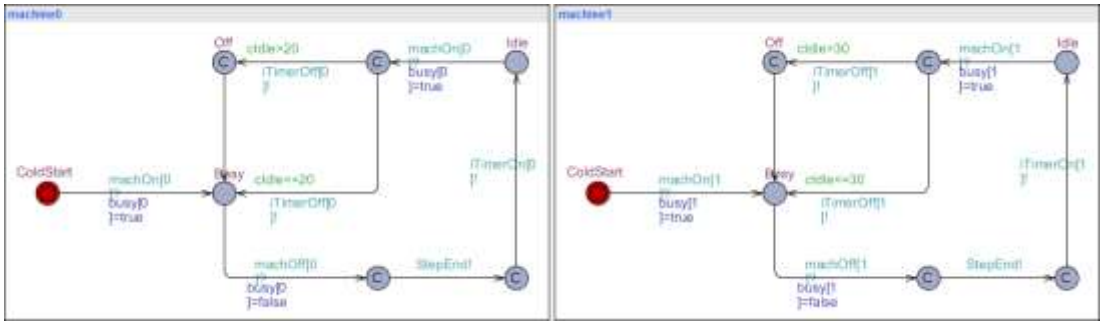
Στο συγκεκριμένο μοντέλο δεν εξετάζεται αυτό το ενδεχόμενο.

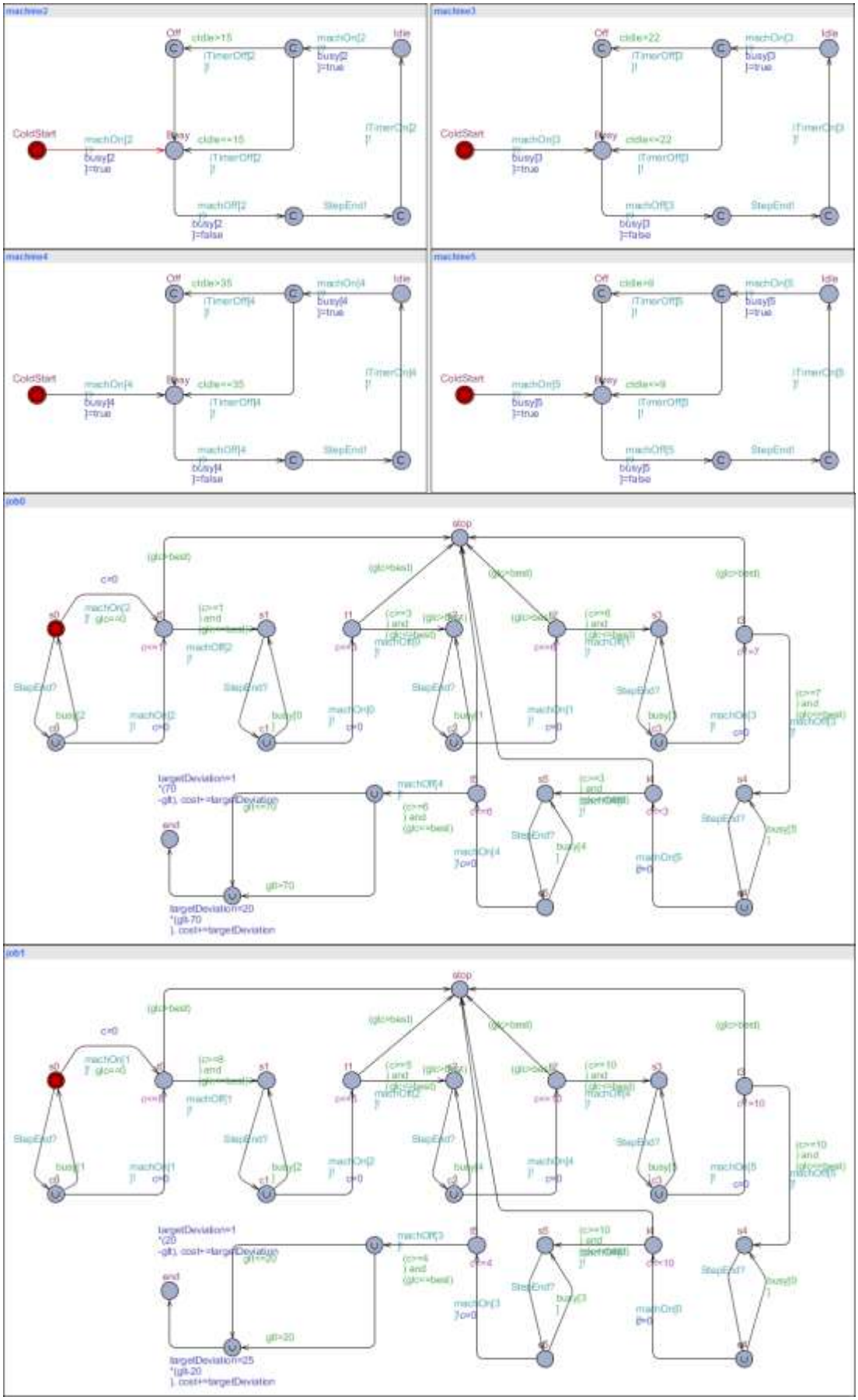
8.2.2. Προσομοίωση

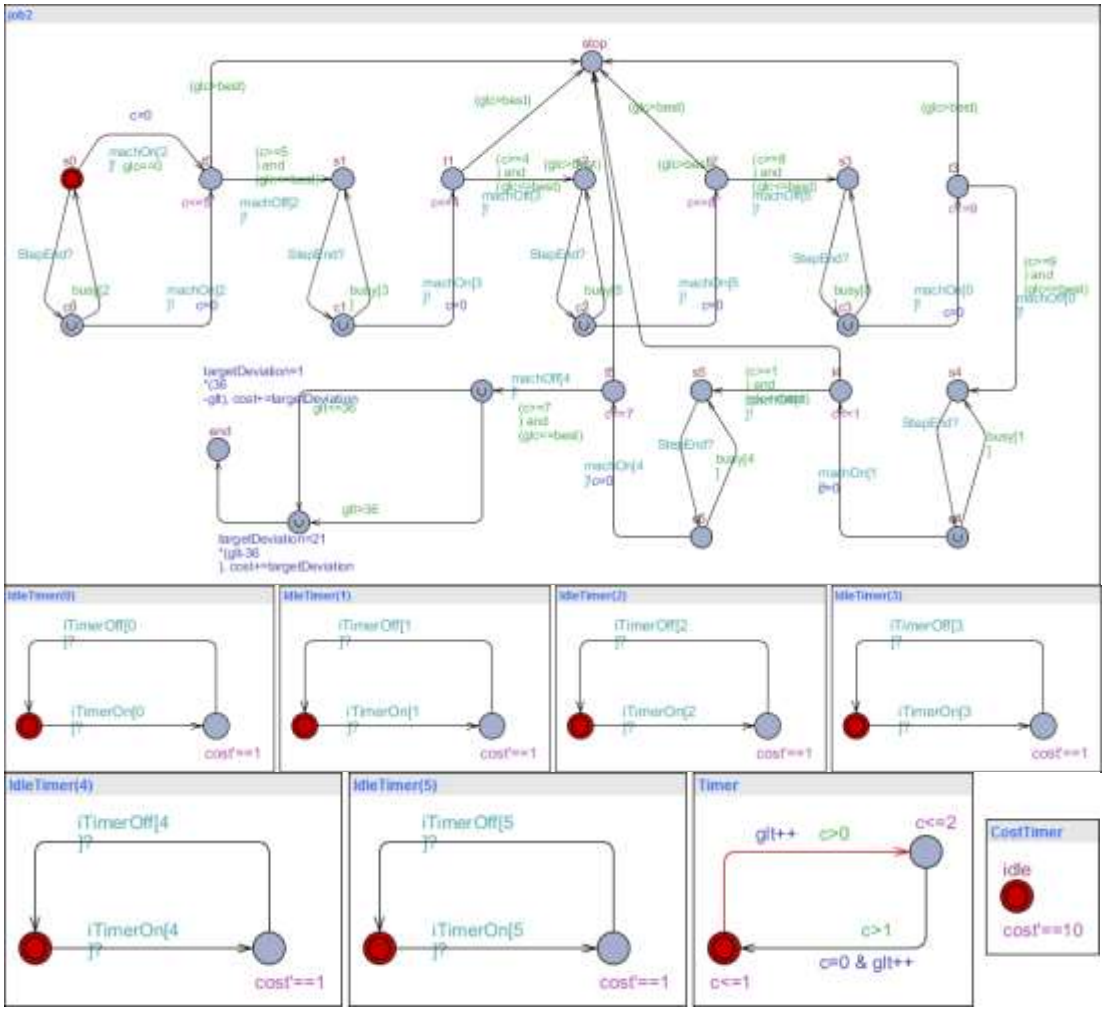
Χρησιμοποιώντας τον Simulator του Uppaal μπορούμε να ελέγξουμε τη ροή του προγράμματος με γραφική αναπαράσταση, γραμμική ροή, αλλά και ζωντανή ενημέρωση των μεταβλητών, των καταστάσεων και των μεταβάσεων. Υπάρχει επίσης δυνατότητα για εισαγωγή τυχαίων μεταβάσεων στο σύστημα για να εξεταστεί η συμπεριφορά του και να εντοπιστούν πιθανά κενά ή περιορισμοί.

Γραφική αναπαράσταση

Εδώ φαίνεται η αρχική κατάσταση κάθε στοιχείου του προγράμματος, βάσει του αντίστοιχου template. Όπως φαίνεται από τα σχήματα έχουν χρησιμοποιηθεί 6 μηχανές για 3 εργασίες. Επίσης στη γραφική αναπαράσταση φαίνονται και οι 6 μετρητές αναμονής, όπως επίσης και ο μετρητής χρόνου και συνολικού κόστους.



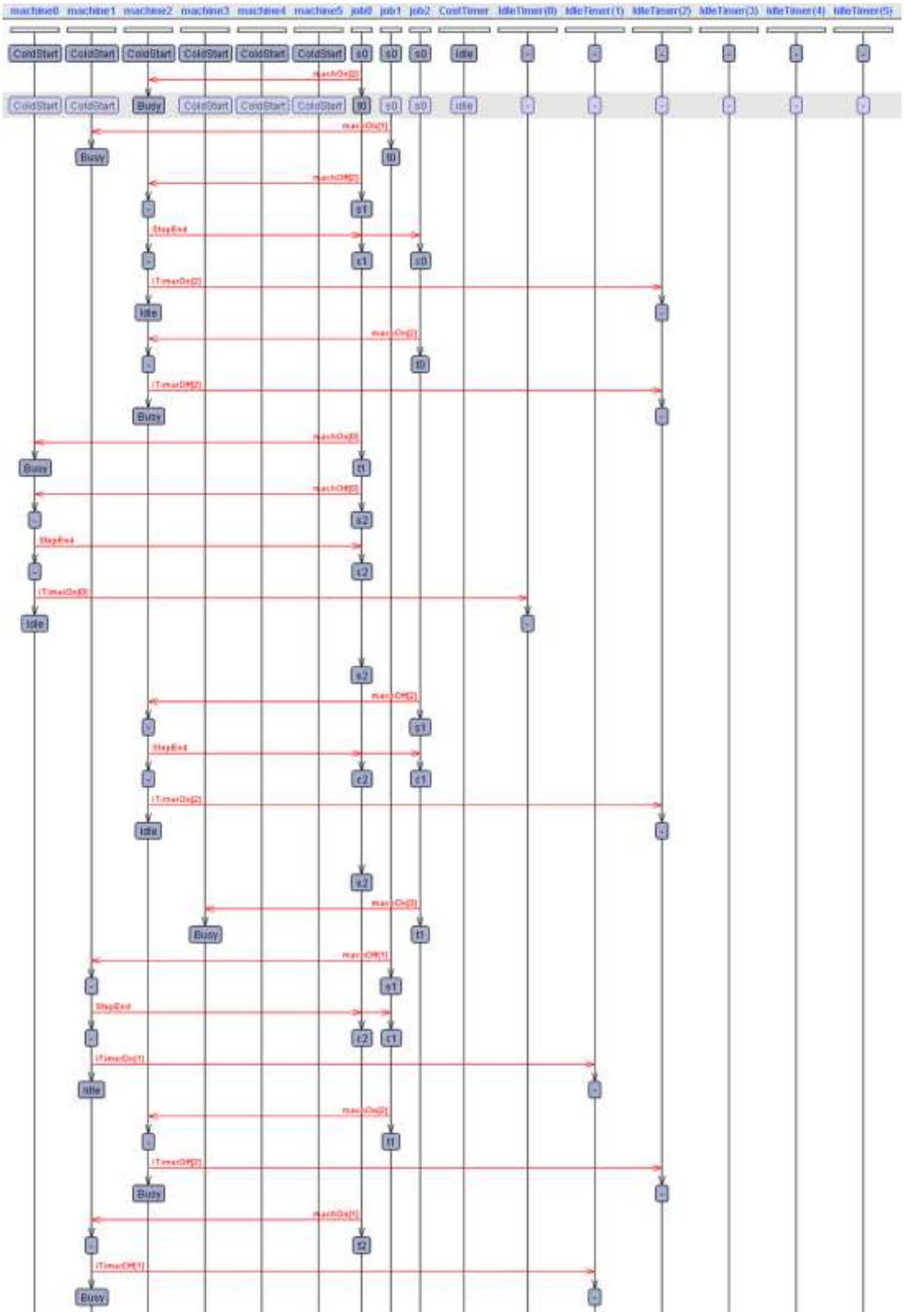

machine2
machine3
machine4
machine5
job0
job1
ColdStart
Off
Busy
Idle
StepEnd!
StepEnd?
stop
end

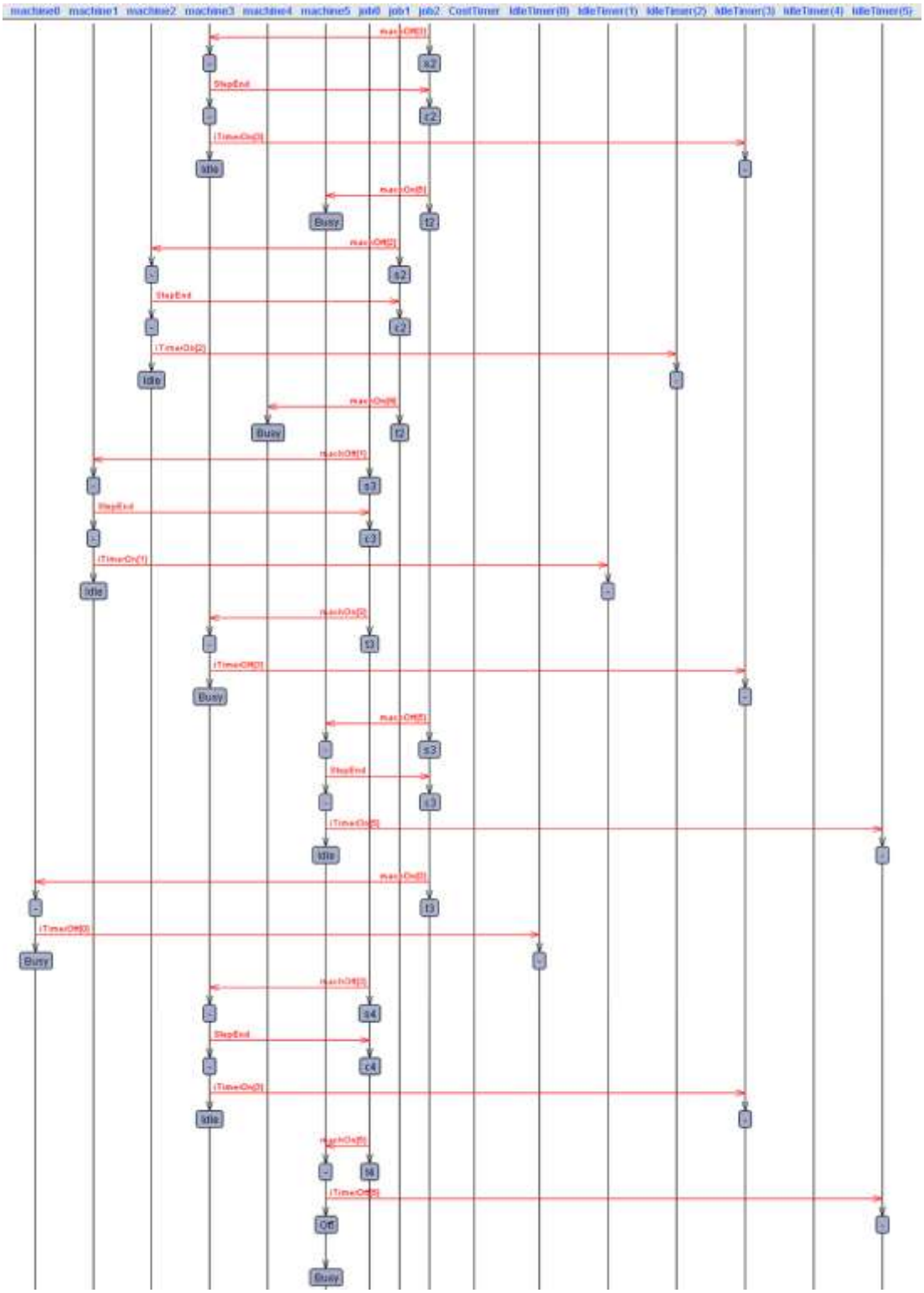


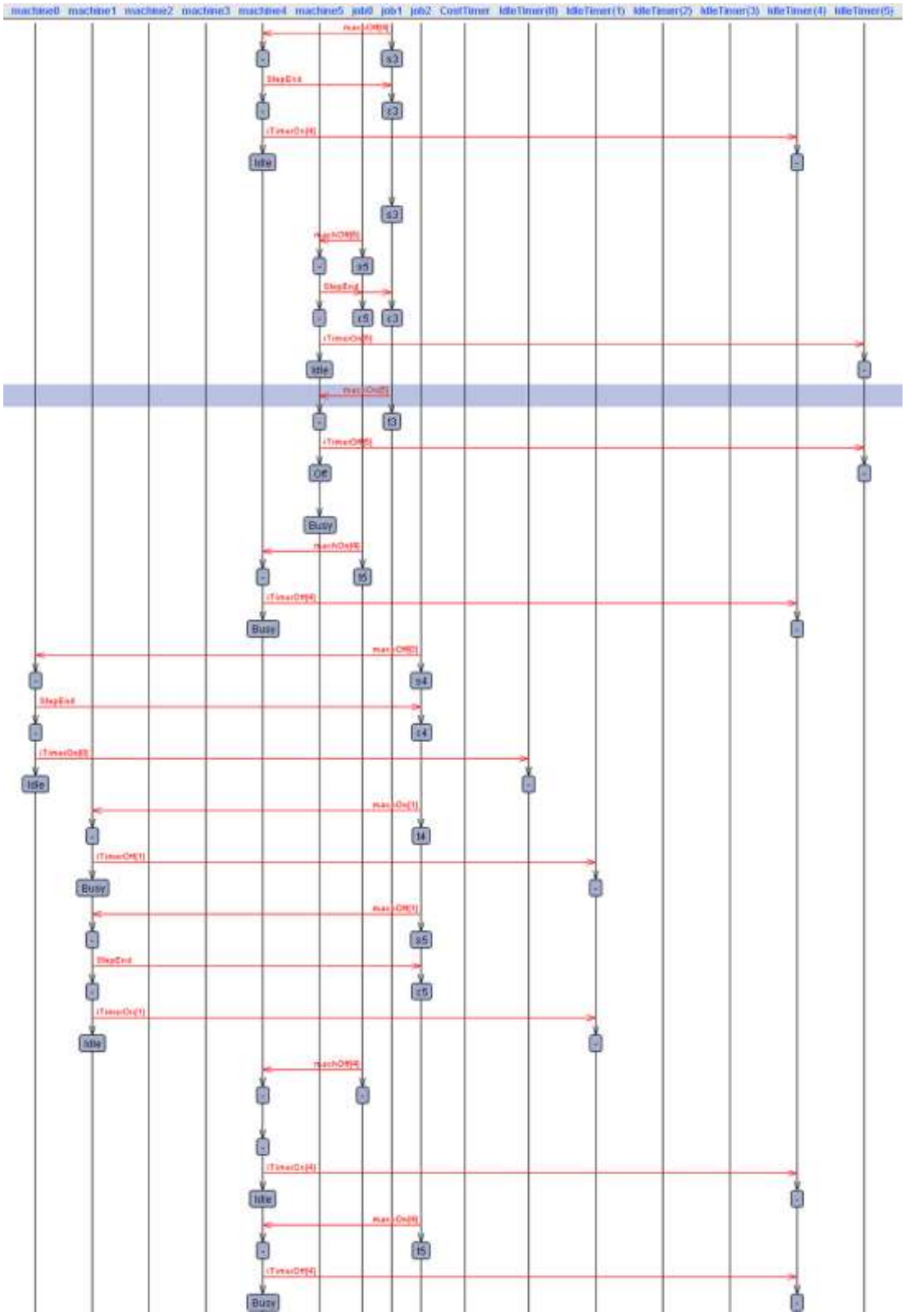
Σε κάθε μονάδα χρόνου γίνεται κόκκινη η κατάσταση που είναι ενεργή σε κάθε στοιχείο, ή η μετάβαση που εκτελείται.

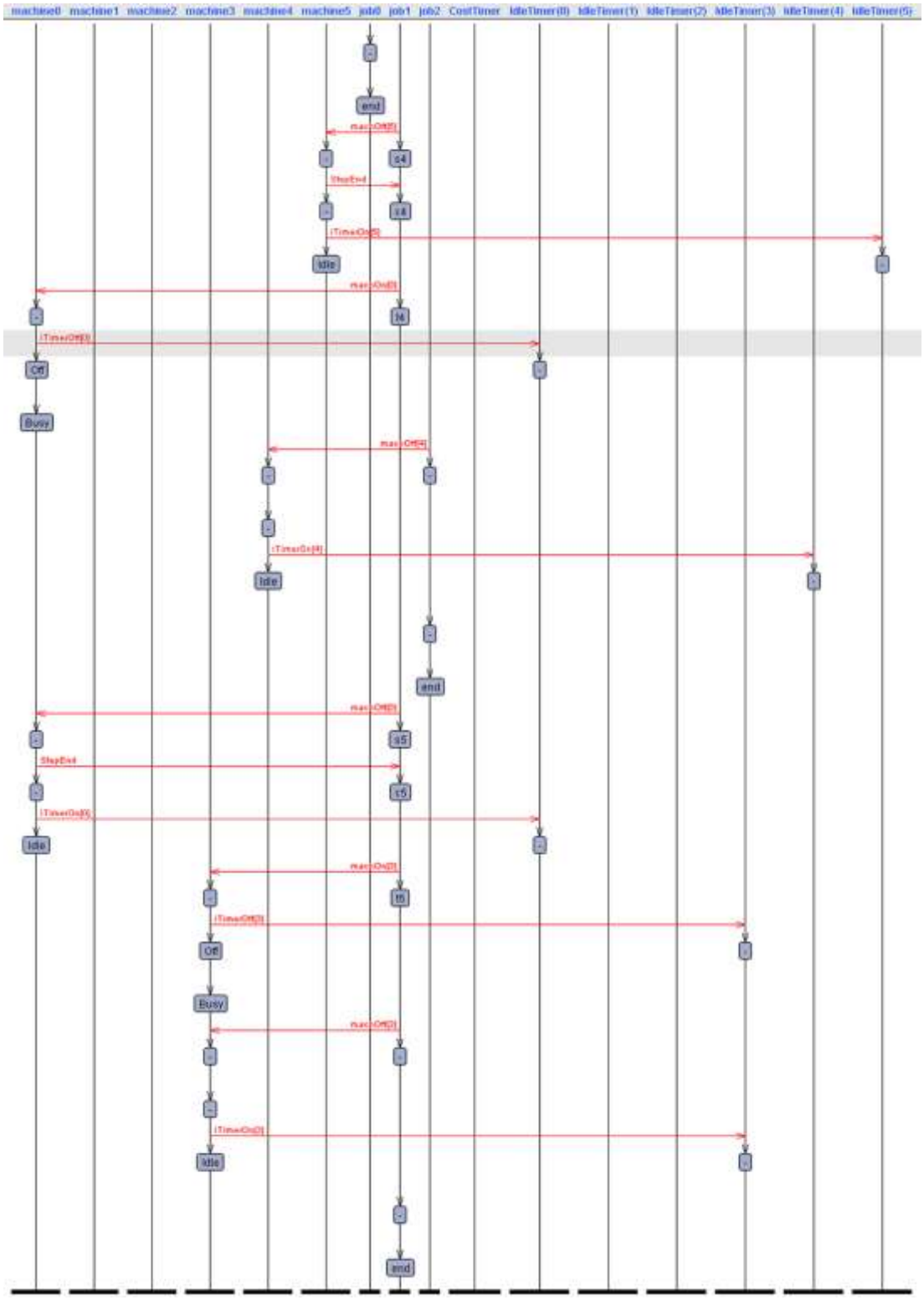
Γραμμική ροή

Στα παρακάτω σχήματα φαίνεται η ροή του προγράμματος για τυχαίες μεταβάσεις, σύμφωνα με την ενσωματωμένη λειτουργία random του Uppaal. Όπως φαίνεται τελικά όλες οι εργασίες φτάνουν στην κατάσταση end.


machine0
machine1
machine2
machine3
machine4
machine5
job0
job1
job2
ColdStart
IdleTimer(0)
IdleTimer(1)
IdleTimer(2)
IdleTimer(3)
IdleTimer(4)
IdleTimer(5)
Busy
Idle
s0
s1
s2
t0
t1
t2


machine0
machine1
machine2
machine3
machine4
machine5
job0
job1
job2
ConstTimer
IdleTimer(0)
IdleTimer(1)
IdleTimer(2)
IdleTimer(3)
IdleTimer(4)
IdleTimer(5)
StepEnd
Idle
Busy
Off


machine0
machine1
machine2
machine3
machine4
machine5
job0
job1
job2
ConstTimer
IdleTimer(0)
IdleTimer(1)
IdleTimer(2)
IdleTimer(3)
IdleTimer(4)
IdleTimer(5)
Idle
Off
Busy



8.2.3. Έλεγχος

Όπως φάνηκε στην προηγούμενη ενότητα, χρησιμοποιώντας την ενσωματωμένη λειτουργία random, το πρόγραμμα κατάφερε να βρει υλοποίηση ώστε όλες οι εργασίες να τελειώνουν. Αυτό ενώ φαίνεται προφανές, στην μοντελοποίηση

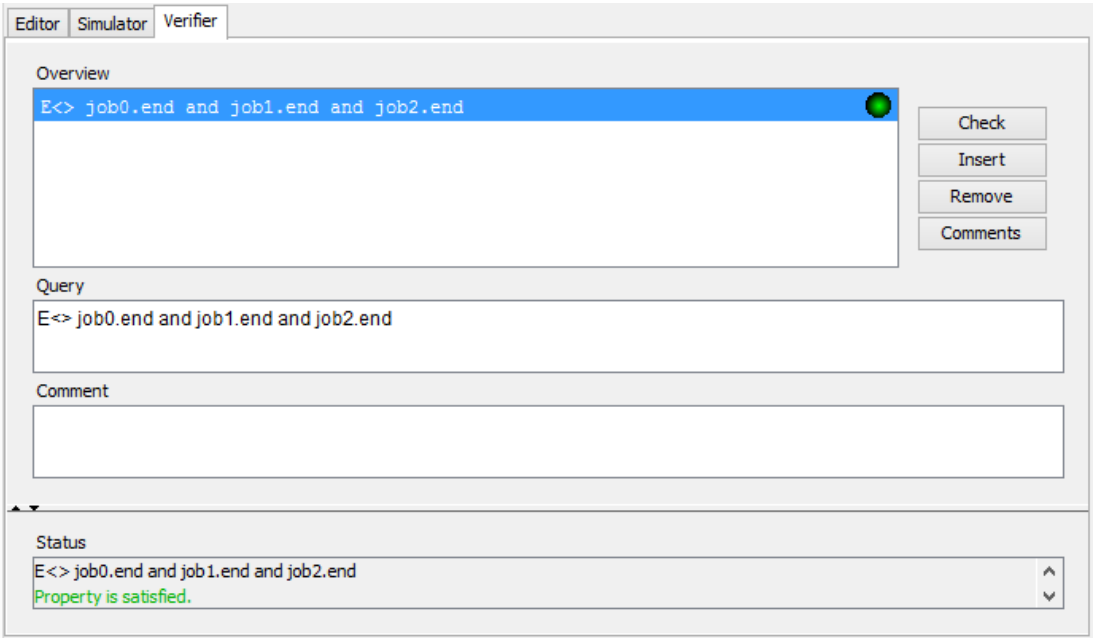
υπάρχουν συνήθως διάφορες δυσκολίες ή λάθη τα οποία μπορεί να προκαλέσουν στο πρόγραμμα να μείνει «κολλημένο» σε κάποια κατάσταση.

Το Uppaal διαθέτει τη μονάδα επαλήθευσης, όπου με κατάλληλα ερωτήματα γίνεται δυνατό να αντληθούν πληροφορίες σχετικά με την ύπαρξη ή όχι λύσης, τη μοναδικότητα και την καθολικότητα. Μπορούν να γίνουν συνδυασμοί των τελεστών A[], A<>, E<>, E[], -->.

Για παράδειγμά αν χρησιμοποιηθεί το ερώτημα

E<> job0.end and job1.end and job2.end,

Τότε η μονάδα επαλήθευσης θα ελέγξει αν το πρόβλημα έχει τουλάχιστον μια λύση για την οποία όλες οι εργασίες φτάνουν στο τέλος. Όπως φαίνεται στην εικόνα 3.11 επαληθεύεται η ύπαρξη λύσης ώστε όλες οι εργασίες να φτάνουν στο τέλος.



Σχήμα 62: Έλεγχος μοντέλου στο Uppaal

8.3. Πειραματική επαλήθευση

Έχοντας επιβεβαιώσει την ορθότητα του μοντέλου, επόμενο βήμα αποτελεί η εφαρμογή αλγορίθμου προσπελασιμότητας με αξιοποίηση της τεχνικής Branch&Bound για την εύρεση του βέλτιστου χρονοπρογράμματος με άξονα την ελαχιστοποίηση του κόστους.

Η προσθήκη της βιβλιοθήκης Cora στο Uppaal επιτρέπει την εφαρμογή αλγορίθμων βελτιστοποίησης πάνω στα μοντέλα αυτομάτων με άξονα το κόστος κατά απόλυτη αντιστοιχία με την εφαρμογή των αλγορίθμων που παρουσιάστηκαν στο κεφάλαιο 6 που είχαν ως άξονα το χρόνο.

Για την πειραματική επαλήθευση έγινε με τη βοήθεια της βιβλιοθήκης του Uppaal η εφαρμογή του ***αλγόριθμου αναζήτησης προτεραιότητας βέλτιστου* (*best–first search algorithm*)** ο οποίος έχει αναλυθεί στο Κεφάλαιο 4 της διατριβής.

Για την επαλήθευση του μοντέλου εξετάστηκαν δύο διακριτά προβλήματα. Το πρώτο αποτελεί ένα υποθετικό πρόβλημα που αναπαριστά σύστημα παραγωγής τύπου job shop, ενώ το δεύτερο πρόκειται για πρόβλημα πάνω σε πραγματικά δεδομένα παραγωγής μιας βιοτεχνίας παραγωγής στρωμάτων που εδρεύει στο Νομό Αττικής.

Για κάθε πρόβλημα γίνονται δύο εκτελέσεις του αλγορίθμου ως εξής: Στην πρώτη εκτέλεση επιλέγεται να γίνει βελτιστοποίηση με κριτήριο το χρόνο ολοκλήρωσης, ενώ στη δεύτερη με κριτήριο το κόστος (στο οποίο έχουν ενσωματωθεί τόσο μια αποτίμηση του κόστους ανά χρονική μονάδα όσο και το σύνολο των πηγών κόστους που έχουν μοντελοποιηθεί).

8.3.1. 1ο Πείραμα: Πρόβλημα χρονοπρογραμματισμού 4 εργασιών σε 6 μηχανές

Δήλωση δεδομένων μηχανών: Τα δεδομένα καταχωρούνται μέσα από μια σειρά δηλώσεων που ενσωματώνονται στο μοντέλο των αυτομάτων. Η πρώτη στήλη κάθε δήλωσης δείχνει τον αριθμό της μηχανής και η δεύτερη μετά από πόσο χρόνο συμφέρει να κλείσει η μηχανή αν παραμένει αχρησιμοποίητη.

- machine0 = Machine(0,20);
- machine1 = Machine(1,30);
- machine2 = Machine(2,15);
- machine3 = Machine(3,22);
- machine4 = Machine(4,35);
- machine5 = Machine(5,9);

Αντίστοιχα οι 4 εργασίες καταχωρούνται με τις ακόλουθες δηλώσεις. Τα 5 πρώτα ζεύγη δεδομένων για κάθε εργασία δείχνουν τον αριθμό της μηχανής που πρέπει να απασχολήσουν και τον συνολικό χρόνο που απαιτείται σε κάθε μηχανή. Στο

πρόβλημα που εξετάζεται η σειρά κατά την οποία αναφέρονται τα ζεύγη τηρείται υποχρεωτικά κατά την εκτέλεση του προγράμματος. Τα τελευταία τρία δεδομένα κάθε δήλωσης αφορούν στον προγραμματισμένο χρόνο παράδοσης της κάθε εργασίας και στα κόστη αποθήκευσης και καθυστέρησης παράδοσης αντίστοιχα.

```
job0=Job(1,4,   3,5,   0,8,   2,6,   4,4,   5,8,   70,1,20);
job1=Job(3,6,   1,2,   5,9,   0,11,  2,7,   4,6,   20,1,25);
job2=Job(1,4,   4,7,   3,4,   2,7,   0,2,   5,5,   36,1,21);
job3=Job(5,8,   2,5,   3,6,   1,5,   4,9,   0,4,   11,2,15);
```

8.3.1.1. Βελτιστοποίηση με κριτήριο το χρόνο ολοκλήρωσης

Εκτελώντας την υλοποίηση για βελτιστοποίηση μόνο βάσει του κόστους που αντιστοιχεί στο χρόνο ολοκλήρωσης, το συνολικό κόστος ανέρχεται στις **1009** μονάδες κόστους, ενώ με κατάλληλη επεξεργασία των αποτελεσμάτων που εξάγει ο αλγόριθμος (διαθέσιμα στο παράρτημα) παρουσιάζεται η σειρά μεταβάσεων που προκύπτει:

job1 --> machine3
job0 --> machine1
job3 --> machine5
job2 --> machine1
job0 --> machine3
job2 --> machine4
job1 --> machine1
job0 --> machine0
job3 --> machine2
job1 --> machine5
job2 --> machine3
job0 --> machine2
job3 --> machine3
job1 --> machine0
job0 --> machine4
job3 --> machine1
job2 --> machine2
job3 --> machine4
job0 --> machine5
job2 --> machine0
job1 --> machine2
job1 --> machine4
job2 --> machine5
job3 --> machine0

8.3.1.2. Βελτιστοποίηση με κριτήριο το κόστος

Στη συγκεκριμένη περίπτωση επιλέχθηκε να ληφθούν υπόψη όλα τα κόστη για την εξαγωγή της βέλτιστης λύσης. Το πλήρες log των μεταβάσεων δίνεται στο παράρτημα. Με την εκτέλεση του προγράμματος, το συνολικό κόστος ανέρχεται στις **961** μονάδες.

Λίστα μεταβάσεων:

job1 --> machine3
job0 --> machine1
job3 --> machine5
job2 --> machine1
job0 --> machine3
job2 --> machine4
job1 --> machine1
job0 --> machine0
job3 --> machine2
job1 --> machine5
job2 --> machine3
job0 --> machine2
job3 --> machine3
job1 --> machine0
job0 --> machine4
job2 --> machine2
job3 --> machine1
job3 --> machine4
job0 --> machine5
job2 --> machine0
job1 --> machine2
job1 --> machine4
job2 --> machine5
job3 --> machine0

Στη δεύτερη περίπτωση, όπου λαμβάνουμε υπόψη τη βελτιστοποίηση κόστους μέσω περισσότερων παραγόντων, παρατηρούμε ότι υπάρχει μια εξοικονόμηση της τάξης του 5%. Παρατηρούμε επίσης ότι υπάρχει αλλαγή στη σειρά εκτέλεσης των εργασιών, το οποίο είναι επίσης αναμενόμενο και καταδεικνύει τη σημασία που έχει το κριτήριο βελτιστοποίησης για την εξεύρεση του κατάλληλου χρονοπρογράμματος σε κάθε περίπτωση.

8.3.2. 2ο Πείραμα: Προγραμματισμός 4 εργασιών σε 5 μηχανές με ευελιξία

Στο πείραμα αυτό, εξετάζεται η παραγωγική διαδικασία 4 προϊόντων μίας μονάδας παραγωγής στρωμάτων που εδρεύει στα Σπάτα Αττικής. Τα τέσσερα προϊόντα αντιστοιχούν σε τέσσερις διαφορετικές ποιότητες στρωμάτων και εξετάζεται η παραγωγή τους υπό τη μορφή τεσσάρων εντολών παραγωγής (εργασιών) που δηλώνονται ως job0, job1, job2 και job3 αντίστοιχα.

Για την κατασκευή, επεξεργασία και συσκευασία των στρωμάτων χρησιμοποιούνται, ανάλογα με τον τύπο από 4 μέχρι 5 κέντρα κατεργασίας (μοντελοποιούνται ως μηχανές).

Τα πέντε κέντρα που διαθέτει η μονάδα είναι τα εξής:

- Καπιτονιέρα (μηχανή που φτιάχνει τα «καπάκια» - το πάνω και κάτω μέρος του υφάσματος στο στρώμα)
- Δημιουργία τελάρου (ενώνει τις σούστες για τη δημιουργία του τελάρου)
- Δημιουργία φάσας (πλαϊνό ύφασμα του στρώματος)
- Ρέλιασμα (ράψιμο του στρώματος – ένωση του καπακιού με τη φάσα)
- Συσκευαστική (συσκευασία των προϊόντων)

Οι εργασίες job1, job2 δεν περνάνε από τη 2η μηχανή (machine1), επειδή τα τελάρα των στρωμάτων εισάγονται έτοιμα από εξωτερικό προμηθευτή.

Οι τιμές που χρησιμοποιήθηκαν για τη συγκεκριμένη περίπτωση αναφέρονται σε παραγωγή στρωμάτων ποσότητας 100 έως 400 τεμαχίων.

```
job0=Job(0,1,  1,8,    2,3,   3,16,  4,2,   5,0,   32,0,20);
job1=Job(0,1,  1,15,   2,5,   3,23,  4,2,   5,0,   50,0,25);
job2=Job(0,2,  1,11,   2,4,   3,18,  4,3,   5,0,   41,0,25);
job3=Job(0,2,  1,23,   2,5,   3,29,  4,2,   5,0,   65,0,30);
```

Σημείωση: Στη στήλη του κόστους αποθήκευσης έχουμε συμπληρώσει 0, αφού το εργοστάσιο έχει ιδιόκτητη αποθήκη και το κόστος θεωρείται αμελητέο.

8.3.2.1. Βελτιστοποίηση με κριτήριο το χρόνο

Το συνολικό κόστος περάτωσης των εργασιών, όταν βελτιστοποιείται βάσει χρόνου, ανέρχεται στις **2050** μονάδες κόστους.

Στον πίνακα που ακολουθεί φαίνεται η σειρά μεταβάσεων που εξήγαγε ο αλγόριθμος βελτιστοποίησης:

job0 --> machine0
job1 --> machine0
job0 --> machine1
job3 --> machine0
job2 --> machine0
job2 --> machine1
job0 --> machine2
job0 --> machine3
job2 --> machine2
job1 --> machine1
job2 --> machine3
job0 --> machine4
job3 --> machine1
job1 --> machine2
job1 --> machine3
job2 --> machine4
job3 --> machine2
job3 --> machine3
job1 --> machine4
job3 --> machine4
job0 --> machine5
job2 --> machine5
job1 --> machine5
job3 --> machine5

8.3.2.2. Βελτιστοποίηση με κριτήριο το κόστος

Σε αντιστοιχία με το 1ο πείραμα, ο αλγόριθμος εφαρμόστηκε εκ νέου λαμβάνοντας όμως υπόψη όλα τα διαθέσιμα στοιχεία κόστους.

Η εφαρμογή του αλγορίθμου έδωσε ως βέλτιστη λύση τις 1775 μονάδες κόστους, οδηγώντας στην ακόλουθη σειρά μεταβάσεων:

job0 --> machine0
job0 --> machine1
job1 --> machine0
job2 --> machine0
job3 --> machine0
job2 --> machine1

job0 --> machine2
job0 --> machine3
job1 --> machine1
job2 --> machine2
job2 --> machine3
job0 --> machine4
job1 --> machine2
job3 --> machine1
job2 --> machine4
job1 --> machine3
job3 --> machine2
job3 --> machine3
job1 --> machine4
job3 --> machine4
job0 --> machine5
job2 --> machine5
job1 --> machine5
job3 --> machine5

Όπως και στο προηγούμενο πείραμα παρατηρείται μείωση του κόστους, έπειτα από την εφαρμογή των βελτιστοποιήσεων της τάξης του 13%. Επίσης παρατηρούνται ιδιαίτερα σημαντικές διαφορές στη σειρά εκτέλεσης των μεταβάσεων, δηλαδή στο προτεινόμενο πρόγραμμα παραγωγής.

8.3.3. Σχολιασμός

Τα πειράματα τα οποία διεξήχθησαν επιβεβαίωσαν την αποτελεσματικότητα της μεθόδου και για την περίπτωση της σύνθετης μοντελοποίησης κατά την οποία μια σειρά διαφορετικών παραγόντων κόστους λαμβάνονται υπόψη. Η έλλειψη προβλημάτων αναφοράς τέτοιας πολυπλοκότητας δεν επιτρέπει φυσικά οποιαδήποτε σύγκριση με άλλες μεθόδους, εν τούτοις θεωρείται εξ ορισμού δεδομένο από την ίδια τη φύση του αλγορίθμου που αξιοποιήθηκε μέσω της μηχανής βελτιστοποίησης του Uppaal ότι οι τιμές οι οποίες υπολογίστηκαν είναι οι βέλτιστες, καθώς ο αλγόριθμος τερμάτισε χωρίς να προσεγγισθεί το όριο χρόνου ή αριθμου μεταβάσεων το οποίο είχε τεθεί. Η σημασία της επιβεβαίωσης της λειτουργίας της μεθόδου με σύνθετα στοιχεία κόστους αναδεικνύεται ακόμα περισσότερο στο επόμενο κεφάλαιο κατά το οποίο εφαρμόζεται η προσέγγιση σε δίκτυο παραγωγικών

μονάδων όπου το κόστος αποτελεί το βασικό παράγοντα για την δρομολόγηση των εργασιών σε κάθε εναλλακτικό προμηθευτή.

Κεφάλαιο 9: Εφαρμογή στον χρονικό προγραμματισμό Δυναμικών Δικτύων Παραγωγής

9.1. Το πρόβλημα του προγραμματισμού παραγωγικών δικτύων

Δεδομένων των ιδιαίτερα ανταγωνιστικών συνθηκών στο χώρο της βιομηχανίας σε παγκόσμιο επίπεδο, οι επιχειρήσεις του βιομηχανικού κλάδου έχουν στραφεί στην υιοθέτηση στρατηγικών συνεργασίας στην κατεύθυνση της μείωσης τόσο του κόστους όσο και του απαιτούμενου χρόνου παραγωγής των προϊόντων. Τα δυναμικά δίκτυα παραγωγής αποτελούν συνεργατικούς σχηματισμούς, οι οποίοι, στη βάση συμφωνηθέντων δεδομένων, αξιοποιούν τις ικανότητες των μελών τους μέσα από μια συνεργασία δυναμικής φύσης – η οποία εξαρτάται από τις ανάγκες της παραγωγής. Στόχος των δυναμικών δικτύων παραγωγής είναι η επίτευξη βελτιστοποιημένων, ως προς το κόστος, το χρόνο και την ποιότητα, διαδικασιών παραγωγής για την κάλυψη των συνεχώς πιο σύνθετων αναγκών της ανταγωνιστικής αγοράς. Αν και δεν αποτελεί θέσφατο στη θεωρία, πρακτικά τα δυναμικά δίκτυα παραγωγής δημιουργούνται με άξονα μια μεγάλη βιομηχανική επιχείρηση η οποία είναι και ο τελικός παραγωγός των προϊόντων και έχει την επαφή με την αγορά και τον τελικό πελάτη [111]. Προκειμένου να επιτευχθούν επίπεδα κόστους αλλά και

χρονικής ανταπόκρισης που δύσκολα θα μπορούσαν να καλυφθούν από την παραδοσιακή δομή προμηθευτή – πελάτη, οι βιομηχανίες επιδιώκουν να εντάξουν σε ένα δίκτυο τους προμηθευτές με τους οποίους κατά κύριο λόγο συνεργάζονται – κατά βάση Μικρές και Μεσαίες Επιχειρήσεις (ΜΜΕ), από τη μία δεσμεύοντάς τους για την αμεσότερη ανταπόκριση στα αιτήματα παραγωγής και προμήθειας και από την άλλη διασφαλίζοντάς τους μια σταθερή συνεργασία και άρα ένα «εγγυημένο» όγκο πωλήσεων. Στο πλαίσιο αυτό δημιουργούνται τα λεγόμενα δυναμικά δίκτυα παραγωγής, αποτελούμενα από τους τελικούς κατασκευαστές των προϊόντων και από ένα πλήθος προμηθευτών τους οι οποίοι συμφωνούν να συνεργαστούν και να ενταχθούν σε ένα κοινό δικτύο χωρίς αυτό να σημαίνει ότι παύουν να λειτουργούν αυτόνομα ή ότι παύει ο μεταξύ τους ανταγωνισμός. Όλοι οι συμμετέχοντες προμηθευτές συμφωνούν στη διάθεση πληροφοριών σχετικά με την παραγωγή τους και τη δυναμικότητα των παραγωγικών τους τμημάτων προς τον τελικό κατασκευαστή, πάντα υπό πλήρη έλεγχο στα επίπεδα πρόσβασης στα διατιθέμενα στοιχεία [111].

Το χαρακτηριστικό των Δυναμικών Δικτύων Παραγωγής (ΔΔΠ) είναι ότι οι συμμετέχουσες επιχειρήσεις όχι μόνο αναλαμβάνουν μέσω του δικτύου να εκτελέσουν βιομηχανικές διεργασίες αντί του τελικού κατασκευαστή αλλά κατά περίπτωση συνεργάζονται μεταξύ τους (παρά το γεγονός ότι μπορεί να είναι ανταγωνίστριες στον κλάδο τους) προκειμένου να πετύχουν τη μείωση του κόστους και του χρόνου διάθεσης του τελικού προϊόντος στην αγορά ή αλλιώς τη βελτιστοποίηση της όλης παραγωγικής του διαδικασίας [99].

Προκειμένου να επιτευχθεί η επιδιωκόμενη βελτιστοποίηση σε ένα ΔΔΠ, ο μεσοπρόθεσμος προγραμματισμός παραγωγής εμπλέκεται με την επιλογή των απαιτούμενων βιομηχανικών διεργασιών που θα εκτελέσει κάθε μέλος του δικτύου. Ουσιαστικά δηλαδή ο μεσοπρόθεσμος προγραμματισμός μετατρέπεται σε ένα πολυπλοκότερο πρόβλημα ανάθεσης εργασιών σε συγκεκριμένα παραγωγικά τμήματα των συμμετεχουσών επιχειρήσεων, πάντα με βάση και τη διαθεσιμότητα αλλά και την τεχνική ικανότητά τους να ανταπεξέλθουν. Για το πρόβλημα αυτό έχει διατυπωθεί μια σειρά εναλλακτικών προσεγγίσεων αντιμετώπισης που ανάγονται στην επιλογή προμηθευτών βάσει των χαρακτηριστικών τους ή/και βάση της παραγωγικής τους ικανότητας. Όμως η πλειοψηφία των προσεγγίσεων που εμφανίζονται στη βιβλιογραφία περιορίζονται σε μακροσκοπικά στοιχεία και όχι σε δεδομένα πραγματικού χρόνου για την παραγωγική ικανότητα, ενώ επίσης μερικώς

μόνο λαμβάνουν υπόψη το γεγονός ότι οι παραγωγοί – προμηθευτές, εκτός του ότι ανήκουν στο ΔΔΠ, έχουν και ανεξάρτητη παρουσία, αναλαμβάνοντας έτσι να εκτελέσουν και βιομηχανικές διεργασίες για παραγγελίες εκτός του δικτύου [110].

Στο κεφάλαιο αυτό προτείνεται μια ολοκληρωμένη προσέγγιση η οποία αξιοποιεί τη μεθοδολογία και τα μοντέλα των χρονισμένων αυτομάτων που περιγράφηκαν νωρίτερα στην παρούσα διατριβή και τα προσαρμόζει στην κατεύθυνση του προγραμματισμού δυναμικών δικτύων παραγωγής, λαμβάνοντας υπόψη τόσο τους χρόνους και τα κόστη παραγωγής όσο και τα κόστη της εφοδιαστικής (αποθήκευσης και διανομής) τα οποία συχνά αγνοούνται από το μεγαλύτερο μέρος των υφιστάμενων προσεγγίσεων.

9.2. Μοντελοποίηση δεδομένων

Η μείωση του συνολικού κόστους αποτελεί, όπως είναι προφανές, τον άξονα γύρω από τον οποίο κινείται κάθε προσπάθεια προγραμματισμού παραγωγής. Στα Δυναμικά Δίκτυα Παραγωγής η βελτιστοποίηση του κόστους για την κάλυψη μιας παραγωγικής ανάγκης ανάγεται στη βέλτιστη επιλογή των μελών του δικτύου που θα εκτελέσουν τις παραγωγικές διεργασίες. Δεδομένου ότι αυτό που ενδιαφέρει είναι το συνολικό κόστος για το τελικό προϊόν και την παραγωγή του, για την επιλογή των συμμετεχόντων μελών σε κάθε περίπτωση θα πρέπει να λαμβάνεται υπόψη τόσο το καθαρό κόστος της παραγωγής (εκτέλεσης των βιομηχανικών διεργασιών) όσο και κόστη που σχετίζονται με την αποθήκευση και μεταφορά των προϊόντων. Ένα ερώτημα που προκύπτει είναι το πώς μπορεί να μοντελοποιηθεί ο χρόνος, δεδομένου ότι θεωρείται ως εκκίνηση της παραγωγικής διαδικασίας η παραγγελία που δίνει ο πελάτης στον τελικό κατασκευαστή, η οποία έχει συγκεκριμένες προθεσμίες παράδοσης. Προκειμένου να συμπεριληφθεί στη βελτιστοποίηση ο παράγων του χρόνου και καθώς το να θεωρηθεί ο χρόνος ολοκλήρωσης απλά ως ένας περιορισμός του προβλήματος δεν ανταποκρίνεται στην πραγματική λειτουργία των βιομηχανικών επιχειρήσεων, κάθε χρονική απόκλιση επιλέγεται να μεταφραστεί σε ένα κόστος αποθήκευσης ή καθυστερημένης παράδοσης (πχ λόγω επιβολής ρητρών). Επιπρόσθετα και καθώς στα ΔΔΠ η γεωγραφική διασπορά είναι συνηθισμένη περίπτωση, προστίθεται στο πρόβλημα και ο έλεγχος του κόστους της μεταφοράς από μία επιχείρηση – μέλος του δικτύου – σε μία άλλη όπως επίσης και ο απαιτούμενος χρόνος που δαπανάται.

Ας θεωρήσουμε ότι για μια παραγγελία που δέχεται από ένα πελάτη, ο τελικός κατασκευαστής εντοπίζει, εντός του ΔΔΠ που έχει δημιουργήσει με τους προμηθευτές του, ένα σετ από μέλη του δικτύου (K) ικανά να συμμετέχουν στην παραγωγή του προϊόντος μέσω από την εκτέλεση ορισμένων εκ των απαιτούμενων διεργασιών παραγωγής του (J). Έστω K ο αριθμός των προμηθευτών (δείκτης k) and J ο αριθμός των διεργασιών που πρέπει να εκτελεστούν (δείκτης j), ενώ $\{\lambda_i\}$ το σύνολο των αναθέσεων των διεργασιών σε παραγωγικά τμήματα των προμηθευτών ($i = 1, 2, 3, \ldots, I$).

Το συνολικό κόστος προς βελτιστοποίηση έχει τρεις συνιστώσες:

$$C(\lambda_i) = C_{\mathrm{p}}(\lambda_i) + C_{\mathrm{s}}(\lambda_i) + C_{\mathrm{t}}(\lambda_i)$$

όπου:

$C_{\mathrm{p}}(\lambda_i)$ είναι τα κόστη εκτέλεσης των βιομηχανικών διεργασιών, $C_{\mathrm{s}}(\lambda_i)$ τα κόστη μεταφοράς και $C_{\mathrm{t}}(\lambda_i)$ τα κόστη που πηγάζουν από τη χρονική διάσταση του προγραμματισμού.

Στη συνέχεια οι μεταβλητές του προβλήματος συμβολίζονται ως εξής:

(j,k): συνδυασμός διεργασίας – προμηθευτή όπου η διεργασία j εκτελείται από τον προμηθευτή k

$T_p(\lambda_i)$: συνολικός χρόνος παραγωγικών διεργασιών για το σύνολο των αναθέσεων λ_i

$T_s(\lambda_i)$: συνολικός χρόνος μεταφοράς προϊόντων

c_{jk}: κόστος διεργασίας j που εκτελείται από τον προμηθευτή k

t_{jk}: διάρκεια διεργασίας j που εκτελείται από τον προμηθευτή k

T_d: απαιτούμενος από τον πελάτη χρόνος παράδοσης της παραγγελίας

$T(\lambda_i)$: πραγματικός χρόνος παράδοσης της παραγγελίας στον πελάτη

c_{km}: κόστος μεταφοράς μιας παρτίδας προϊόντων (ανεξαρτήτου μεγέθους θεωρούμενο) από τον προμηθευτή k στον προμηθευτή m

c_{α}: κόστος αποθήκευσης ανά μονάδα χρόνου του τελικού προϊόντος μέχρι την παράδοση στον πελάτη

c_{β}: κόστος καθυστέρησης ανά μονάδα χρόνου της παράδοσης του προϊόντος στον πελάτη

Βάσει των παραπάνω, ο παράγων κόστους που σχετίζεται με το χρόνο ισούται, όπως είναι σαφές με:

$$C_t(\lambda_i) = c_\alpha E(\lambda_i) + c_\beta D(\lambda_i)$$

όπου:

$E(\lambda_i) = \max\{0, T_d - T(\lambda_i)\}$, *εκφράζει τη νωρίτερη ολοκλήρωση*
$D(\lambda_i) = \max\{0, T(\lambda_i) - T_d\}$, *εκφράζει την καθυστέρηση παράδοσης*

Θεωρούμε επίσης τη βοηθητική μεταβλητή z_{jk} η οποία παίρνει τιμή 1 εφόσον η διεργασία j εκτελείται από τον προμηθευτή k και 0 στην αντίθετη περίπτωση. Έτσι το κόστος εκτέλεσης των βιομηχανικών διεργασιών ισούται με:

$$C_p(\lambda_i) = \sum_{(j,k)\in\lambda_i} z_{jk} c_{jk}$$

Ενώ αντίστοιχα ο συνολικός χρόνος διεργασίας με:

$$T_p(\lambda_i) = \sum_{(j,k)\in\lambda_i} z_{jk} t_{jk}$$

Ο τρίτος παράγοντας κόστους που λαμβάνεται υπόψη αναφέρεται στη μεταφορά προϊόντων μεταξύ των επιχειρήσεων – μελών του δικτύου. Για την παράσταση του κόστους αυτού θεωρούμε τη βοηθητική μεταβλητή δ_{km} η οποία παίρνει τιμή 1 εφόσον k<>m, δηλαδή στις περιπτώσεις που οι διεργασίες j και l εκτελούνται από διαφορετικό προμηθευτή εφόσον l=j+1 ή j=λ+1 (δηλ μόνο για διαδοχικές εργασίες) και 0 σε κάθε άλλη περίπτωση.

Έτσι το κόστος μεταφορών μπορεί να εκφραστεί ως εξής:

$$C_s(\lambda_i) = \sum_{(j,k)\in\lambda_i} \sum_{(l,m)\in\lambda_i} z_{jk} z_{lm,} \delta_{km} c_{km}$$

Το πρόβλημα σε επίπεδο μοντέλου ανάγεται τελικά στην ελαχιστοποίηση του

$$C(\lambda_i) = C_p(\lambda_i) + C_s(\lambda_i) + C_t(\lambda_i)$$

εν τούτοις η αντιμετώπισή του είναι ιδιαίτερα πολύπλοκη για να μπορεί να επιτευχθεί με μαθηματική καθαρά προσέγγιση σε αποδεκτό χρόνο, δεδομένου του αριθμού των εναλλακτικών συνδυασμών $\{\lambda_i\}$ αλλά και του γεγονότος ότι ο χρόνος που μπορεί κάθε υποψήφιος προμηθευτής να ολοκληρώσει μια παραγγελία εξαρτάται μεταξύ άλλων και από τις άλλες παραγγελίες που έχει ο προμηθευτής να εκτελέσει και οι οποίες ενδέχεται να μην προέρχονται από ανάγκες του ΔΔΠ αλλά από ανεξάρτητη

πελατεία του.

9.3. Μεθοδολογία επίλυσης του προβλήματος με χρονισμένα αυτόματα

Στην παράγραφο αυτή προτείνεται μια μεθοδολογία επίλυσης του προβλήματος χρονικού προγραμματισμού δυναμικών δικτύων παραγωγής, όπως διατυπώθηκε στην προηγούμενη παράγραφο, αξιοποιώντας τα μοντέλα χρονισμένων αυτομάτων με κόστη τα οποία αναπτύχθηκαν στα προηγούμενα στάδια της διατριβής. Η μεθοδολογία είναι προσαρμοσμένη στην επίλυση πραγματικών προβλημάτων και δεν περιορίζεται στο γενικό μαθηματικό πρόβλημα. Στο πλαίσιο αυτό πρέπει να αναφερθούν τα εξής:

Η κατασκευάστρια του τελικού προϊόντος η οποία και έχει τη διαχείριση του ΔΔΠ απαιτείται να έχει μια σειρά στοιχείων προκειμένου να μπορεί να αναπτύξει τα εναλλακτικά χρονοπρογράμματα τα οποία ανάγονται σε αναθέσεις διεργασιών σε προμηθευτές – μέλη του δικτύου. Συγκεκριμένα θεωρείται ότι είναι σε θέση να γνωρίζει τα παραγωγικά τμήματα των μελών του δικτύου και τις δυνατότητές τους να εκτελέσουν εργασίες συγκεκριμένου τύπου. Πρόκειται για πληροφορία την οποία εξ ορισμού οφείλει να διαθέτει μια επιχείρηση προκειμένου να ενταχθεί στο παραγωγικό δίκτυο και την οποία θεωρείται ότι επικαιροποιεί σε κάθε αλλαγή. Με το παραπάνω δεδομένο η μεθοδολογία αναλύεται στα ακόλουθα στάδια:

9.3.1. Στάδιο 1ο: Ανάλυση διεργασιών και προεπιλογή προμηθευτών

Το στάδιο αυτό έχει ως στόχο να προκρίνει όλους τους δυνατούς συνδυασμούς διεργασιών – παραγωγικών τμημάτων προμηθευτών που θα μπορούσαν να οδηγήσουν στην ολοκλήρωση μιας παραγγελίας. Αρχικά το στάδιο περιλαμβάνει την ανάλυση της παραγγελίας του πελάτη σε μια σειρά διεργασιών που θα πρέπει να λάβουν χώρα. Οι διεργασίες περιγράφονται ως προς την τεχνική τους διάσταση προκειμένου να είναι εφικτή η αντιστοίχισή τους με προμηθευτές που έχουν τη δυνατότητα να τις εκτελέσουν. Καθώς στην πλειονότητα των περιπτώσεων οι διεργασίες πρέπει να υλοποιηθούν με συγκεκριμένη σειρά, είναι εφικτή η δημιουργία ενός διαγράμματος ροής το οποίο αναπαριστά τους πιθανούς συνδυασμούς διεργασιών και μονάδων (παραγωγικών τμημάτων) προμηθευτών. Για κάθε διεργασία δημιουργούνται και αναπαρίστανται όλοι οι εφικτοί εναλλακτικοί συνδυασμοί, με

βάση από τη μία τα χαρακτηριστικά του τελικού προϊόντος και από την άλλη τις δεδομένες τεχνικές και παραγωγικές ικανότητες του κάθε προμηθευτή.

Προτείνεται η ακόλουθη διαδικασία για τη δημιουργία του διαγράμματος αντιστοίχισης διεργασιών:

- Ορίζεται ένας κόμβος εκκίνησης που αναπαριστά την έναρξη της παραγωγικής διαδικασίας της παραγγελίας
- Εντοπίζεται η πρώτη διεργασία που πρέπει να εκτελεστεί και τα τεχνικά χαρακτηριστικά της
- Ελέγχεται ποια εκ των μελών του δικτύου έχουν την τεχνική ικανότητα να εκτελέσουν τη διεργασία
- Για κάθε ένα εκ των μελών αυτών εντοπίζεται η παραγωγική (ή οι παραγωγικές αν είναι παραπάνω από μία) μονάδα του στην οποία μπορεί να ανατεθεί η εκτέλεση της διεργασίας και αναπαριστάται από ένα κόμβο στο διάγραμμα ο οποίος συνδέεται με τον κόμβο εκκίνησης
- Με τον ίδιο τρόπο εντοπίζεται η επόμενη διεργασία, η οποία οδηγεί σε νέους κόμβους συνδεδεμένους κάθε φορά με την προηγούμενη διεργασία και η διαδικασία συνεχίζεται μέχρι την ολοκλήρωση της ανάλυσης όλων των απαιτούμενων διεργασιών.

9.3.2. Στάδιο 2ο: Απόδοση στοιχείων κόστους

Ο ρόλος του σταδίου αυτού είναι η συλλογή και απόδοση στοιχείων κόστους πάνω στο διάγραμμα ροής το οποίο έχει κατασκευαστεί. Πρόκειται για στοιχεία κόστους τα οποία εξαρτώνται από τον κάθε προμηθευτή και την κάθε διεργασία αλλά που δεν απαιτούν τη διάθεση δεδομένων σε πραγματικό χρόνο (πχ για την ελεύθερη δυναμικότητα των παραγωγικών μονάδων).

Συγκεκριμένα ο διαχειριστής του ΔΔΠ εντοπίζει και καταγράφει τα ακόλουθα στοιχεία κόστους και χρόνου με βάση το διάγραμμα ροής του προηγούμενου σταδίου:

- Κόστη μεταφοράς προϊόντων μεταξύ των προμηθευτών: τα κόστη υπολογίζονται ή εκτιμώνται για το σύνολο της παρτίδας που αφορά στην παραγγελία που εξετάζεται και μπορεί να προκύπτουν είτε από ανά τεμάχιο

χρέωση είτε από πάγια χρέωση ανά μεταφορά της παρτίδας

- Χρόνοι μεταφοράς μεταξύ των προμηθευτών: πρόκειται για δεδομένο το οποίο ήδη διαθέτει ο διαχειριστής του δικτύου
- Κόστος παραγωγής για κάθε διεργασία ανά παραγωγική μονάδα που έχει εντοπιστεί ότι δύναται να την εκτελέσει
- Καθαρός χρόνος παραγωγής για κάθε διεργασία ανά παραγωγική μονάδα

Όπως είναι σαφές, ενώ για τα δύο πρώτα στοιχεία ο διαχειριστής του δικτύου διαθέτει ήδη τη σχετική πληροφορία, για τα δύο τελευταία κάτι τέτοιο δεν είναι βέβαιο. Εφόσον πρόκειται για κάποια τυποποιημένη διεργασία που το κόστος και ο χρόνος της αποτελούν δεδομένα για κάθε προμηθευτή τότε τα στοιχεία αυτά μπορούν να υπολογιστούν, με βάση φυσικά και το μέγεθος της παραγγελίας και άρα του όγκου επεξεργασίας κάθε διεργασίας. Η κατάσταση είναι πιο πολύπλοκη για τις περιπτώσεις που κάποια διεργασία δεν ανήκει στις τυποποιημένες. Στην περίπτωση αυτή είναι απαραίτητη η αποστολή των προδιαγραφών στα μέλη του δικτύου και αναμονή απαντήσεων από κάθε προμηθευτή σχετικά με το κόστος και τον απαιτούμενο χρόνο. Σημειώνεται ότι μέχρι το σημείο αυτό δεν έχει γίνει οποιοσδήποτε έλεγχος της κατάστασης των παραγωγικών μονάδων του κάθε προμηθευτή σε σχέση με το φόρτο τους από άλλες παραγγελίες. Ακόμα και στην περίπτωση που απαιτηθεί να ληφθεί πληροφορία χρόνου και κόστους, όπως αναφέρθηκε παραπάνω, αυτή λαμβάνεται από το τμήμα σχεδίασης/ προγραμματισμού του κάθε προμηθευτή (το οποίο μπορεί άμεσα να απαντήσει σε σχετικά αιτήματα) και όχι από την ίδια την παραγωγή καθώς κάτι τέτοιο θα απαιτούσε μη αποδεκτό χρόνο.

9.3.3. Στάδιο 3ο: Μοντελοποίηση με χρονισμένα αυτόματα και αναλυτική επίλυση προβλήματος

Πρόκειται για το κυρίως μέρος της προτεινόμενης μεθοδολογίας που έχει ως σκοπό τον καθορισμό της βέλτιστης ανάθεσης των διεργασιών στους υποψήφιους προμηθευτές / μέλη του ΔΔΠ με τελικό στόχο την ελαχιστοποίηση του συνολικού κόστους και το σεβασμό των προθεσμιών. Το κρισιμότερο σημείο προς αντιμετώπιση είναι το γεγονός ότι τα μέλη του δικτύου, εκτός από τις διεργασίες που τους

ανατίθενται από το διαχειριστή του δικτύου, έχουν να εκτελέσουν κι άλλες παραγωγικές διεργασίες για δικούς τους πελάτες. Έτσι ακόμα κι αν για παράδειγμα μια παραγωγική μονάδα ενός προμηθευτή χρειάζεται κάποιο δεδομένο χρόνο για να ολοκληρώσει μια διεργασία, αν η μονάδα βρίσκεται στο στάδιο εκτέλεσης άλλης διεργασίας δε θα μπορέσει να ανταποκριθεί στον προβλεπόμενο χρόνο. Το πρόβλημα αυτό, φυσικά, υπεισέρχεται σε κάθε προσπάθεια χρονοπρογραμματισμού παραγωγικών δικτύων εκτός των περιπτώσεων που το σύνολο της παραγωγικής ικανότητας των συμμετεχόντων μελών διατίθεται αποκλειστικά στο δίκτυο, οπότε και ο προγραμματισμός ουσιαστικά είναι ιδιαίτερα απλός με δεδομένα τα κόστη παραγωγής. Στον πραγματικό κόσμο, όμως, κάτι τέτοιο είναι ιδιαίτερα σπάνιο και άρα δε μπορεί να βρει εφαρμογή στη γενική περίπτωση. Για το λόγο αυτό προτείνεται η δημιουργία ενός συστήματος βασισμένου στα χρονισμένα αυτόματα δύο επιπέδων:

- Το πρώτο επίπεδο βρίσκεται υπό τη διαχείριση του τελικού κατασκευαστή / διαχειριστή του δικτύου και αναπαριστά τις διεργασίες που πρέπει να εκτελεστούν για να καλυφθούν οι ανάγκες του πελάτη
- Το δεύτερο επίπεδο βρίσκεται υπό τη διαχείριση των μελών του δικτύου. Κάθε προμηθευτής αναπτύσσει το δικό του μοντέλο το οποίο αναπαριστά την παραγωγή του και τις διεργασίες που έχει να επιτελέσει ως αποτέλεσμα των παραγγελιών που έχει δεσμευτεί εκτός δικτύου να υλοποιήσει. Η διαχείριση του μοντέλου αυτού και η επικαιροποίησή του γίνεται από κάθε μέλος του δικτύου εξ αποστάσεως, προκειμένου από τη μία να βρίσκεται συγκεντρωμένη η απαιτούμενη πληροφορία «σχεδόν πραγματικού χρόνου» στο σύστημα χρονοπρογραμματισμού για κάθε προμηθευτή, αλλά από την άλλη η πληροφορία αυτή να μην είναι διαθέσιμη στα μέλη του δικτύου και ενδεχομένως (εφόσον υπάρξει σχετική πρόβλεψη σε τεχνικό επίπεδο) ούτε στον ίδιο το διαχειριστή του. Στην πραγματικότητα πρόσβαση στα μοντέλα του κάθε προμηθευτή απαιτείται να έχει μόνο η μηχανή βελτιστοποίησης η οποία επιλύει τα προβλήματα των χρονισμένων αυτομάτων.

Προκειμένου να βρεθεί η βέλτιστη ανάθεση των διεργασιών, η μηχανή βελτιστοποίησης εκτελεί αναζήτηση ακριβώς με τον ίδιο τρόπο που περιγράφηκε στα προηγούμενα κεφάλαια. Ο συγχρονισμός στην περίπτωση αυτή αντί να γίνεται ανά μηχανή γίνεται ανά παραγωγική μονάδα, ενώ σε αντιστοιχία με τις εργασίες, βρίσκονται οι ροές διεργασιών που πρέπει να εκτελεστούν. Εδώ διακρίνονται 2

περιπτώσεις:

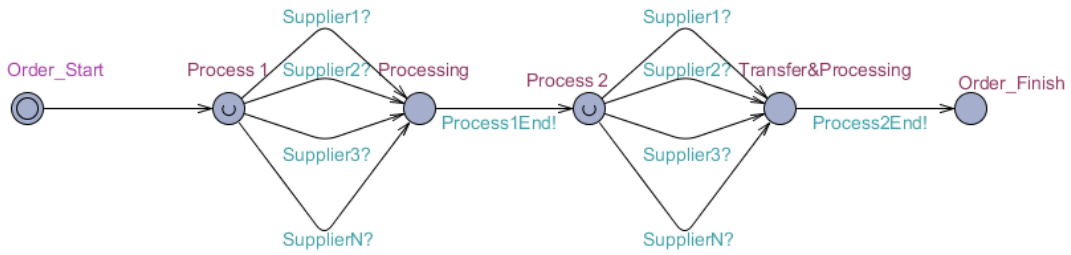
- 1η περίπτωση: Οι προμηθευτές έχουν απόλυτη δέσμευση και θέτουν σε απόλυτη προτεραιότητα τις προερχόμενες από το δίκτυο παραγγελίες. Στην περίπτωση αυτή δεν τίθεται θέμα καθυστέρησης κάποιας διεργασίας λόγω άλλης υπό εκτέλεση από ένα προμηθευτή. Εν τούτοις δίνοντας προτεραιότητα στο να εκτελεστεί η προερχόμενη από το δίκτυο διεργασία, ενδέχεται αυτό να δημιουργήσει αυξημένα κόστη (λόγω καθυστερήσεων) στις λοιπές παραγγελίες του προμηθευτή. Δεδομένου ότι από τη μία ο προμηθευτής επιθυμεί να εξυπηρετήσει το δίκτυο ενώ από την άλλη δεν έχει λόγο να επωμιστεί το βάρος κόστους καθυστέρησης των παραγγελιών του, το προτεινόμενο μοντέλο προσθέτει το κόστος αυτό στο κόστος εκτέλεσης των διεργασιών της παραγγελίας του ΔΔΠ. Με τον τρόπο αυτό ουσιαστικά το κόστος της κάθε διεργασίας στον κάθε προμηθευτή (ή αλλιώς η τιμή στην οποία ο προμηθευτής χρεώνει την εκτέλεσή της) τροποποιείται ανάλογα με το φόρτο του και την «αναστάτωση» που επιφέρει στον χρονοπρογραμματισμό του.
- 2η περίπτωση: Οι προμηθευτές εντάσσουν τις παραγγελίες που προέρχονται από το δίκτυο στον χρονοπρογραμματισμό τους χωρίς απόλυτη προτεραιότητα. Στην περίπτωση αυτή μπορεί κάποιος προμηθευτής ουσιαστικά να καθυστερήσει την εκτέλεση μιας διεργασίας που του ανατίθεται από το ΔΔΠ βάζοντας κάποια(-ες) άλλη(-ες) σε προτεραιότητα. Φυσικά αυτό μπορεί να επιφέρει καθυστέρηση και άρα πρόσθετο κόστος στην παράδοση του παραγγελίας από τον τελικό κατασκευαστή στον πελάτη του, αλλά από την άλλη μπορεί να οδηγήσει σε εξοικονόμηση συνολικά για το δίκτυο αν αντιμετωπιστεί ενιαία (πχ μέσω της μείωσης του κόστους άλλων παραγγελιών που έχουν αναλάβει οι προμηθευτές).

Η προσέγγιση με βάση τα χρονισμένα αυτόματα η οποία προτείνεται μπορεί να καλύψει και τις δύο παραπάνω περιπτώσεις καθώς και με κατάλληλη προσαρμογή κάθε ενδιάμεση περίπτωση. Εν τούτοις η 2η περίπτωση απαιτεί στη συνέχεια ένα μηχανισμό διαμοιρασμού του οφέλους εξοικονόμησης μεταξύ των επιχειρήσεων που συμμετέχουν. Μηχανισμοί τέτοιου τύπου με βάση τις αρχές της Θεωρίας Παιγνίων έχουν προταθεί στη βιβλιογραφία αλλά η εξέτασή τους δεν εμπίπτει στο πλαίσιο της παρούσας διατριβής. Για το λόγο αυτό αλλά και δεδομένου ότι η 1η περίπτωση είναι

η πιο συνηθισμένη στην πράξη, στη συνέχεια εξετάζεται ενδελεχώς αυτή. Τονίζεται εν τούτοις ότι σε επίπεδο διαδικασίας βελτιστοποίησης και εξαιρώντας το διαμοιρασμό του κόστους ή κέρδους, η προσέγγιση μπορεί με πλήρη αποτελεσματικότητα να καλύψει και τη δεύτερη των περιπτώσεων.

Στη συνέχεια αναλύεται διεξοδικά η προτεινόμενη προσέγγιση μέσα από την παρουσίαση ενός βασικού μοντέλου επίλυσης. Θεωρούμε το πρόβλημα μιας παραγγελίας η οποία αποτελείται από δύο διαδοχικές διεργασίες (Process1 & Process2) αντίστοιχα. Η κάθε διεργασία μπορεί να αναληφθεί και να εκτελεστεί από μια σειρά διαφορετικών προμηθευτών που ανήκουν στο ίδιο δυναμικό δίκτυο παραγωγής.

Πρώτο βήμα αποτελεί η ανάπτυξη του αυτομάτου εκτέλεσης της παραγγελίας:



Σχήμα 63: Το γενικό αυτόματο της παραγγελίας

Όπως φαίνεται και στο σχήμα, το αυτόματο ξεκινάει από τη θέση έναρξης της επεξεργασίας της παραγγελίας και μεταβαίνει στον κόμβο Process1 όπου πρέπει να γίνει η επιλογή του προμηθευτή που θα εκτελέσει την πρώτη διεργασία. Αν δεν υπάρχει κάποιος συγκεκριμένος λόγος, το αυτόματο δεν περιμένει στη θέση Process1 αλλά προσπαθεί να μεταβεί μέσα από κάποια διαθέσιμη διαδρομή στη θέση Processing. Οι διαθέσιμες διαδρομές αναπαριστούν και συνδέονται αντίστοιχα με τα παραγωγικά τμήματα των προμηθευτών Supplier1, Supplier2 ως SupplierN. Τα ομώνυμα κανάλια Supplier1?, Supplier2? κλπ είναι αυτά μέσω των οποίων γίνεται ο συγχρονισμός με τα επιμέρους αυτόματα των προμηθευτών. Στην γενική περίπτωση του μοντέλου θεωρούμε ότι όλοι οι προμηθευτές μπορούν να εκτελέσουν όλες τις διεργασίες. Φυσικά στην πράξη (όπως θα παρουσιαστεί και στην εφαρμογή στη συνέχεια) κάτι τέτοιο δεν ισχύει αλλά κάθε διεργασία μπορεί να πραγματοποιηθεί από

ένα υποσύνολο μόνο των προμηθευτών και άρα να οδηγηθούμε σε ένα απλούστερο τελικά μοντέλο.

Μεταβαίνοντας στη θέση Processing, ξεκινάει η εκτέλεση της πρώτης διεργασίας. Ανάλογα ποια διαδρομή ακολουθήθηκε για τη μετάβαση από τη θέση Process1 στη θέση Processing, ο αντίστοιχος προμηθευτής ξεκινάει να εκτελεί η διεργασία. Κατά τη διάρκεια εκτέλεσης το αυτόματο θα παραμείνει στη θέση Processing όσες χρονικές μονάδες απαιτούνται προκειμένου ο συγκεκριμένος προμηθευτής να ολοκληρώσει τη διεργασία.

Το γενικό αυτόματο «παρακολουθείται» από δύο global μεταβλητες: Τη μεταβλητή του ολικού χρόνου (glc) και τη μεταβλητή του κόστους (cost).

Όσο η διεργασία εκτελείται και άρα παραμένει το αυτόματο στη θέση Processing, η μεταβλητή του ολικού χρόνου (η οποία είχε αρχική τιμή μηδέν) αυξάνει κατά όσες χρονικές μονάδες διαρκεί η επεξεργασία της πρώτης διεργασίας. Όταν αυτή ολοκληρωθεί τότε μέσω του καναλιού συγχρονισμού Process1End! το αυτόματο μεταβαίνει στην επόμενη διεργασία που πρέπει να προγραμματίσει, δηλαδή στη θέση Process2. Κατά τη διάρκεια της μετάβασης αυτής επιλέγεται να ενημερωθεί η μεταβλητή ολικού κόστους (cost'). Η μεταβλητή αυξάνεται τόσο όσο είναι το κόστος του προμηθευτή που εκτέλεσε τη διεργασία – σημειώνεται ότι το σύστημα θεωρεί ως δεδομένο για κάθε προμηθευτή το κόστος παραγωγής του ανά διεργασία, όπως αναλύθηκε προηγούμενα.

Περνώντας στη θέση Process2, το αυτόματο έχει να ελέγξει τις εναλλακτικές διαδρομές προς τη θέση Transfer&Processing, κατά πλήρη αντιστοιχία με την θέση Process1 που περιγράφηκε ανωτέρω. Η διαφορά στο σημείο αυτό έγκειται στο χρόνο που η διεργασία θα μείνει στη θέση Processing. Δεδομένου ότι κάποιος προμηθευτής ξεκίνησε την επεξεργασία του προϊόντος εκτελώντας την πρώτη διεργασία, προκειμένου κάποιος άλλος προμηθευτής να μπορέσει να συνεχίσει, θα πρέπει πρώτα να γίνει η μεταφορά από τον πρώτο προμηθευτή στον δεύτερο η οποία θα απαιτήσει συγκεκριμένο χρόνο και θα έχει συγκεκριμένο κόστος. Για το λόγο αυτό, το αυτόματο μεταβαίνοντας στη θέση Transfer&Processing, έχοντας δώσει εντολή σε κάποιο προμηθευτή να προχωρήσει στην εκτέλεση της $2^{ης}$ διεργασίας, θα παραμείνει στη θέση αυτή όχι μόνο όσο κρατάει η εκτέλεση της διεργασίας: ο χρόνος αυτός θα προσαυξηθεί με το χρόνο που απαιτείται προκειμένου να μεταφερθεί το προϊόν (ως

παρτίδα) από τον προμηθευτή της 1ης στον προμηθευτή της 2ης διεργασίας. Κατά τόσο χρόνο θα αυξηθεί αντίστοιχα και η μεταβλητή ολικού χρόνου glc.

Αντίστοιχα, σε ότι αφορά στη μεταφορά της παρτίδας, αντιμετωπίζεται και το θέμα του κόστους μεταφοράς. Συγκεκριμένα η ολική μεταβλητή cost' στο σημείο αυτό προσαυξάνεται:

- κατά το καθαρό κόστος για την εκτέλεση της διεργασίας από τον προμηθευτή που επελέγη και
- κατά το κόστος μεταφοράς από τον προμηθευτή της 1ης στον προμηθευτή της 2ης διεργασίας.

Τόσο οι χρόνοι όσο και τα κόστη μεταφοράς μεταξύ όλων των μελών του δικτύου (προμηθευτών) θεωρούνται δεδομένα ανά παρτίδα, όπως αναλύθηκε στην αρχή του κεφαλαίου. Φυσικά σημειώνεται ότι τόσο ο πρόσθετος χρόνος όσο και το πρόσθετο κόστος λαμβάνονται υπόψη μόνο αν δύο διαδοχικές διεργασίες εκτελούνται από διαφορετικούς προμηθευτές. Στην ειδικότερη περίπτωση που ο ίδιος προμηθευτής εκτελέσει διαδοχικές διεργασίες, τα κόστη αυτά απαλείφονται και λαμβάνεται υπόψη μόνο το καθαρό κόστος και ο καθαρός χρόνος για την εκτέλεση των διεργασιών. Τελικά φτάνοντας το αυτόματο στην τελική θέση που η παραγγελία θεωρείται ολοκληρωμένη, η μεταβλητή του κόστους θα περιλαμβάνει όλα τα πραγματικά κόστη μεταφοράς και επεξεργασίας, τα οποία είναι συνάρτηση του ποιες διαδρομές ακολουθήθηκαν και ποιοι προμηθευτές αντίστοιχα εκτέλεσαν την κάθε διεργασία. Κατ' αντιστοιχία η μεταβλητή του ολικού χρόνου θα έχει τιμή ίση με το συνολικό χρόνο που έχει διέλθει από την έναρξη μέχρι την ολοκλήρωση της παραγγελίας. Στο σημείο αυτό, στον τελευταίο κόμβο του αυτομάτου, χρειάζεται να υπολογιστεί στο συνολικό κόστος το δυνητικό κόστος που οφείλεται στο χρόνο παράδοσης της παραγγελίας στον πελάτη έναντι του αρχικά συμφωνηθέντος. Έτσι η μεταβλητή του κόστους προσαυξάνεται:

- κατά το κόστος αποθήκευσης της παρτίδας * το διάστημα που μεσολαβεί από την ολοκλήρωση της παραγγελίας μέχρι τη συμφωνηθείσα παράδοση (αν η παραγγελία έχει ολοκληρωθεί νωρίτερα)

  ή

- κατά το κόστος καθυστέρησης παράδοσης * το διάστημα που μεσολαβεί από τη συμφωνηθείσα παράδοση μέχρι το χρόνο ολοκλήρωσης της παραγγελίας (αν η παραγγελία έχει ολοκληρωθεί μετά το συμφωνηθέντα χρόνο)

Φυσικά το ύψος του κόστους αποθήκευσης όσο αυτό του κόστους καθυστέρησης εξαρτώνται από τη συμφωνία που κλείνεται με τον πελάτη κατά τη στιγμή της παραγγελίας και δύναται να είναι και μηδενικά σε ορισμένες ειδικές περιπτώσεις.

Φτάνοντας στο σημείο αυτό, στη μεταβλητή του κόστους για την παραγγελία έχουν συνυπολογιστεί:

- τα κόστη εκτέλεσης των διεργασιών,
- τα κόστη μεταφοράς και
- τα κόστη νωρίτερης / αργότερης περάτωσης,

δηλαδή όλες οι παράμετροι κόστους που αναλύθηκαν στην μοντελοποίηση του προβλήματος των Δυναμικών Δικτύων Παραγωγής. Παρ' όλα αυτά το αυτόματο δε μπορεί να θεωρηθεί ακόμα ολοκληρωμένο για να καλύψει τα πραγματικά προβλήματα που παρουσιάζονται στα παραγωγικά δίκτυα στην πράξη. Ο λόγος έγκειται στο γεγονός ότι, όπως αναλύθηκε προηγούμενα, οι εταιρείες – προμηθευτές, οι οποίες είναι μέλη ενός ΔΔΠ, δεν παύουν να έχουν ανεξάρτητη δραστηριότητα και άρα να έχουν να εκτελέσουν διεργασίες πρόσθετες αυτών που τους ανατίθενται μέσα από τις παραγγελίες του δικτύου. Καθώς όμως τα παραγωγικά δίκτυα κατά κόρον βασίζονται σε μεγάλες εταιρείες – τελικούς κατασκευαστές για τις οποίες οι χρόνοι παράδοσης είναι ιδιαίτερα δεσμευτικοί, δεν αφήνονται περιθώρια για χρονικές καθυστερήσεις στους προμηθευτές, προκειμένου να βάλουν σε προτεραιότητα δικές τους παραγγελίες. Έτσι αυτό που πρέπει να εξεταστεί είναι το πόσο «κοστίζει» στους προμηθευτές μέλη του ΔΔΠ να εκτελέσουν κατά απόλυτη προτεραιότητα τις διεργασίες που προέρχονται από παράγγειλες του δικτύου, εις βάρος ενδεχομένως της τήρησης δικών τους προθεσμιών παράδοσης. Ουσιαστικά και προκειμένου να μην επιβαρυνθούν το κόστος αυτό, το σύστημα θα πρέπει να το λαμβάνει υπόψη του ουσιαστικά ως πρόσθετο κόστος παραγωγής το οποίο ο προμηθευτής θα το «χρεώσει» (στο σύνολό του ή σε ένα ποσοστό του) στο κόστος των διεργασιών που θα εκτελέσει για τις παραγγελίες του δικτύου. Έτσι για όλους τους προμηθευτές, παρά το γεγονός ότι η δυναμικότητά μπορεί να επαρκεί για να εκτελέσουν μια διεργασία, λαμβάνεται επιπρόσθετα υπόψη σε κάθε παραγγελία ο φόρτος που ήδη

έχουν από ανεξάρτητες υποχρεώσεις τους. Εφόσον η αποδοχή της παραγγελίας του δικτύου δε τους προξενεί πρόσθετο κόστος ή καθυστερήσεις, τότε οι πιθανότητα να επιλεγούν από το εργαλείο βελτιστοποίησης είναι συνάρτηση μόνο των στοιχείων κόστους και χρόνου της παραγωγής και των μεταφορικών τους. Αν όμως οι παραγωγικές μονάδες ενός προμηθευτή είναι δεσμευμένες από ανεξάρτητες παραγγελίες που δεν έχουν περιθώρια καθυστέρησης, τότε η αποδοχή μιας διεργασίας του δικτύου σημαίνει ένα πρόσθετο κόστος που το σύστημα χρονοπρογραμματισμού θα πρέπει να συνυπολογίσει.

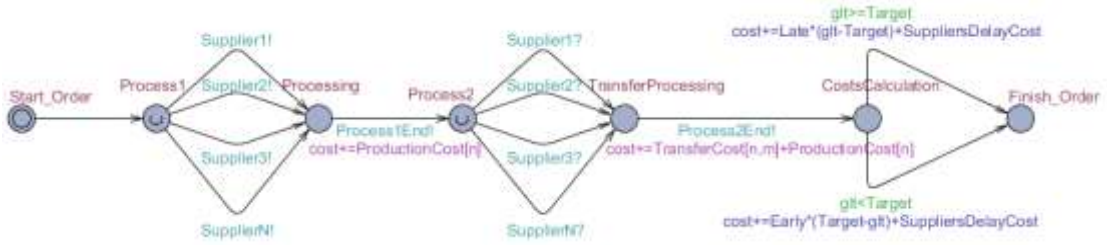
9.3.4. Διαδικασία αντιμετώπισης προβλημάτων σε ΔΔΠ

Βάσει των όσων αναλύθηκαν παραπάνω, παρατίθεται στη συνέχεια μια σειρά ενεργειών στις οποίες βασίζεται η προτεινόμενη προσέγγιση:

- Κάθε μέλος του δικτύου κατά τη είσοδό του αναλαμβάνει να δημιουργήσει ένα ανεξάρτητο αυτόματο που αναπαριστά τα παραγωγικά του τμήματα. Το αυτόματο αυτό εντάσσεται στο σύστημα χρονοπρογραμματισμού του δικτύου.
- Σε περιοδική βάση κάθε μέλος του δικτύου «ανεβάζει» στο σύστημα χρονοπρογραμματισμού το πρόγραμμα παραγωγής του σε επίπεδο παραγωγικού τμήματος καθώς και τις προθεσμίες παράδοσης στις οποίες έχει δεσμευτεί για τις ανεξάρτητες παραγγελίες του. Τα στοιχεία αυτά δεν γίνονται γνωστά στα μέλη του δικτύου καθώς στο σύστημα χρονοπρογραμματισμού κάθε μέλος έχει το δικό του αυτόνομο χώρο τήρησης και επικαιροποίησης στοιχείων σε (σχεδόν) πραγματικό χρόνο. Σημειώνεται ότι αρκεί το ανέβασμα ενός αρχείου xml σε τακτική βάση προκειμένου να γίνεται με αυτοματοποιημένο τρόπο η επικαιροποίηση.
- Κατά τη λήψη και τον χρονοπρογραμματισμό μιας παραγγελίας, το σύστημα εξετάζει τα συνδυαστικά σενάρια συμμετοχής των παραγωγικών τμημάτων των μελών του δικτύου στην υλοποίησή της.
- Σε κάθε συνδυαστικό σενάριο που συμμετέχει ένας αριθμός προμηθευτών, δημιουργείται από ένα νέο χρονοπρόγραμμα για κάθε προμηθευτή το οποίο περιλαμβάνει τόσο τις αρχικές παραγγελίες που είχε εισάγει όσο και τις διεργασίες που προέρχονται από την παραγγελία του δικτύου.

- Εξετάζεται κατά πόσον στους συμμετέχοντες προμηθευτές σε κάθε σενάριο προκύπτουν πρόσθετα κόστη λόγω καθυστερήσεων παράδοσης στις δικές τους παραγγελίες, βάσει και των στοιχείων παραγγελίας που έχουν διαθέσει στο σύστημα
- Στο τελικό υπολογιζόμενο κόστος ανά σενάριο, λαμβάνονται υπόψη τα πρόσθετα αυτά κόστη για τους συμμετέχοντες προμηθευτές
- Τελικά το σύστημα προκρίνει το σενάριο το οποίο έχει το μικρότερο συνολικό κόστος για το δίκτυο, συμπεριλαμβανομένου (τμήματος) του κόστους κάλυψης των καθυστερήσεων των προμηθευτών που τελικά συμμετέχουν.

Λαμβάνοντας υπόψη όλα τα παραπάνω παρουσιάζονται στη συνέχεια τα αυτόματα της προτεινόμενης προσέγγισης όπως τελικά διαμορφώνονται.



Σχήμα 64: Αυτόματο παραγγελίας του ΔΔΠ με κόστη

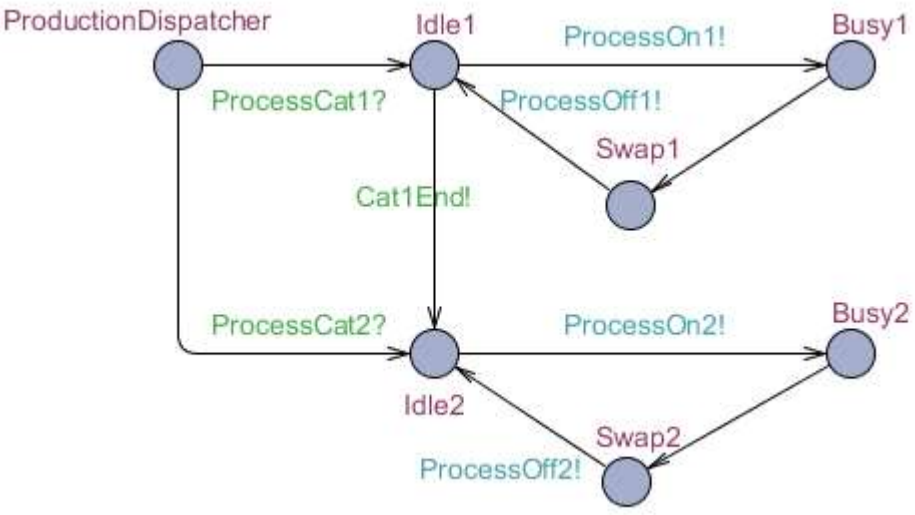
Στο παραπάνω σχήμα φαίνεται το αυτόματο μιας παραγγελίας με τους υπολογισμούς του κόστους για κάθε εναλλακτικό συνδυασμό προμηθευτών. Η μεταβλητή cost αθροίζει τις μεταβλητές:

- ProductionCost[n]: εκφράζει το κόστος του κάθε process ανάλογα με τον προμηθευτή n
- TransferCost [n,m]: εκφράζει το κόστος μεταφοράς μιας παρτίδας προϊόντων από τον προμηθευτή n στον προμηθευτή m για την εκτέλεση διαδοχικών εργασιών. Σε περίπτωση που δύο διαδοχικές διεργασίες εκτελούνται από τον ίδιο προμηθευτή το κόστος μεταφοράς για αυτές απαλοίφεται.
- Early/Late: Πολλαπλασιαζόμενο με το χρόνο κατά τον οποίο είτε καθυστερεί είτε προπορεύεται η ολοκλήρωση της παραγγελίας έναντι τις συμφωνηθείσας με τον πελάτη ημερομηνίας, εκφράζει το αντίστοιχο κόστος αποθήκευσης ή καθυστέρησης παράδοσης.

- SuppliersDelayCost[1,2,..,n]: Πρόκειται για πίνακα κόστους που δημιουργείται από τον υπολογισμό του πρόσθετου κόστους που επιφέρει η παραγγελία στους προμηθευτές που συμμετέχουν στην υλοποίησή της.

Στο σχήμα που ακολουθεί παρουσιάζεται το αυτόματο ενός τυπικού προμηθευτή που έχει δύο γραμμές παραγωγής / παραγωγικά τμήματα τα οποία είτε λειτουργούν αυτόνομα είτε σειριακά, ανάλογα με τα υπό παραγωγή είδη. Στον κόμβο ProductionDispatcher γίνεται η δρομολόγηση των διεργασιών:

- Εφόσον το προϊόν πρέπει να εισαχθεί πρώτα στη γραμμή παραγωγής 1 και κατόπιν στη γραμμή 2 τότε ο Dispatcher δρομολογεί τη διεργασία στον πρώτο idle κόμβο. Από κει και πέρα ενεργοποιείται η διαδικασία επεξεργασίας και μέτρησης χρόνου και κόστους κατά πλήρη αντιστοιχία με αυτή που περιγράφηκε στα προηγούμενα κεφάλαια για τη μοντελοποίηση παραγωγικών πόρων.
- Μετά την ολοκλήρωση της επεξεργασίας στη γραμμή 1, το προϊόν, ανάλογα με τα ορίσματα εισόδου που προέρχονται από την ανάλυση της παραγγελίας είτε θεωρείται ολοκληρωμένο οπότε βγαίνει από την επεξεργασία στο συγκεκριμένο προμηθευτή, είτε χρειάζεται να διέλθει και από τη γραμμή παραγωγής 2 οπότε μεταβαίνει σε αυτή ακολουθώντας την ίδια λογική. Σημειώνεται ότι από τη στιγμή που ένα προϊόν μπει σε μια γραμμή παραγωγής αυτή χαρακτηρίζεται ως busy και ως εκ τούτου κανένα άλλο προϊόν μέχρι την ολοκλήρωση της επεξεργασίας δε μπορεί να εισαχθεί προς παραγωγή και αντίστοιχα καμία πρόσθετη διεργασία δε μπορεί να εκτελεστεί από τον προμηθευτή.



Το παραπάνω αυτόματο αποτελεί, όπως αναφέρθηκε μια τυπική περίπτωση ενός προμηθευτή/ παραγωγού με δύο σειριακά παραγωγικά τμήματα. Παρ' όλα αυτά η μεγάλη προστιθέμενη αξία της προτεινόμενης μεθόδου αφορά στο γεγονός ότι κάθε προμηθευτής έχει τη δυνατότητα να αναπαραστήσει με ακρίβεια τον δικό του τρόπο λειτουργίας των παραγωγικών του μονάδων ώστε να αποτυπώνεται ακριβώς πάνω στο αυτόματο. Με τον τρόπο αυτό μπορούν να καλυφθούν όλες οι περιπτώσεις παραγωγικών δομών, από τις τυπικές (flow-shop, jobshop κλπ) μέχρι σύνθετες δομές με παράλληλες γραμμές παραγωγής ίδιων ή διαφορετικών χαρακτηριστικών όπως και διάφορες συνδυαστικές δομές (πχ αρχικά τύπου flowshop και στη συνέχεια τύπου jobshop). Σε κάθε περίπτωση φυσικά, σε όσο μεγαλύτερη λεπτομέρεια αποτυπώνονται οι παραγωγικές δομές τόσο πιο πολύ αυξάνουν τα δεδομένα και ο χρόνος επίλυσης του προβλήματος – έτσι προτείνεται στην περίπτωση των Δυναμικών Δικτύων Παραγωγής η μοντελοποίηση να γίνεται σε επίπεδο παραγωγικής μονάδας ή παραγωγικού τμήματος και όχι σε μεγαλύτερη λεπτομέρεια (πχ σε επίπεδο κέντρου κατεργασίας). Εν τούτοις αν και μια μεγάλη λεπτομέρεια αυξάνει τον υπολογιστικό χρόνο, σε επίπεδο διαδικασίας επίλυσης η προτεινόμενη λύση δεν έχει περιορισμούς οποιουδήποτε τύπο.

9.4. Εφαρμογή σε πραγματική περίπτωση

Προκειμένου να επιβεβαιωθεί στην πράξη η αποτελεσματικότητα της προτεινόμενης προσέγγισης, αυτή εφαρμόστηκε σε ένα πραγματικό πρόβλημα ενός επιχειρηματικού δικτύου που αποτελείται από έναν τελικό κατασκευαστή και από τρεις προμηθευτές/ υπεργολάβους του. Σε ότι αφορά στον τελικό κατασκευαστή πρόκειται για την εταιρεία ΚΑΖΗΣ ΑΒΕΕ, μια βιομηχανική μονάδα μεσαίου μεγέθους η οποία εδρεύει στην Αττική και παράγει κατά παραγγελία λέβητες θέρμανσης σε ένα πλήθος επιλογών και διαστάσεων. Η εταιρεία έχει τρεις βασικούς προμηθευτές/υπεργολάβους οι οποίοι αναλαμβάνουν κατά βάση την προετοιμασία (χύτευση και κατεργασία) των σωμάτων των λεβήτων ενώ η βιομηχανική μονάδα της επιχείρησης κάνει την τελική συναρμολόγηση, τοποθετώντας τον πίνακα ελέγχου, μονώσεις και διάφορα εξαρτήματα όπως σωλήνες προσαγωγής, ελάσματα στεγανοποίησης κλπ. Παρά το γεγονός ότι με τους προμηθευτές της δεν έχει συνάψει συγκεκριμένη συμφωνία δημιουργίας παραγωγικού δικτύου, στην πραγματικότητα οι τέσσερις επιχειρήσεις λειτουργούν σε αυτή τη λογική, χάρη στην ιδιαίτερα μακροχρόνια αρμονική συνεργασία τους αλλά και στο γεγονός ότι η βιομηχανία αποτελεί το βασικό πελάτη

των τριών προμηθευτών/ υπεργολάβων της και σε αυτή καταλήγει το 60% ως 70% της ολικής ετήσιας παραγωγής τους. Καθώς τα σώματα των λεβήτων είναι εξ ορισμού χυτά (από χυτοσίδηρο), οι προμηθεύτριες της βιομηχανίας έχουν μονάδες χύτευσης και μονάδες κατεργασίας χυτών εξαρτημάτων.

Συγκεκριμένα, από τους τέσσερις προμηθευτές/ υπεργολάβους, οι Π1 και Π2 διαθέτουν χυτήριο και κατασκευάζουν τα στοιχεία για τοιχώματα των λεβήτων (εμπρόσθιο, οπίσθιο και ενδιάμεσα στοιχεία), ενώ οι Π3 και Π4 διαθέτουν μονάδα κατεργασίας χυτών εξαρτημάτων και αναλαμβάνουν αρχικά την κατεργασία των στοιχείων (λείανση, δημιουργία οπών κλπ), το μοντάρισμα τους σε ενιαίο σώμα καθώς και τον έλεγχο στεγανότητας και αντοχής του σώματος.

Προκειμένου να ελεγχθεί η προτεινόμενη προσέγγιση με πραγματικά στοιχεία, εξετάστηκε μια τυχαία μεγάλη παραγγελία που έγινε προς την εταιρεία ΚΑΖΗΣ για την παραγωγή 400 όμοιων λεβήτων ειδικών χαρακτηριστικών με χρόνο παράδοσης 25 ημέρες. Από την ανάλυση που πραγματοποίησε το τμήμα σχεδιασμού της εταιρείας προέκυψε ότι προκειμένου να μπορέσει η εταιρεία να συναρμολογήσει την παραγγελία χωρίς καθυστέρηση θα έπρεπε να έχει εντός 20 ημερών το σύνολο των σωμάτων των λεβήτων διαθέσιμα.

Στους πίνακες που ακολουθούν παρουσιάζονται τα κόστη (για παρτίδα 400 λεβήτων) όπως και οι χρόνοι μεταφοράς μεταξύ των 4 προμηθευτών/υπεργολάβων της εταιρείας:

$$c_{km} = \begin{bmatrix} & \Pi1 & \Pi2 & \Pi3 & \Pi4 \\ \Pi1 & - & 400 & 350 & 300 \\ \Pi2 & 400 & - & 350 & 300 \\ \Pi3 & 350 & 350 & - & 550 \\ \Pi4 & 300 & 300 & 550 & - \end{bmatrix}$$

$$t_{km} = \begin{bmatrix} & \Pi1 & \Pi2 & \Pi3 & \Pi4 \\ \Pi1 & - & 2 & 1 & 1 \\ \Pi2 & 2 & - & 1 & 1 \\ \Pi3 & 1 & 1 & - & 2 \\ \Pi4 & 1 & 1 & 2 & - \end{bmatrix}$$

Για την παρτίδα αυτή το κόστος αποθήκευσης ανά ημέρα καθώς και το κόστος καθυστέρησης της παράδοσης της υπολογίστηκαν από το τμήμα σχεδιασμού της εταιρείας σε:

$$c_{\alpha} = 150 \text{ €} / \text{ ημέρα}$$

και

$$c_{\beta} = 600 \text{ €} / \text{ ημέρα}$$

Στον πίνακα που ακολουθεί φαίνονται τα βασικά χαρακτηριστικά και οι ικανότητες των 4 προμηθευτών για κάθε μία από τις τέσσερις διεργασίες που πρέπει να ολοκληρωθούν προκειμένου να ολοκληρωθούν τα σώματα των λεβήτων και να παραληφθούν από την εταιρεία προκειμένου να ξεκινήσει η συναρμολόγησή τους.

	Π1	Π2	Π3	Π4
ΧΥΤΕΥΣΗ				
Χρόνος Αναμονής (h)	5	4	-	-
Χρόνος Προετοιμασίας (h)	1	0.5	-	-
ΚΑΤΕΡΓΑΣΙΑ				
Χρόνος Αναμονής (h)	-	-	3	21
Χρόνος Προετοιμασίας (h)	-	-	0.5	2
ΜΟΝΤΑΡΙΣΜΑ				
Χρόνος Αναμονής (h)	-	-	4	12
Χρόνος Προετοιμασίας (h)	-	-	0,5	2
ΕΛΕΓΧΟΣ				
Χρόνος Αναμονής (h)	-	-	4	3
Χρόνος Προετοιμασίας (h)	-	-	0.5	0.5

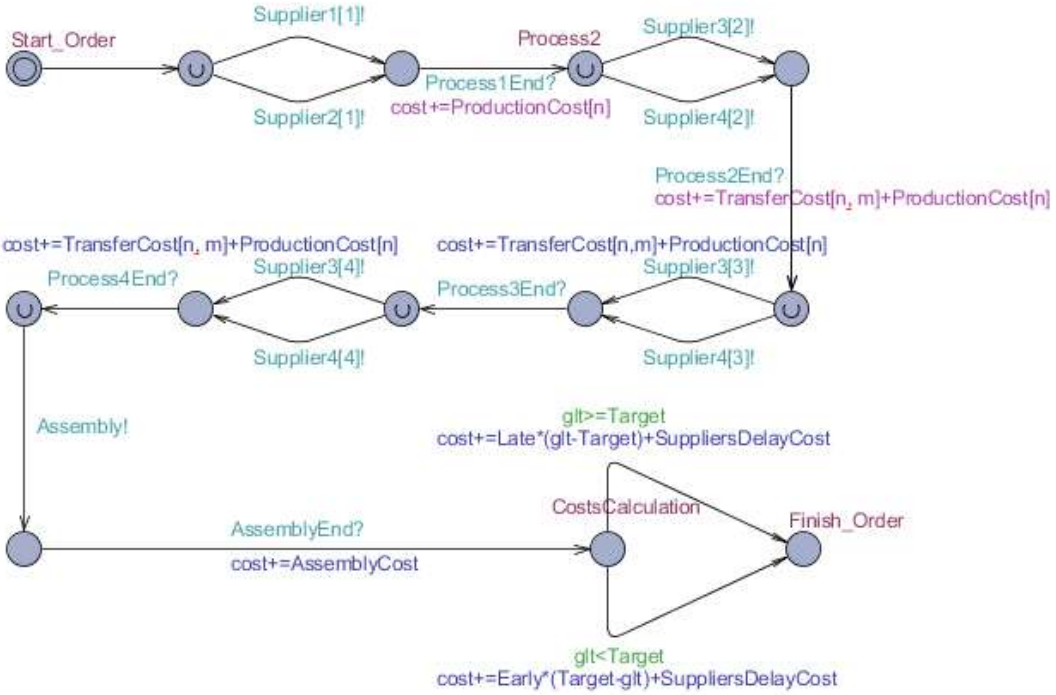
Για τη συγκεκριμένη παραγγελία η εταιρεία ζήτησε από τα τμήματα μελέτης και σχεδιασμού των προμηθευτών της να αποστείλουν τους χρόνους ανά φάση κατεργασίας καθώς και το αντίστοιχο κόστος παραγωγής, λαμβάνοντας τις ακόλουθες απαντήσεις / προσφορές:

Φάση (j)	Προμηθευτής (k)	Ημέρες Παράδοσης (t_{jk})	Κόστος (c_{jk})
1	1	10	1800
1	2	3	2112
2	3	2	1024
2	4	5	1040
3	3	2	1012
3	4	4	1356
4	3	7	7532
4	4	10	8000
Συναρμολόγηση λεβήτων από τον κατασκευαστή		5	10000

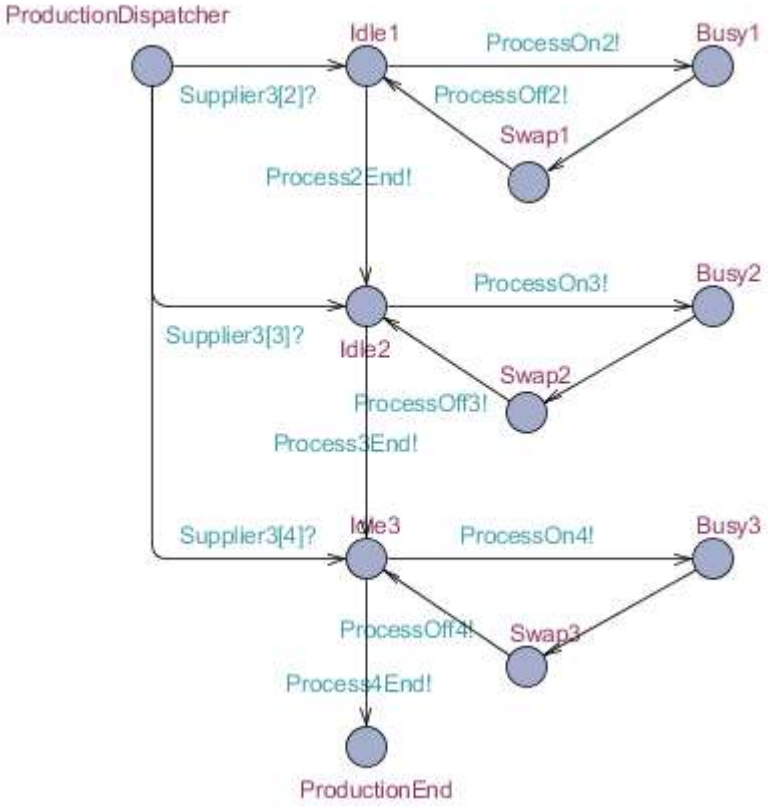
Το πρόβλημα που τίθεται είναι η επιλογή της βέλτιστης ανάθεσης ώστε η παραγγελία να εκτελεστεί με το ελάχιστο δυνατό κόστος. Αυτό συμπεριλαμβάνει, εκτός του κόστους παραγωγής που δίνεται στον παραπάνω πίνακα, τα κόστη μεταφορών μεταξύ των προμηθευτών καθώς και πιθανό κόστος αποθήκευσης ή καθυστέρησης εφόσον η παράδοση του σώματος του λέβητα στην εταιρεία γίνει σε χρόνο διαφορετικό των 20 ημερών που είναι ο επιδιωκόμενος χρόνος αν συνυπολογιστούν οι 5 μέρες που απαιτούνται (χωρίς ελαστικότητα) προκειμένου να συναρμολογηθούν οι λέβητες και να αποσταλούν στη συνέχεια στον πελάτη.

Η μοντελοποίηση του προβλήματος βάσει της προτεινόμενης προσέγγισης με χρονισμένα αυτόματα με κόστη παρουσιάζεται στη συνέχεια. Η δημιουργία του μοντέλου της παραγγελίας είναι ιδιαίτερα απλή υπόθεση η οποία μπορεί να γίνει με πλήρως αυτοματοποιημένο τρόπο εφόσον καθοριστεί το ποιοι είναι οι εναλλακτικοί προμηθευτές / υπεργολάβοι για κάθε μία από τις διεργασίες (processes) 1 ως 4.

Σε ότι αφορά τις μοντελοποιήσεις των προμηθευτών, καθώς οι προμηθευτές Π1 και Π2 , όπως και οι Π3 και Π4 έχουν μεταξύ τους κοινή δομή, τα αυτόματα που τους αναπαριστούν είναι ίδια, με διαφορετικές φυσικά τιμές παραμέτρων για τους χρόνους, τα κόστη κλπ. Ενδεικτικά παρουσιάζεται στη συνέχεια το αυτόματο του προμηθευτή Π3 όπως και το γενικό αυτόματο της παραγγελίας.



Σχήμα 65: Αυτόματο Παραγγελίας



Σχήμα 66: Αυτόματο Προμηθευτή

Ολοκληρώνοντας τη μοντελοποίηση του προβλήματος και αφού ορίστηκαν όλες οι παράμετροι και περάστηκαν στο μοντέλο τα κόστη και οι χρόνοι όπως παρουσιάστηκαν ανωτέρω, το ολοκληρωμένο σύστημα αποθηκεύτηκε και προωθήθηκε στη μηχανή ελέγχου και βελτιστοποίησης μοντέλων.

Η βέλτιστη λύση που εντοπίστηκε είχε συνολικό κόστος:

$C(\lambda_i)$ = Cost' = 12.546 €

Τα επιμέρους κόστη που συναθροίζουν το παραπάνω είναι τα ακόλουθα

$C_p(\lambda_i)$ = 11.696 €, το οποίο εκφράζει το καθαρό κόστος παραγωγής

$C_s(\lambda_i)$ = 850 €, το οποίο εκφράζει τα κόστη μεταφορών

$C_t(\lambda_i)$= 0 €, μηδενικό κόστος λόγω του γεγονότος ότι με το βέλτιστο χρονοπρόγραμμα η παρτίδα ολοκληρώνεται ακριβώς στο ζητούμενο χρόνο.

Τα βασικά transitions τα οποία αντιστοιχούν στο βέλτιστο χρονοπρόγραμμα είναι τα ακόλουθα:

Start_Order	→	Supplier2[1]
Process2	→	Supplier4[2]
Process3	→	Supplier3[3]
Process4	→	Supplier3[4]
Assembly	→	Finish_Order

Βάσει των παραπάνω, για την παραγγελία αυτή το βέλτιστο χρονοπρόγραμμα έχει ως εξής:

- Η επεξεργασία της παραγγελίας ξεκινάει από τον προμηθευτή Π2 όπου γίνεται η χύτευση του υλικού και παράγονται τα στοιχεία του σώματος του λέβητα.
- Μετά την ολοκλήρωση της χύτευσης τα στοιχεία μεταφέρονται στον προμηθευτή/υπεργολάβο Π4 για μηχανουργική κατεργασία.
- Στη συνέχεια μεταφέρονται στον προμηθευτή/υπεργολάβο Π3 στον οποίο πραγματοποιείται το μοντάρισμα καθώς και ο έλεγχος.
- Τέλος τα σώματα των λεβήτων μεταφέρονται στον τελικό κατασκευαστή για τη συναρμολόγησή τους και στη συνέχεια παραδίδονται στον πελάτη ακριβώς στο συμφωνηθέντα χρόνο.

Η εφαρμογή που υλοποιήθηκε πάνω στο πρόβλημα που αναλύθηκε προηγούμενα, τα στοιχεία του οποίου παραχωρήθηκαν από την επιχείρηση ΚΑΖΗΣ ΑΒΕΕ, κατέδειξε την αποτελεσματικότητα της προτεινόμενης προσέγγισης καθώς δεν απαιτήθηκαν παρά ελάχιστα λεπτά τόσο για τη μοντελοποίηση του προβλήματος όσο και στη συνέχεια για την επίλυσή του (ο χρόνος υπολογισμού της βέλτιστης λύσης ήταν 0,7 sec). Φυσικά πρέπει να σημειωθεί ότι η συγκεκριμένη εφαρμογή μπορεί να θεωρηθεί χαμηλής, αναλογικά, πολυπλοκότητας κυρίως λόγω του γεγονότος ότι οι χρόνοι παράδοσης που αξιοποιήθηκαν για κάθε προμηθευτή δεν ελέγχθηκαν έναντι του προγράμματος παραγωγής του για πρόκληση πιθανών καθυστερήσεων σε άλλες παραγγελίες του και άρα προσθήκη σχετικού κόστους. Τέτοιος έλεγχος δεν ήταν εφικτός λόγω έλλειψης στοιχείων τα οποία δεν παραχωρήθηκαν από τους προμηθευτές (καθώς η συνεργασία τους με την εταιρεία δεν προβλέπει σχετική διάθεση στοιχείων παραγωγής και παραγγελιών τρίτων). Προκειμένου να επιβεβαιωθεί η λειτουργία του μοντέλου διεξήχθησαν έλεγχοι με βάση υποθετικά σενάρια για τις παραγγελίες των διαφόρων προμηθευτών. Η πιο ενδιαφέρουσα των περιπτώσεων που υποθετικά ελέγχθηκαν ήταν αυτή κατά την οποία ο προμηθευτής 4 προκειμένου να ολοκληρώσει εντός του προγραμματισμένου χρόνου τη Διεργασία 2 (Κατεργασία) οδηγήθηκε σε καθυστέρηση μιας θεωρούμενης παραγγελίας του με κόστος καθυστέρησης μόλις 300 €.

Η παραπάνω υπόθεση άλλαξε ουσιαστικά το τελικό αποτέλεσμα του προβλήματος σε επίπεδο ανάθεσης εργασιών όσο και σε επίπεδο συνολικού κόστους. Συγκεκριμένα το νέο κόστος που προέκυψε στο σενάριο αυτό ήταν:

$C'(\lambda_i)$ = Cost' = 12.780 €

Τα επιμέρους κόστη που συναθροίζουν το παραπάνω είναι τα ακόλουθα

$C'_p(\lambda_i)$ = 11.680 €, το οποίο εκφράζει το καθαρό κόστος παραγωγής

$C'_s(\lambda_i)$ = 350 €, το οποίο εκφράζει τα κόστη μεταφορών

$C'_t(\lambda_i)$= 750 €, το οποίο εκφράζει κόστος αποθήκευσης καθώς η παραγγελία ετοιμάστηκε 5 ημέρες νωρίτερα από το ζητούμενο χρόνο.

Τα βασικά transitions τα οποία αντιστοιχούν στο νέο (υποθετικό) βέλτιστο χρονοπρόγραμμα είναι τα ακόλουθα:

Start_Order → Supplier2[1]

Process2	→	Supplier3[2]
Process3	→	Supplier3[3]
Process4	→	Supplier3[4]
Assembly	→	Finish_Order

Βάσει των παραπάνω προκύπτει ότι το μικρό αναλογικά κόστος που ο προμηθευτής Π4 «χρέωσε» στο παραγωγικό δίκτυο οδήγησε στο να μην επιλεγεί τελικά για καμία διεργασία σε ότι αφορά στη συγκεκριμένη παραγγελία. Παράλληλα, δε, η παραγγελία ετοιμάστηκε σε χρόνο ταχύτερο από τον προγραμματιζόμενο. Κάτι τέτοιο, εφόσον ενώ γενικά θα μπορούσε να θεωρηθεί ως θετικό στοιχείο, στη συγκεκριμένη περίπτωση προσμετρήθηκε αρνητικά καθώς οδήγησε σε πρόσθετα κόστη αποθήκευσης.

Σε κάθε περίπτωση το συμπέρασμα το οποίο προέκυψε από την ανάλυση αλλά και από την εφαρμογή που έγινε τόσο σε δοκιμαστικό επίπεδο όσο και σε πραγματικό πρόβλημα παραγωγής είναι ότι η προτεινόμενη μέθοδος έχει ιδιαίτερα πλεονεκτήματα τα οποία εντοπίζονται στην ευκολία με την οποία μοντελοποιείται κάθε τύπος παραγγελίας και κάθε είδος παραγωγικής μονάδας, ενώ επιτρέπει τη βελτιστοποίηση βάσει σύνθετων συναρτήσεων κόστους.

Κεφάλαιο 10: Συμπεράσματα – προοπτικές

10.1. Σύνοψη διατριβής

Η παρούσα διατριβή επικεντρώθηκε στην αξιοποίηση των υπολογιστικών δυνατοτήτων των χρονισμένων αυτομάτων για τη μοντελοποίηση και επίλυση σύνθετων προβλημάτων χρονικού προγραμματισμού εργασιών. Στο πλαίσιο αυτό αναλύθηκαν οι διεθνείς εξελίξεις στην μοντελοποίηση συστημάτων με χρήση αυτομάτων πεπερασμένων καταστάσεων και εντοπίσθηκαν περιοχές οι οποίες μέχρι σήμερα δεν είχαν καλυφθεί από τις ερευνητικές προσπάθειες στα επιστημονικά πεδία του προγραμματισμού παραγωγής και των πεπερασμένων αυτομάτων. Με βάση την ανάλυση που πραγματοποιήθηκε, η διατριβή κινήθηκε στις εξής κατευθύνσεις:

- Παρουσιάστηκε μια ολοκληρωμένη προσέγγιση για τη μοντελοποίηση προβλημάτων χρονικού προγραμματισμού εργασιών η οποία βασίζεται στην απεικόνιση των διακριτών καταστάσεων στις οποίες περιέρχεται ένα παραγωγικό σύστημα κατά την εκτέλεση των προγραμματισμένων διεργασιών του.
- Εντοπίστηκαν και προτάθηκαν προσεγγίσεις για την βέλτιστη επίλυση συνήθων προβλημάτων χρονοπρογραμματισμού με χρήση των προτεινόμενων μοντέλων χρονισμένων αυτομάτων, αξιοποιώντας

αλγορίθμους προσπελασιμότητας. Στη συνέχεια και προκειμένου να μειωθεί ο υπολογιστικός χρόνος για την αντιμετώπιση προβλημάτων μεγάλων διαστάσεων προτάθηκαν βελτιωμένα μοντέλα αυτομάτων με όρια τα οποία οδηγούν σε περιορισμό του χώρου των δυνατών λύσεων, βελτιώνοντας έτσι την αποτελεσματικότητα των αλγορίθμων επίλυσης.

- Η προσέγγιση της μοντελοποίησης και επίλυσης με χρονισμένα αυτόματα προσαρμόστηκε στη συνέχεια προκειμένου να μπορεί να εφαρμοστεί σε προβλήματα χρονοπρογραμματισμού διακοπτόμενων εργασιών, κατά τα οποία επιτρέπεται η διακοπή μιας εργασίας πριν την ολοκλήρωσή της προκειμένου να επεξεργασθεί κατά προτεραιότητα η συγκεκριμένη μηχανή κάποια άλλη εργασία και στη συνέχεια να ολοκληρώσει την αρχική.
- Τα μοντέλα χρονισμένων αυτομάτων που αναπαριστούν τις παραγωγικές διεργασίες αναβαθμίστηκαν με την προσθήκη στοιχείων κόστους τόσο στις μεταβάσεις όσο και στις διακριτές καταστάσεις. Με τον τρόπο αυτό έγινε εφικτή η αντιμετώπιση προβλημάτων στα οποία ο στόχος βελτιστοποίησης δεν είναι συνάρτηση αποκλειστικά του χρόνου αλλά λαμβάνονται υπόψη στοιχεία και περιορισμοί που σχετίζονται με τα κόστη παραγωγής.
- Αναπτύχθηκαν μοντέλα για την αναπαράσταση διαφόρων τύπων κόστους που υπεισέρχονται στην παραγωγική διαδικασία, όπως του κόστους επεξεργασίας, του κόστους μεταφοράς, του κόστους αποθήκευσης και του κόστους καθυστερημένης παράδοσης.
- Με άξονα τα παραπάνω μοντέλα χρονισμένων αυτομάτων με κόστη αναπτύχθηκε μια ολοκληρωμένη διαδικασία μοντελοποίησης και επίλυσης προβλημάτων ανεξάρτητη της δομής του παραγωγικού συστήματος και των περιορισμών που τη διέπουν. Έτσι κατέστη εφικτή η επίλυση προβλημάτων με παράλληλες μηχανές ή γραμμές παραγωγής όπως και με εναλλακτικές διαδρομές για τις διεργασίες, λαμβάνοντας παράλληλα υπόψη όλα τα κόστη τα οποία υπεισέρχονται και βελτιστοποιώντας είτε ως προς το χρόνο είτε ως προς συγκεκριμένα στοιχεία κόστους ή/και το συνολικό κόστος για το παραγωγικό σύστημα.
- Αναλύθηκε το πρόβλημα του χρονικού προγραμματισμού σε δυναμικά δίκτυα παραγωγής με έμφαση σε δίκτυα τα οποία δημιουργούνται με άξονα μεγάλες παραγωγικές εταιρείες – τελικούς κατασκευαστές και περιλαμβάνουν τους

βασικούς προμηθευτές και συνεργάτες τους (κατά βάση ΜΜΕ). Εντοπίσθηκαν τα προβλήματα συντονισμού μεταξύ των παραγωγικών μονάδων των επιχειρήσεων που συμμετέχουν σε ένα δυναμικό δίκτυο παραγωγής καθώς και τα κόστη που πρέπει να ληφθούν υπόψη προκειμένου να βελτιστοποιηθεί η λειτουργία του όλου δικτύου.

- Με βάση την ανάλυση των στοιχείων των δυναμικών δικτύων παραγωγής και αξιοποιώντας τα μοντέλα χρονικού προγραμματισμού εργασιών που αναπτύχθηκαν στη διατριβή, προτάθηκε μια ολοκληρωμένη προσέγγιση για τον προγραμματισμό της ανάθεσης εργασιών και της παραγωγής σε επίπεδο παραγωγικού δικτύου.
- Ιδιαίτερη έμφαση δόθηκε στη δεδομένη επιφύλαξη ή/και απροθυμία κάθε επιχείρησης να διαθέσει αναλυτικά στοιχεία για την παραγωγή, τους πόρους και τις παραγγελίες της στα υπόλοιπα μέλη του παραγωγικού δικτύου. Προκειμένου να αντιμετωπιστεί το ζήτημα αυτό, προτάθηκε μια κατανεμημένη προσέγγιση βάσει της οποίας το κάθε μέλος του παραγωγικού δικτύου αποτυπώνει αυτόνομα, με χρήση των μοντέλων αυτομάτων που προτείνονται στη διατριβή, τους πόρους, τα κόστη και τις διεργασίες του και παρέχει πρόσβαση στα στοιχεία αυτά μόνο στη μηχανή βελτιστοποίησης του δικτύου, αποφεύγοντας έτσι πιθανή διαρροή κρίσιμων παραγωγικών δεδομένων.
- Μοντελοποιήθηκε μια σειρά διαφορετικών στοιχείων κόστους τα οποία υπεισέρχονται σε ένα παραγωγικό δίκτυο, ενώ δόθηκε έμφαση στην αντιμετώπιση της συνηθισμένης όσο και σύνθετης περίπτωσης κατά την οποία η βελτιστοποίηση της ροής εργασιών σε επίπεδο δικτύου παραγωγής προκαλεί πρόσθετο κόστος σε κάποιο μέλος αυτού παρά το γεγονός ότι το συνολικό κόστος για το σύνολο του δικτύου μειώνεται.
- Η προτεινόμενη προσέγγιση εφαρμόστηκε με απόλυτη επιτυχία σε μια πραγματική περίπτωση παραγωγικού δικτύου βάσει στοιχείων τα οποία διατέθηκαν από βιομηχανική επιχείρηση κατασκευής λεβήτων θέρμανσης η οποία δραστηριοποιείται στον ελληνικό χώρο. Η επιχείρηση παρά το γεγονός ότι δεν έχει δημιουργήσει συστήσει επίσημα κάποιο δυναμικό δίκτυο παραγωγής, διαθέτει μια σειρά προμηθευτών και υπεργολάβων που λειτουργούν με άξονα αυτήν και οι σχέσεις συνεργασίας τους προσομοιώνουν σε μεγάλο βαθμό ένα δομημένο παραγωγικό δίκτυο. Στο πλαίσιο της

εφαρμογής εξετάστηκε ένα χαρακτηριστικό πραγματικό πρόβλημα τα στοιχεία του οποίου παραχωρήθηκαν από την επιχείρηση και εντοπίστηκε η βέλτιστη κατανομή εργασιών με άξονες το κόστος αλλά και το συμφωνηθέντα χρόνο παράδοσης στον τελικό πελάτη.

10.2. Συμπεράσματα

Η παρούσα διατριβή κατέδειξε τις δυνατότητες που παρέχουν τα αυτόματα πεπερασμένων καταστάσεων για τη μοντελοποίηση και επίλυση σύνθετων προβλημάτων χρονικού προγραμματισμού εργασιών, τόσο σε επίπεδο αυτόνομης παραγωγικής μονάδας όσο και σε επίπεδο δικτύου παραγωγής. Ξεκινώντας από το βασικό μοντέλο αυτομάτων με χρόνους (χρονισμένων αυτομάτων) το οποίο εντοπίστηκε στη βιβλιογραφία και επιτρέπει την αναπαράσταση βασικών προβλημάτων δρομολόγησης εργασιών, αναπτύχθηκε, στο πλαίσιο της διατριβής, μια ολοκληρωμένη καινοτόμος προσέγγιση αντιμετώπισης σύνθετων παραγωγικών προβλημάτων. Η προτεινόμενη προσέγγιση διαθέτει βασικά στοιχεία καινοτομίας τα οποία αφορούν:

- **Στην ανάπτυξη μιας ολοκληρωμένης διαδικασίας μοντελοποίησης για την ακριβή αναπαράσταση σύνθετων παραγωγικών δομών εκτός των πρότυπων μορφών.**

  Η πλειοψηφία των μεθόδων επίλυσης προβλημάτων χρονοπρογραμματισμού επικεντρώνεται σε συγκεκριμένες παραγωγικές δομές (όπως job-shop, batch-shop, flow-shop κλπ), προσπαθώντας να αποτυπώσει μέσω αυτών κάθε παραγωγικό σύστημα, κάτι που περιορίζει την εφαρμογή τους σε πραγματικά προβλήματα, δεδομένου ότι οι σύγχρονες παραγωγικές μονάδες χαρακτηρίζονται από πιο σύνθετες όσο και ευέλικτες δομές. Η προτεινόμενη στη διατριβή διαδικασία μοντελοποίησης, αντίθετα, επιτρέπει κάθε παραγωγική δομή να παρασταθεί ακριβώς ως έχει χωρίς να απαιτείται οποιουδήποτε τύπου παραδοχή ή απλοποίηση. Η δυνατότητα δε της γραφικής αναπαράστασης του κάθε παραγωγικού συστήματος παρέχει ένα ιδιαίτερο πλεονέκτημα καθώς επιτρέπει τη μοντελοποίηση όσο και των έλεγχο της ορθότητας αυτής χωρίς να τίθεται ως προϋπόθεση η γνώση στοιχείων της θεωρίας αυτομάτων στην οποία βασίζεται η προτεινόμενη προσέγγιση.

- **Στην ανάπτυξη τεχνικών και μοντέλων που επιτρέπουν το μετασχηματισμό κάθε προβλήματος χρονοπρογραμματισμού, ανεξαρτήτως περιορισμών, σε πρόβλημα εξεύρεσης βέλτιστης διαδρομής σε δίκτυα χρονισμένων αυτομάτων.**

  Στη διατριβή αναπτύσσονται μοντέλα τα οποία μπορούν να αναπαραστήσουν κάθε διεργασία που εισέρχεται σε ένα παραγωγικό σύστημα λαμβάνοντας υπόψη τους περιορισμούς που τη συνοδεύουν. Έτσι επιτρέπεται, χωρίς μεγάλη διαφοροποίηση σε επίπεδο τεχνικής μοντελοποίησης, η αντιμετώπιση περιπτώσεων διαφορετικών χαρακτηριστικών όπως περιπτώσεων συνεχούς ροϊκής παραγωγής, διακριτής παραγωγής, παραγωγής σε παρτίδες και παραγωγής με εναλλακτικά φασεολόγια.

- **Στη δυνατότητα μοντελοποίησης όλων των στοιχείων κόστους που υπεισέρχονται στην παραγωγική διαδικασία προκειμένου να ληφθούν υπόψη κατά την εύρεση βέλτιστων χρονοπρογραμμάτων.**

  Την ώρα που η πλειοψηφία των μεθόδων χρονικού προγραμματισμού εργασιών επικεντρώνεται στο στοιχείο του χρόνου και της βελτιστοποίησης παραμέτρων που σχετίζονται με αυτό (πχ χρόνων ολοκλήρωσης, αναμονής, προετοιμασίας κλπ), η προτεινόμενη προσέγγιση δίνει κύρια έμφαση στο ζήτημα το κόστους. Επιτρέπει, με σχετικά απλή μοντελοποίηση, την αποτύπωση ακόμα και σύνθετων στοιχείων κόστους όπως για παράδειγμα του κόστους εκκίνησης και επανεκκίνησης μιας γραμμής παραγωγής, του κόστους ενδιάμεσης αποθήκευσης ημι-έτοιμων προϊόντων και του κόστους μεταφοράς μεταξύ των παραγωγικών πόρων σε περίπτωση διεσπαρμένης γεωγραφικά παραγωγής.

- **Στην επίλυση των μοντελοποιημένων προβλημάτων χρονοπρογραμματισμού ως προς διαφορετικούς παράγοντες/ κριτήρια βελτιστοποίησης χωρίς να απαιτείται οποιαδήποτε περαιτέρω προσαρμογή του μοντέλου.**

  Η προσέγγιση που προτείνεται στη διατριβή μετατρέπει κάθε κριτήριο βελτιστοποίησης σε ένα παράγοντα κόστους. Κάθε χρονικό στοιχείο που υπεισέρχεται σε κάποιο πρόβλημα προσεγγίζεται με ένα αυτόνομο χρονιστή (ρολόι) ο οποίος στη συνέχεια το μετατρέπει σε ένα τύπο κόστους. Από την άλλη τα επιμέρους στοιχεία κόστους μπορούν να συντεθούν τόσο επιλεκτικά

όσο και συνολικά σε συναρτήσεις συνδυαστικού κόστους (είτε γραμμικές είτε όχι) και η βελτιστοποίηση γίνεται με βάση αυτό. Έτσι μπορεί να γίνει βελτιστοποίηση τόσο βάσει ενός απλού κριτηρίου όπως πχ η ελαχιστοποίηση του χρόνου χαλάρωσης, μετατρέποντας το χρόνο χαλάρωσης σε κόστος με αντιστοιχία 1 χρονική μονάδα = 1 μονάδα κόστους, όσο και βάσει σύνθετων κριτηρίων που συνδυάζουν πολλές πηγές κόστους. Στο πλαίσιο της διατριβής εξετάστηκαν τόσο απλές όσο και σύνθετες περιπτώσεις, ενώ ως ιδιαίτερα περίπλοκη περίπτωση αντιμετωπίστηκε επιτυχώς και η περίπτωση των δυναμικών δικτύων παραγωγής στα οποία τα κόστη δε βαραίνουν μόνο συνολικά το δίκτυο αλλά και ξεχωριστά τις συμμετέχουσες επιχειρήσεις.

- **Στο βέλτιστο προγραμματισμό εργασιών για Δυναμικά Δίκτυα Παραγωγής λαμβάνοντας υπόψη τα κόστη με τα οποία επιβαρύνονται τα μέλη τους όσο και την προστασία των παραγωγικών τους δεδομένων.**

  Το συγκεκριμένο εξαγόμενο της διατριβής ενέχει σημαντική καινοτομία και ως εκ τούτου ιδιαίτερη αξία στην πράξη. Εξετάζοντας την περίπτωση των δυναμικών παραγωγικών δικτύων τόσο στην πράξη όσο και σε ερευνητικό επίπεδο, απουσιάζουν ολοκληρωμένες προσεγγίσεις που να επιτρέπουν το συντονισμό των διεργασιών σε επίπεδο παραγωγικού δικτύου, συμπεριλαμβανομένης τόσο της ανάθεσης εργασιών όσο και του επιμέρους χρονοπρογραμματισμού της παραγωγής. Η προτεινόμενη στη διατριβή προσέγγιση όχι μόνο αντιμετωπίζει επιτυχώς το παραπάνω πρόβλημα αλλά επιπρόσθετα λαμβάνει υπόψη την ανάγκη προστασίας των παραγωγικών δεδομένων της κάθε συμμετέχουσας επιχείρησης, μέσω μιας αποκεντρωμένης προσέγγισης. Συγκεκριμένα, παρέχοντας τη δυνατότητα στο κάθε μέλος του παραγωγικού δικτύου να μοντελοποιεί αυτόνομα την παραγωγή του και να επικαιροποιεί σε σταθερή βάση τα σχετικά δεδομένα της, υπερσκελίζεται το πρόβλημα της άρνησης κοινής χρήσης στοιχείων που είναι απαραίτητα για να γίνει κεντρικός προγραμματισμός της παραγωγής. Χωρίς δηλαδή να διατίθεται τα στοιχεία του κάθε μέλους του δικτύου σε οποιονδήποτε (συμπεριλαμβανομένου και της επιχείρησης που το διαχειρίζεται), παρέχεται πρόσβαση σε αυτά μόνο στη μηχανή βελτιστοποίησης η οποία συγχρονίζει τα αυτόματα των επιμέρους προμηθευτών και στη συνέχεια παράγει χρονοπρογράμματα

βελτιστοποιημένα ως προς το κόστος (ή και το χρόνο δεδομένου ότι στην πράξη κάθε χρονική διάσταση μπορεί να εκφραστεί μέσα από ένα είτε πραγματικό είτε θεωρητικό κόστος). Σε ό,τι αφορά στα Δυναμικά Δίκτυα Παραγωγής δίνεται έμφαση και σε ένα άλλο ιδιαίτερα σημαντικό στοιχείο που σχετίζεται με το γεγονός ότι η βελτιστοποίηση της λειτουργίας του δικτύου δε συνεπάγεται σε καμία περίπτωση βελτιστοποίηση της λειτουργίας των επιμέρους μελών αυτού. Ενδέχεται δηλαδή προκειμένου να βελτιστοποιηθεί η παραγωγή που αφορά μιας υπο-εκτέλεση παραγγελίας του δικτύου να απαιτηθεί ορισμένα μέλη του δικτύου να καθυστερήσουν δικές τους εξωτερικές παραγγελίες και άρα να επωμιστούν σχετικά κόστη. Το ζήτημα αυτό αντιμετωπίζεται μέσω του σημαντικού πλεονεκτήματος που παρέχει η προτεινόμενη προσέγγιση να συνδυάζει την ανάθεση εργασιών σε προμηθευτές με τον επιμέρους χρονοπρογραμματισμό της παραγωγής τους. Έτσι το σύστημα δίνει τη δυνατότητα υπολογισμού του πρόσθετου κόστους που προκύπτει σε κάθε μέλος του δικτύου λόγω της «αναστάτωσης» του προγράμματος παραγωγής του ως αποτέλεσμα της δικτυακής βελτιστοποίησης και μεταθέτει το κόστος αυτό στο ίδιο το δίκτυο. Εφόσον η χρησιμοποίηση ενός προμηθευτή που έχει ένα τέτοιο πρόσθετο κόστος είναι και πάλι προς συμφέρον του δικτύου (έχοντας συνολικό όφελος μεγαλύτερο από το επιμέρους κόστος του προμηθευτή) τότε το ίδιο το δίκτυο επωμίζεται το πρόσθετο αυτό κόστος χωρίς να προκαλείται ζημιά στον προμηθευτή. Αντίθετα, εφόσον η μεταφορά του κόστους προς το δίκτυο υπερσκελίζει το όφελος από τη χρησιμοποίηση του συγκεκριμένου προμηθευτή τότε αυτόματα η μηχανή βελτιστοποίησης επιλέγει το επόμενο εναλλακτικό χρονοπρόγραμμα και την αντίστοιχη ανάθεση εργασιών μέχρι να βρεθεί η πλέον συμφέρουσα λύση.

Εξετάζοντας στο σύνολό της την προσέγγιση που προτείνεται στην παρούσα διατριβή, σε συνδυασμό με τις σύγχρονες τάσεις στον τομέα της παραγωγής που αξιοποιούν συστήματα αυτομάτων για την αντιμετώπιση διαφόρων ζητημάτων βελτιστοποίησης ή/και αυτομάτου ελέγχου, προκύπτει το συμπέρασμα ότι η πρόταση της διατριβής έχει σημαντική αξία όχι μόνο σε ερευνητικό αλλά και σε πρακτικό επίπεδο. Η συνεχής εξέλιξη των εργαλείων μοντελοποίησης αυτομάτων, σε συνδυασμό με το γεγονός ότι προσεγγίσεις όπως η προτεινόμενη συνδυάζουν την αποτελεσματικότητα και τη σχετική ευκολία μοντελοποίησης με την αυξημένη ευελιξία

η οποία χαρακτηρίζει τις σύγχρονες παραγωγικές μονάδες, παρέχουν τα εχέγγυα για περαιτέρω αξιοποίηση των ερευνητικών αποτελεσμάτων και συμπερασμάτων της εργασίας στην κατεύθυνση της δημιουργίας ενός ολοκληρωμένου πληροφοριακού συστήματος το οποίο θα συνδυάζει τις δυνατότητες της προσέγγισης με τα χαρακτηριστικά που απαιτείται να διαθέτει ένα εμπορικό σύστημα προγραμματισμού προκειμένου να μπορεί να αξιοποιηθεί ως αξιόπιστη επιχειρηματική λύση.

10.3. Προοπτικές

Η προσέγγιση που προτάθηκε στην παρούσα διατριβή διαθέτει σημαντική ερευνητική αξία καθώς αναδεικνύει και πιστοποιεί τις δυνατότητες των πεπερασμένων αυτομάτων στην κατεύθυνση της αντιμετώπισης σύνθετων προβλημάτων χρονικού προγραμματισμού εργασιών. Δεδομένου ότι η ανάπτυξη αντιστοίχων προσεγγίσεων αποτελεί σύγχρονο αντικείμενο έρευνας, το οποίο αναπτύχθηκε κυρίως μέσα στην τελευταία δεκαετία, η παρούσα διατριβή συμβάλει στο πεδίο αυτό σημαντικά, ενώ παράλληλα δημιουργεί προοπτικές για περαιτέρω θεωρητική αλλά και εμπορική ανάπτυξη. Στη συνέχεια αναλύονται ορισμένες προοπτικές οι οποίες εντοπίστηκαν κατά τη διάρκεια εκπόνησης της διατριβής ως αντικείμενα περαιτέρω ανάπτυξης:

- **Ενσωμάτωση εξειδικευμένων αλγορίθμων/ τεχνικών βελτιστοποίησης ανά τύπο προβλήματος**

  Η προτεινόμενη στη διατριβή προσέγγιση εστιάζει ως επί το πλείστον στη διαδικασία μοντελοποίησης σύνθετων παραγωγικών προβλημάτων στο πλαίσιο της αντιμετώπισης προβλημάτων βελτιστοποίησης ως προς διάφορες αντικειμενικές συναρτήσεις. Σε ότι αφορά στον αλγόριθμο τον οποίο χρησιμοποιεί η μηχανή βελτιστοποίησης, αυτός εντάσσεται στην οικογένεια των αλγορίθμων διακλάδωσης και οριοθέτησης (branch & bound). Ο αλγόριθμος έχει το πλεονέκτημα ότι μπορεί να αντιμετωπίσει οποιοδήποτε πρόβλημα μοντελοποιηθεί βάσει της προτεινόμενης προσέγγισης, αλλά όπως όλοι οι αλγόριθμοι οι οποίοι δεν είναι απόλυτα εξειδικευμένοι πάνω σε ένα συγκεκριμένο τύπο προβλήματος δεν έχει βέλτιστη αποδοτικότητα (μετρούμενη σε απαιτούμενο υπολογιστικό χρόνο επίλυσης). Στο πλαίσιο αυτό ανοίγεται η ερευνητική προοπτική της ανάπτυξης μιας σειράς εναλλακτικών αλγορίθμων που θα εκτελούνται πάνω στα μοντέλα αυτομάτων και αναλύοντας τη δομή του κάθε μοντέλου θα ρυθμίζουν τις παραμέτρους

βελτιστοποίησης (πχ τα Lower & Upper Bounds). Δεδομένου του πλήθους των εναλλακτικών παραγωγικών προβλημάτων πρόκειται για ιδιαίτερα πολύπλοκο πρόβλημα που χρήζει περαιτέρω ερευνητικής διερεύνησης.

- **Ενσωμάτωση στοιχείων θεωρίας παιγνίων για την αντιμετώπιση των «συγκρούσεων» στην επίλυση προβλημάτων δυναμικών δικτύων παραγωγής.**

  Η προτεινόμενη προσέγγιση για το χρονοπρογραμματισμό δυναμικών δικτύων παραγωγής αντιμετωπίζει με επιτυχία τις κατηγορίες δικτύων στις οποίες δε τίθεται ζήτημα διαμοιρασμού του κόστους μεταξύ των συμμετεχουσών επιχειρήσεων. Υπάρχουν εν τούτοις περιπτώσεις που η βελτιστοποίηση των στοιχείων του δικτύου έρχεται σε πλήρη σύγκρουση με τη βελτιστοποίηση των στοιχείων των επιμέρους μελών του. Στις περιπτώσεις αυτές επέρχεται μία σύγκρουση συμφερόντων η επίλυση της οποίας ανάγεται στο ερευνητικό πεδίο της θεωρίας παιγνίων. Η ιδέα είναι ότι η επίλυση της κάθε σύγκρουσης θα πρέπει τελικά να οδηγεί σε μια συνολική ανακατανομή του κόστους ή του κέρδους από μια παραγωγική διαδικασία μεταξύ των μελών του δικτύου. Πρόκειται για ένα ζήτημα το οποίο σε ερευνητικό επίπεδο έχει ιδιαίτερο ενδιαφέρον κατά την αντιμετώπιση προβλημάτων παραγωγικών δικτύων και το οποίο η παρούσα διατριβή αναδεικνύει και ενισχύει καθώς παρέχει την πλατφόρμα για πλήρη ανάλυση των εμπλεκόμενων στοιχείων κόστους.

- **Ενσωμάτωση στοιχείων πολυκριτηριακής ανάλυσης για την επιλογή και σύνθεση των κριτηρίων βελτιστοποίησης κατά περίπτωση.**

  Η προτεινόμενη προσέγγιση έχει τη δυνατότητα βελτιστοποίησης βάσει διαφορετικών κριτηρίων ανάλογα με τις επιλογές του χρήστη κατά τη διαδικασία επίλυσης. Σημαντική προοπτική για την ολοκλήρωση της της προσέγγισης θα ήταν η ενσωμάτωση μιας πολυκριτηριακής θεώρησης στη σύνθεση των κριτηρίων βελτιστοποίησης, ανάλογα με τα ειδικά δεδομένα κάθε προβλήματος και τις επιλογές του χρονοπρογραμματιστή/ χρήστη του συστήματος. Η προοπτική αυτή έχει τόσο ερευνητική διάσταση – καθώς απαιτεί σχετική ερευνητική προσπάθεια για την ανάπτυξη ενός προσαρμοσμένου πολυκριτηριακού μοντέλου - όσο και εμπορική, καθώς θα

μπορούσε να ενσωματωθεί σε ένα σύστημα λήψης απόφασης που θα βασίζεται στην προτεινόμενη στη διατριβή προσέγγιση.

- **Ανάπτυξη πληροφοριακού συστήματος χρονοπρογραμματισμού εργασιών βασισμένο στην προτεινόμενη προσέγγιση.**

  Παρά το γεγονός ότι το εργαλείο σχεδίασης των μοντέλων αυτομάτων είναι ιδιαίτερα φιλικό προς το χρήστη, εν τούτοις η μοντελοποίηση συνολικά ενός προβλήματος, είτε σε επίπεδο αυτόνομης παραγωγικής μονάδας είτε σε επίπεδο δικτύου παραγωγής απαιτεί γνώσεις αυτομάτων και κατανόηση της μεθοδολογίας σε βάθος. Στο πλαίσιο αυτό η ανάπτυξη ενός πληροφοριακού εργαλείου στο οποίο τα στοιχεία και η δομή της παραγωγής και των παραγγελιών θα μπορούσαν να εισαχθούν με τρόπο πιο φιλικό προς το χρήστη ή, ιδανικά, να ληφθούν απευθείας από το λογισμικό που ήδη χρησιμοποιεί η επιχείρηση για τη διαχείριση των παραγωγικών δεδομένων της είναι μια επέκταση ιδιαίτερα σημαντική για την πρακτική αξιοποίηση της μεθόδου. Σε ότι αφορά στην περίπτωση των δικτύων παραγωγής, το πληροφοριακό εργαλείο που θα φτιαχτεί θα πρέπει επιπρόσθετα να διαθέτει δυνατότητες τήρησης ανεξάρτητου χώρου αποκλειστικής πρόσβασης από τα συμμετέχοντα μέλη του δικτύου, το οποίο θα παρέχει στη μηχανή βελτιστοποίησης την απαραίτητη πληροφορία για το συγχρονισμό και τη βελτιστοποίηση των χρονοπρογραμμάτων των μελών του δικτύου, αλλά θα διαφυλάττει παράλληλα τα στοιχεία της κάθε επιχείρησης από τα υπόλοιπα μέλη του δικτύου.

Βιβλιογραφία

[1] Aarts, B. M. (1996) *A Parallel Local Search Algorithm for the Job Scheduling Problem, Master Thesis*, Department of Mathematics & Computing Science, Eindhoven University of Technology, Eidhoven, The Netherlands, April.

[2] Aarts, E. H. L. and Lenstra, J, K, (eds) (1997) *Local Search in Combinatorial Optimization*, Wiley, Chichester.

[3] Aarts, E. H. L., Van Laardhoven, P. J. M., Lenstra, J. K. and Ulder, N. L. J. (1994) A Computational Study of Local Search Algorithms for Job-Shop Scheduling, *ORSA Journal on Computing*, 6(2), Spring, 118-125.

[4] Abdeddaiem, Y., Asarin, E., Maler, O. (2006) Scheduling with timed automata, Theoretical Computer Science 354(2), 272-300

[5] Abdeddaim, Y. and Maler, O. (2002) Job shop scheduling using timed automata, LNCS 2102, Springer, pp. 478-492

[6] Abdeddaim, Y. and Maler, O., (2001) Job-Shop Scheduling using Timed Automata in G. Berry, H. Comon and A. Finkel (Eds) *Proc CAV'01*, 478-492, LNCS 2102 Springer 2001.

[7] Abdeddaim, Y. and Maler, O., (2002), Preemptive Job-Shop Scheduling using Stopwatch Automata, in Katoen and P. Stevens (Eds), *TACAS'02*, 113-126, LNCS 2280, Springer 2002.

[8] Abdeddaim, Y., Kerbaa, A. and Maler, O. (2003) Task graph scheduling using timed automata, Proceedings of the 17th International Symposium on Parallel and Distributed Processing, IPDPS '03, IEEE Computer Society, Washington, DC, USA, pp. 237-245

[9] Adams, J,. Balas, E. and Zawack, D. (1988) The Shifting Bottleneck Procedure for Job-Shop Scheduling, *Management Science*, March, 34(3), 391-401.

[10] Akers, S. B. (1956) A Graphical Approach to Production Scheduling Problems, *Operations Research*, vol 4. 244-245.

[11] Altisen, G., Goessler, A., Pnueli, J., Sifakis, S., Tripakis and Yovine, S.(1999) A Framework for Scheduler Synthesis, *Proc. RTSS'99*, 154-163, IEEE, 1999.

[12] Alur, R., Courcoubetis, C. and Dill, D. L., (1993) Model Checking in Dense Real Time, *Information and Computation* 104, 2-34.

[13] Alur, R., Dill, D. L., (1994) A Theory of Timed Automata, *Theoritical Computer Science*, 126, 183-235.

[14] Applegate, D. and Cook, W. (1991) A Computational Study of the Job-Shop Scheduling Problem, *ORSA Journal on Computing*, Spring, 3(2), 149-156.

[15] Asarin, E., Maler, O. (1999) As Soon as Possible: Time Optimal Control for Timed Automata, *Proc, HSCC'99*, 19-30, LNCS 1569, Springer.

[16] Asarin, E., Maler, O., Pnueli, A. and Sifakis, J. (1998) Controller Synthesis for Timed Automata, *Proc. IFAC Symposium on System Structure and Control*, 469-474, Elsevier.

[17] Ashour, S. (1967) A Decomposition Approach for the Machine Scheduling Problem, *International Journal of Production Research* 6(2), 109-122.

[18] Baker, K., R. (1974) *Introduction to Sequencing and Scheduling*, John Wiley, New York.

[19] Balas, E. (1969) Machine Scheduling via Disjunctive Graphs: An Implicit Enumeration Algorithm, *Operations Research*, vol 17, 941-957.

[20] Balas, E., Lenstra, J. K. and Vazacopoulos, A. (1995) The One-Machine Problem with Deleyed Precedence Constraints and its use in Job-Shop Scheduling, *Management Science*, Jan, 41(1), 94-109.

[21] Barker, J. R. and McMahon, G. B. (1985) Scheduling in General Job-Shop, *Management Science*, May, 31(5), 594-598.

[22] Barnes, J. W. and Chambers, J. B. (1995) Solving the Job-Shop Scheduling Problem Using Tabu Search, *IEE Transactions*, vol 27, 257-263.

[23] Barnes, J. W. and Laguna, M. (1993) A Tabu Search Experience in Production Scheduling, *Annals of Operations Research*, vol 41141-156.

[24] Bean, J. (1994) Genetic Algorithms and Random Keys for Sequencing and Optimization, *ORSA J. Computing*, 6(2), 154-160.

[25] Behrmann, G., Fehnker, A., Hune, T. S., Petterson, P., Larsen, K. G. and Romijn, J. (2001) Efficient guiding towards cost-optimality in uppaal, Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pp. 174-188

[26] Bhaskaran, K. and Pinedo, M. (1991) Dispatching in Salvendy, G. (ed.) *Handbook of Industrial Engineering*, John Wiley and Sons, New York, chapter 83.

[27] Blazewich, J., Dror, M. and Weglerz, J. (1991) Mathematical Programming Formulations for Machine Scheduling: A Survey, *European Journal of Operational Research*, Invited Review, 51(3), April 15, 283-300.

[28] Bongers, M. M. P. and Bakker, B. H. (2006) Application of multi-stage scheduling, Proceedings of the 16th European Symposium on Computer Aided Process Engi-neering, ESCAPE ’06, pp. 1917-1922

[29] Bowman, E. H. (1959) The Schedule-Sequencing Problem, *Operations Research*, vol 7, 621-624.

[30] Brooks, G. H. and White, C. R. (1965) An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem, *Journal of Industrial Engineering*, January, 16(1), 34-40.

[31] Brown, A. P. G. and Lomnicki, Z. A. (1966) Some Applications of the Branch-and-Bound Algorithm to the Machine Scheduling Problem, *Operational Research Quarterly*, 17(2), 173-186.

[32] Brucker, P. (1988) An Efficient Algorithm for the Job-Shop Problem with Two Jobs, *Computing*, vol 40, 353-359.

[33] Brucker, P. and Jurisch, B. (1993) A New Lower Bound for the Job-Shop Scheduling Problem, *European Journal of Operationl Research*, vol 64, 156-167.

[34] Brunel OR-Library http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html

[35] Carlier, J. (1982) The One-Machine Scheduling Problem, *European Journal of Operational Research*, vol 11, 42-57.

[36] Carlier, J. and Prinson, E. (1989) An Algorithm for Solving the Job-Shop Problem, *Management Science*, Feb, 35(2), 164-176.

[37] Cassez, F. and Larsen, K. G. (2000) On the Impressive Power of Stopwatches, in C. Palamidessi (Ed.) *Proc. CONCUR’2000*, 138-152, LNCS 1877, Springer.

[38] Chang, Y. L., Sueyoshi, T. and Sullivan, R. S. (1996) Ranking Dispatching Rules by Data Envelopment Analysis in a Job-Shop Environment, *IIE Transactions*, 28(8), 631-642.

[39] Chaochen, Z., Hoare, C. A. R. and Ravn, A., P. (1991) A Calculus of Durations, *Information Processing Letters*, 40, 269-276.

[40] Chase, R., Aquliano, N., Jacobs, R. (1998) Production and operations management: manufacturing and services, Irwin/McGraw-Hill

[41] Cheng, R., Gen, M., Tujimura, Y. (1996) A Tutorial Survey of Job-Shop Scheduling Problems using Genetic Algorithms-I Representation, *Computers & Industrial Engineering*, 30(4), 983-997.

[42] Chu, C., Portmann, M., C. and Proth, J. M. (1992) A Splitting-Up Approach to Simplify Job-Shop Scheduling Problems, *International Journal of Production Research*, 30(4), 859-870.

[43] Conway, R. W., Maxwell, W. L. and Miller, L. W. (1967) *Theory of Scheduling*, Addison-Wesley, Reading Massachusetts.

[44] Crowston, W. B., Glover, F., Thompson, G. L., Trawick, J. D. (1963) Probabilistic and Parametric Learning Combinations of Local Job-Shop Scheduling Rules, ONR Research Memoradum, No 117, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburg, PA 15213, USA.

[45] Dauzere-Peres, S. (1995) A Procedure for the One Machine Sequencing Problem with Dependent Jobs, *European Journal of Operational Research*, vol 81, 579-589.

[46] Dauzere-Peres, S. and Lasserre, J. B. (1993) A Modified Shifting Bottleneck Procedure for Job-Shop Scheduling, *International Journal of Production Research*, April 31(4), 923-932.

[47] Davenport, A. and Kalagnanam, J. (2007) Scheduling steel production using mixed- integer programming and constraint programming, Proceedings of MISTA 2007, pp. 120-127

[48] Davis, L. (1985) Job-Shop Scheduling with Genetic Algorithms, in Greffestette J. J. (ed) *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, Pittsburg, PA, USA, Lawrence Erlbaum, pp. 136-140.

[49] Della Croce, F., Menga, G., Tadei, R., Cavalotto, M. and Petri, L. (1993) Cellular Control of Manufacturing Systems, *European Journal of Operational Research*, vol 69, 498-509.

[50] Della Croce, F., Tadei, R., and Ronaldo, R. (1994) Solving a Real World Project Scheduling Problem with a Genetic Approach, *Belgian Journal of Operations Research, Statistics and Computer Science*, 33(1-2), 65-78.

[51] Demirkol, E., Mehta, S. and Uzsoy, R. (1998), Benchmarks for shop scheduling problems, European Journal of Operational Research 109(1), 137–141

[52] Deshpande A., Gollu A. and Semenzato L. (1998) The SHIFT programming language for dynamic networks of hybrid automata, IEEE Transactions on Automatic Control, 43(4):584-587, April 1998.

[53] Dijkstra, E. (1959) A note on two problems connection with graphs, Numerische Mathematik 1, 269-271.

[54] Dorndorf, U. and Pesch, E. (1995) Evolution Based Learning in a Job-Shop Scheduling Environment, *Computers and Operations Research*, Jan, 22(1), 25-40.

[55] Erschler, J., Roubellat, F. and Vernhes, J. P. (1976) Finding Some Essential Characteristics of the Feasible Solutions for a Scheduling Problem, *Operations Research*, 24(4), Jul-Aug, (Technical Note), 774-783.

[56] Falkenauer, E. and Bouffouix, S. (1991) A Genetic Algorithm for the Job-Shop, *Proceedings of the 5th International Conference on Robotics and Automation*, Sacramento, California, USA, pp. 824-829.

[57] Fehnker, A. (1999) Scheduling a steel plant with timed automata, In Proceedings of the 6th International Conference on Real-Time Computing Systems and Applica- tions, pp. 280-286

[58] Fehnker, A. (2000) Bounding and heuristics in forward reachability algorithms, Internal report, Computing Science Institute Nijmegen

[59] Fisher, H. and Thompson, G. L. (1963) Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules, in Muth, J. F. and Thompson, G. L. (eds) *Industrial Scheduling*, Prentice Hall, Englewood Cliffs, New Jersey, Ch 15, pp. 225-251.

[60] Fisher, M. L. (1973a) Optimal Solution of Scheduling Problem using Langrange Multipliers: Part I, *Operations Research*, vol 21, 1114-1127.

[61] Fisher, M. L. (1973b) Optimal Solution of Scheduling Problem using Langrange Multipliers: Part II, *Symposium on the Theory of Scheduling and its Applications*, Springer, Berlin.

[62] Fizzano, P., Stein, C., Karger, D. and Wein, J. (1997) Job Scheduling in Rings, *Journal of Parallel and Distributed Computing*, 45(2), 122-133.

[63] Foo, S. Y. and Takefuji, Y. (1988a) Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 1. Problem Representation, in Kosko B. (ed), *IEEE International Conference on Neural Networks*, San Diego, USA, 24-27 Jul, vol 2, pp. 275-282.

[64] Foo, S. Y. and Takefuji, Y. (1988b) Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 2: Architecture and Simulations, in Kosko B. (ed), *IEEE International Conference on Neural Networks*, San Diego, California, USA, 24-27 Jul, vol 2, pp. 283-290.

[65] Foo, S. Y. and Takefuji, Y. (1988c) Integer Linear Programming Neural Networks for Job-Shop Scheduling, in Kosko B. (ed), *IEEE International Conference on Neural Networks*, San Diego, California, USA, 24-27 Jul, vol 2, pp. 341-348.

[66] French, S. (1982) Sequencing and Scheduling – *An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood, John-Wilsey & Sons, New York.

[67] Frits Vaandrager, A First Introduction to Uppaal, Institute for Computing and Information Sciences, Radboud University Nijmegen

[68] Gerd Behrmann, Alexandre David, and Kim G. Larsen, A Tutorial on Uppaal, Department of Computer Science, Aalborg University, Denmark

[69] Gere, W. S. Jr. (1966) Heuristics in Job-Shop Scheduling, *Management Science*, vol 13, 167-190.

[70] Giffler, B. and Thompson, G. L. (1960) Algorithms for Solving Production Scheduling Problems, *Operations Research*, 8(4), 487-503.

[71] Glover, F. (1977) Heuristics for Integer Programming Using Surrogate Constraints, *Decision Sciences*, 8(1), 156-166.

[72] Glover, F. (1986) Future Paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research*, 13(5), 533-549.

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

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

[75] Glover, F. and Greenberg, H. J. (1989) New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence, *European Journal of Operational Research*, 39(2), 119-130.

[76] Google OR-Tools https://code.google.com/p/or-tools/source/browse/trunk/data/jobshop/?r=830

[77] Grabowski, J., Nowicki, E. and Zdrzalka, S. (1986) A Block Approach for Single Machine Scheduling with Release Dates and Due Dates, *European Journal of Operational Research*, August, 26(2), 278-285.

[78] Grafchart website: http://www.control.lth.se/~karlerik/grafchart.html

[79] Greenberg, H. (1968) A Branch and Bound Solution to the General Scheduling Problem, *Operations Research*, March, 16(2), 353-361.

[80] Hara, H. (1995) Job-Shop Scheduling by Minimal Conflict Search, *Japanese Artificial Intelligence Society*, January, 10(1), 88-93.

[81] Haupt, R. (1989) A Survey of Priority Rule Based Scheduling, *OR Spektrum*, vol 11, 3-16.

[82] Hefetz, N. and Adiri, I. (1982) An Efficient Optimal Algorithm for the Two-Machines Unit-Time Job-Shop Schedule-Length Problem, *Mathematics of Operations Research*, vol 7, 354-360.

[83] Henzinger, T., Nicollin, X., Sifakis, J. and Yovine, S. (1994) Symbolic Model-checking for Real-time Systems, *Information and Computation* 111, 193-244.

[84] Hoitomt, D. J,. Luh, P. b. and Pattipati, K. R. (1993) Practical Approach to Job-Shop Scheduling Problems, *IEEE Trans Rob Autom*, Feb, 9(1), 1-13.

[85] Uppaal Official Website http://www.uppaal.org

[86] Diagen Official Website http://www2-data.informatik.unibw-muenchen.de/DiaGen

[87] Kronos Official Website http://www-verimag.imag.fr/TEMPORISE/kronos/

[88] HyTech website http://www-cad.eecs.berkeley.edu/~tah/HyTech/

[89] Ignall, E. and Schrage, L. (1965) Application of the Branch and Bound Technique to some Flow-Shop Scheduling Problem, *Operations Research*, vol 13, 400-412.

[90] Jackson J. R. (1955) Scheduling a Production Line to Minimize Maximum Tardiness, Research Report 43, Management Science Research Projects, University of California, Los Angeles, USA.

[91] Jackson J. R. (1956) An Extension of Johnson's Result on Job Lot Scheduling, *Naval Research Logistics Quarterly*, 3(3), 201-203.

[92] Jackson J. R. (1957) Simulation Research on Job-Shop Production, *Naval Research Logistics Quarterly*, vol 4, 287-295.

[93] Jain, A., S. and Meeran, S. (1999) Deterministic Job-Shop Scheduling: Past, Present and Future, *European Journal of Operational Research*, 113, 390-434.

[94] Johan Bengtsson, Kim Larsen, Fredik Larsson, Paul Pettersson, Wang Yi: UPPAAL - a Tool Suite for Automatic Verification of Real-Time Systems

[95] Johnson, S. M. (1954) Optimal Two- and Three-Stage Production Schedules with Set-Up Times Included, *Naval Research Logistics Quarterly*, vol 1, 61-68.

[96] Julian Proenza,The UPPAAL Model Checker,Systems, Robotics and Vision Group. UIB. SPAIN,Oct. 2008

[97] Kesten, Y., Pnueli, A., Sifakis, J. and Yovine, S. (1999) Decidable Integration Graphs, *Information and Computation*, 150, 209-243.

[98] Kobayashi, S., Ono, I. and Yamamura, M. (1995) An Efficient Algorithm for Job-Shop Scheduling Problems, *Proceedings of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 506-511.

[99] Kokkinakos, P., Markaki, O., Panopoulos, D., Koussouris, S., Askounis, D. (2013) Dynamic Manufacturing Networks monitoring and governance, IFIP Advances in Information and Communication Technology, 398 (PART 2), pp. 446-453

[100] Kruger, K., Shakhlevich, N. V., Sotskov, Y. N. and Werner, F. (1995) A Heuristic Decomposition Algorithm for Scheduling Problems on Mixed Graphs, *Journal of Operational Research Society*, vol 46, 1481-1497.

[101] Lageweg, B. J., Lenstra, K. and Rinnooy Kan, A. H. G. (1977) Job-Shop Scheduling by Implicit Enumeration, *Management Science*, December, 24(4), 441-450.

[102] Laguna, M. and Glover, F. W. (1993) Integrating Target Analysis and Tabu Search for Improved Scheduling Systems, *Expert Systems with Applications*, vol 6, 287-297.

[103] Laguna, M., Barnes, L. W. and Glover, F. W. (1991) Tabu Search Methods for a Single Machine Scheduling Problem, *Journal of Intelligence Manufacturing*, vol 2, 63-74.

[104] Laguna, M., Barnes, L. W. and Glover, F. W. (1993) Intelligence Scheduling with Tabu Search: An Application to Jobs with Linear Delay Penalties and Sequence-Dependent Setup Costs and Times, *Journal of Applied Intelligence*, vol 3, 159-172.

[105] Lawrence, S. (1984) Supplement to Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, USA, October.

[106] Legriel, J. and Maler, O. (2011) Meeting deadlines cheaply, Proceedings of the 2011 23rd Euromicro Conference on Real-Time Systems, ECRTS '11, IEEE Computer So- ciety, Washington, DC, USA, pp. 185-194

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

[108] Manne, A. S. (1960) On the Job-Shop Scheduling Problem, *Operations Research*, vol 8, 219-223.

[109] Marius Mikucionis, Egle Sasnauskaite - On the fly testing using UPPAAL (Master thesis)

[110] Markaki, O., Kokkinakos, P., Panopoulos, D., Koussouris, S., Askounis, D. (2013) Benefits and risks in Dynamic Manufacturing Networks, IFIP Advances in Information and Communication Technology, 398 (PART 2), pp. 438-445

[111] Markaki, O., Panopoulos, D., Kokkinakos, P., Koussouris, S., Askounis, D. (2013) Towards adopting dynamic manufacturing networks for future manufacturing: Benefits and risks of the IMAGINE DMN end-to-end management methodology, Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, art. no. 6570632, pp. 305-310

[112] Martin, O., Otto, S. W. and Felten, E. W. (1989) Large-Step Markov Chains for Traveling Salesman Problem, *Complex Systems*, vol 5, 299-326.

[113] Martin, O., Otto, S. W. and Felten, E. W. (1992) Large-Step Markov Chains for TSP Incorporating Local Search Heuristics, *Operations Research Letters*, vol 11, 219-224.

[114] Martin, P. D. (1996) A Time-Oriented Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem, *Ph. Thesis*, School of Operations Research & Industrial Engineering, Cornell University, Ithaca, New York 14853-3801, August.

[115] Matsuo, H., Suh, C. J. and Sullivan, R. S. (1988) A Control Search Simulated Annealing Method for the General Job-Shop Scheduling Problem, Working

Paper #03-04-88, Graduate School of Business, The University of Texas at Austin, Austin, Texas, USA.

[116] McMahon, G. B. and Florian, M. (1975) On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness, *Operations Research*, May-June, 23(3), 475-482.

[117] McManis, J. and Varaiya, P. (1994) Suspension Automata: A Decidable Class of Hybrid Automata, in D. L. Dill (Ed.) *Proc. CAV'94*, 105-117, LNCS 818, Springer.

[118] Moby/PLC website http://csd.informatik.uni-oldenburg.de/projects/emobyplc

[119] Morton, T. E. and Pentico, D. W. (1993) *Heuristic Scheduling Systems*, Wiley Series in Engineering and Technology Management, Wiley, New York.

[120] Nakano, R. and Yamada, T. (1991) Conventional Genetic Algorithm for Job-Shop Problems, in Kenneth, M. K. Brooker, L. B. (eds) *Proceedings of the 4th International Conference on Genetic Algorithms and their Applications*, San Diego, USA, pp. 474-479.

[121] Nawaz, M., Enscore, E. E. Jr. and Ham, I. (1983) A Heuristic Algorithm for the *m*-Machine *n*-Job Flow-Shop Sequencing Problem, *OMEGA The International Journal of Management Science*, 11(1), 91-95.

[122] Nemhauser, G. L. and Wolsey, L. A. (1988) *Integer and Combinatorial Optimasation*, John Wiley and Sons, New York.

[123] Niebert, P. and Yovine, S. (2000) Computing optimal operation schemes for chem- ical plants in multi-batch mode, in N. Lynch and B. Krogh (eds), Hybrid Systems Computation and Control, LNCS 1790, Springer, pp. 338-351

[124] Nowicki, E. and Smutnicki, C. (1996) A Fast Taboo Search Algorithm for the Job-Shop Problem, *Management Science*, 42(6), 797-813.

[125] Osman, J. H. and Kelly, J. P. (1996a) (eds) *Meta-Heuristics: Theory and Applications*, Kluwer Academic Publishers, Norwell, MA, USA.

[126] Panek, S., Engell, S., Subbiah, S. and Stursberg, O. (2008) Scheduling of multi- product batch plants based upon timed automata models, Computers and Chem- ical Engineering 32, 275-291

[127] Panek, S., Stursberg, O. and Engell, S (2004) Job-shop scheduling by combining reachability analysis with linear programming, 7th Int. IFAC Workshop on Discrete Event Systems, IFAC, Reims, pp. 199-204

[128] Panek, S., Stursberg, O. and Engell, S. (2006) Efficient synthesis of production schedules by optimization of timed automata, Control Engineering and Practice 14(10), 1183-1197

[129] Panopoulos, D., Metaxiotis, K. (2006) Solving production scheduling problems using advanced model checking tools, International Journal of Computer Applications in Technology, 26 (1-2), pp. 37-48

[130] Panopoulos, D., Ptochos, D., Oikonomou, A., Askounis, D., Psarras, J. (2003) Combining Neural Networks and CLP for Production Scheduling, WSEAS Transactions on Circuits and Systems, Issue 4, Vol. 2, pp. 826-831

[131] Panwalkar, S. S. and Inskander, W. (1977) A Survey of Scheduling Rules, *Operations Research*, Jan-Feb, 25(1),45-61.

[132] Paul Pettersson - Modelling and Verification of Real-Time Systems Using Timed-Automata

[133] Perregaard M (1998) Branch and Bound Methods for the Multi-Processor Job Shop and Flow Shop Scheduling Problems, University of Copenhagen

[134] Perregaard, M. and Clausen, J. (1995) Parallel Branch-and-Bound Methods for the Job-Shop Scheduling Problem, Working Paper, University of Copenhagen, Denmark.

[135] PINEDO, M. (2002) Scheduling: Theory, Algorithms and Applications, 2nd Edition, Prentice-Hall, New York

[136] Rowe, A. J. and Jackson, J. R. (1956) Research Problems in Production Routing and Scheduling, *Journal of Industrial Engineering*, vol 7, 116-121.

[137] Roy, B. and Sussmann, B. (1964) Les Problemes d'Ordonnancement avec Constraintes Disjonctives, Note D.S. no. 9 bis, SEMA, Paris, France, Decembre.

[138] Sabuncuoglu, I. and Gurgun, B. (1996) A Neural Network Model for Scheduling Problems, *European Journal of Operational Research*, 93(2), Sept, 288-299.

[139] Shift website: http://www.path.berkeley.edu/shift/Default.htm

[140] Shmoys, D. B., Stein, C. and Wein, C. (1994) Improved Approximation Algorithms for Job-Shop Scheduling Problems, *SIAM J Comput*, June, 23(3), 617-632.

[141] Smith, W. E. (1956) Various Optimizers for Single Stage Production, *Naval Research Logistics Quarterly*, vol 3, 59-66.

[142] Subbiah, S. and Engell, S. (2009) Timed automata models for batch scheduling with sequence-dependent changeovers., 10th International Symposium on Pro¬cess Systems Engineering (PSE2009) pp. 1515-1520

[143] Subbiah, S., Schoppmeyer, C. and Engell, S. (2011) Efficient scheduling of batch plants using reachability tree search for timed automata with lower bound com¬putations, Proceedings ofthe 21st European Symposium on Computer Aided Pro¬cess Engineering pp. 930-934.

[144] Subbiah, S., Schoppmeyer, C. and Engell, S. (2011b) An intuitive and efficient approach to process scheduling with sequence-dependent changeovers us¬ing timed automata models., Industrial and Engineering Chemistry Research 50(9), 5131-5152.

[145] Subbiah, S., Schoppmeyer, C. and Engell, S. (2012) Modeling and solving batch scheduling problems with various storage policies and operational policies us- ingtimed automata., Proceedings of the 11th International Symposium on Process Systems Engineering p.

[146] Subbiah, S., Thomas, T., Sebastian, P. and Engell, S. (2009) Multi-product batch scheduling with intermediate due dates using priced timed automata models, Computers and Chemical Engineering 33(10), 1661-1676.

[147] Subbiah, S., Tometzki, T. and Engell, S. (2008) Batch scheduling with intermediate due dates using timed automata models, 18th European Symposium on Computer Aided Process Engineering, pp. 151-156.

[148] Sundaramoorthy, A. and Karimi, I. A. (2005) Asimpler better slot-based continuous time formulation for scheduling in multipurpose batch plants, Chemical Engineering Science 60, 2697-2702

[149] Supremica website: http://www.supremica.org

[150] Taillard, E. (1994) Parallel Taboo Search Techniques for the Job-Shop Scheduling Problems, *ORSA Journal on Computing*, 16(2), 108-117.

[151] Tamaki, H. and Nishikawa, Y. (1992) A Paralleled Genetic Algorithm Based on a Neighborhood Model and its Application to the Job-Shop Scheduling, in Manner, R. and Manderick, B. (eds) *PPSN'2 Proceedings of the 2nd International Workshopon Parallel Problem Solving from Nature*, Brussels, Belgium, pp. 573-582.

[152] Ten Eikelder, H. M. M., Aarts, B. J. M., Van Verhoeven, M. G. A. and Aarts, E. H. L. (1997) Sequential and Parallel Local Search Algorithms for Job-Shop Scheduling, *MIC'97 2nd International Conference on Meta-heuristics*, Sophia-Antipolis, France, 21-24 July, pp. 75-80 (Extended Abstract).

[153] Thomsen, S. (1997) Meta-heuristics Combined with Branch & Bound, *Technical Report*, Copenhagen Business School, Copenhagen, Denmark (in Danish).

[154] Till, J. (2007) New hybrid evolutionary algorithms for chemical batch scheduling under uncertainty, Dissertation, Universitat Dortmund, Department of Bio-Chemical and Chemical Engineering, Dortmund

[155] Tometzki, T. and Engell, S. (2011) Risk conscious solution of planning problems under uncertainty by hybrid multi-objective evolutionary algorithms, Computers and Chemical Engineering 35, 2521-2539.

[156] Van De Velde, S. (1991) Machine Scheduling and Lagrangian Relaxation, *Ph. Thesis*, CWI Amsterdam, The Netherlands.

[157] Van Hulle, M. M. (1991) A Goal Programming Network for Mixed Integer Linear Programming: A Case Study for the Job-Shop Scheduling Problem, *International Journal of Neural Networks*, 2(3), 201-209.

[158] Van Laarhooven P. J. M., Aarts, E. H. L. and Lenstra, J. K. (1992) Job-Shop Scheduling by Simulated Annealing, *Operations Research*, Jan-Feb, 40(1), 113-125.

[159] Viviers, F. (1983) A Decision Support System for Job-Shop Scheduling, *European Journal of Operational Research*, Sept, 14(1), 95-103.

[160] Werner, F. and Winkler, A. (1995) Insertion Techniques for the Heuristic Solution of the Job-Shop Problem, *Discrete Applied Mathematics*, 58(2), 191-211.

[161] Willems, T. M. and Rooda, J. E. (1994) Neural Networks for Job-Shop Scheduling, *Control Eng. Practice*, Feb, 2(1), 31-39.

[162] Williamson, D. P., Hall, L. A., Hoogeveen, J. A., Hurkens, C. A. J., Lenstra, J. K., Sevast'janov, S. V. and Shmoys, D. B. (1997) Short Shop Schedules, *Operations Research*, March-April, 45(2), 288-294.

[163] Yaqing, T., Wei, H., Yanming, S. and Yong, Y. (2012) Comparative study of different approaches to solve batch process scheduling and optimisation problems, Pro-ceedings ofthe 18th International Conference on Automation and Computing

[164] Zheng, J., Zhou, X. and Ye, T. (2008) A review for short-term scheduling of batch processes, IEEE International Conference on Automation and Logistics, pp. 1960 -1964

[165] Zhou, D. N., Cherkassky, V., Baldwin, T. R. and Olson D. E. (1991) A Neural Network Approach to Job-Shop Scheduling, *IEEE Transactions on Neural Network*, 2(1), 175-179.

[166] Αφράτη, Φ., Παπαγεωργίου Γ. (1993) *Αλγόριθμοι*: *Μέθοδοι Σχεδίασης και Ανάλυση Πολυπλοκότητας*, Εκδ. Συμμετρία.

[167] Εργαζάκης, Κ. (2001) 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.

[168] Μεταξιώτης, Κ. (2001) Ολοκληρωμένη Μεθοδολογία Μοντελοποίησης Συστήματος Διαχείρισης & Ευφυούς Χρονοπρογραμματισμού Παραγωγής σε ERP Περιβάλλον, ΕΜΠ

[169] Μποζάνης, Π. (2003) *Αλγόριθμοι, Σχεδιασμός και Ανάλυση*, Εκδ. Τζιόλα

[170] Παππής, Κ. (2001) *Διοίκηση Παραγωγής - Ο σχεδιασμός παραγωγικών συστημάτων*, Εκδ. Α. Σταμούλης

[171] Παππής, Κ. (2001) *Προγραμματισμός Παραγωγής*, Εκδ. Α. Σταμούλης

Παράρτημα 1: Ενδεικτική δομή δεδομένων και αποτελεσμάτων της μηχανής βελτιστοποίησης

Στο παράρτημα παρουσιάζεται ένα ενδεικτικό αρχείο xml το οποίο αναπαριστά το μοντέλο αυτομάτων το οποίο αναλύθηκε στα πειράματα του Κεφαλαίου 8 της διατριβής, καθώς επίσης και τα εξαγόμενα σε raw format της εφαρμογής του αλγορίθμου επίλυσης μέσω της μηχανής βελτιστοποίησης της βιβλιοθήκης cora

Αρχείο xml αναπαράστασης του συστήματος της παραγράφου 8.3

<?xml version="1.0" encoding="utf-8"?> <!DOCTYPE nta PUBLIC '-//Uppaal Team//DTD Flat System 1.1//EN' 'http://www.it.uu.se/research/group/darts/uppaal/flat-1_1.dtd'> <nta> <declaration>//Insert declarations of global clocks, variables, constants and channels.
chan machOn[6];
chan machOff[6];
chan iTimerOn[6];
chan iTimerOff[6];
clock glc;
bool busy[6]={false, false, false, false, false, false};
urgent broadcast chan StepEnd;
chan trigger;
int best=1000;
meta int[-1000,1000] remaining;
int glt=0;
int extra=0;</declaration> <template> <name x="5" y="5">Job</name> <parameter>const int m0, const int d0, const int m1, const int d1, const int m2, const int d2, const int m3, const int d3, const int m4, const int d4, const int m5, const int d5, const int Target, const int Early, const int Late</parameter> <declaration>clock c;
clock k;
int targetDeviation=0;</declaration> <location id="id0" x="-128" y="504"> <name x="-138" y="474">end</name> </location> <location id="id1" x="-40" y="584"> <urgent/> </location> <location id="id2" x="304" y="440"> <name x="294" y="410">t5</name> <label kind="invariant" x="294" y="455">c<=d5</label> </location> <location id="id3" x="400" y="600"> <name x="390" y="570">c5</name> </location> <location id="id4" x="400" y="440"> <name x="390" y="410">s5</name> </location> <location id="id5" x="552" y="440"> <name x="542" y="410">t4</name> <label kind="invariant" x="542" y="455">c<=d4</label> </location> <location id="id6" x="680" y="600"> <name x="670" y="570">c4</name> <urgent/> </location> <location id="id7" x="680" y="440"> <name x="670" y="410">s4</name> </location> <location id="id8" x="-288" y="208"> <name x="-298" y="178">s0</name> </location> <location id="id9" x="-136" y="208"> <name x="-146" y="178">t0</name> <label kind="invariant" x="-168" y="224">c<=d0</label> </location> <location id="id10" x="8" y="208"> <name x="-2" y="178">s1</name> </location> <location id="id11" x="128" y="208"> <name x="118" y="178">t1</name> <label kind="invariant" x="104" y="224">c<=d1</label> </location> <location id="id12" x="272" y="208"> <name x="262" y="178">s2</name> </location> <location id="id13" x="416" y="208"> <name x="406" y="178">t2</name> <label kind="invariant" x="384" y="224">c<=d2</label> </location> <location id="id14" x="560" y="208"> <name x="550" y="178">s3</name> </location> <location id="id15" x="680" y="216"> <name x="670" y="186">t3</name> <label kind="invariant" x="670" y="231">c<=d3</label> </location> <location id="id16" x="160" y="440"> <urgent/> </location> <location id="id17" x="272" y="-16"> <name x="262" y="-46">stop</name> </location> <location id="id18" x="-288" y="368"> <name x="-298" y="338">c0</name> <urgent/> </location> <location id="id19" x="8" y="368"> <name x="-2" y="338">c1</name> <urgent/> </location> <location id="id20" x="272" y="368"> <name x="262" y="338">c2</name> <urgent/> </location> <location id="id21" x="560" y="368"> <name x="550" y="338">c3</name> <urgent/> </location> <init ref="id8"/> <transition> <source ref="id1"/> <target ref="id0"/> <nail x="-128" y="584"/> </transition> <transition> <source ref="id16"/> <target ref="id1"/> <label kind="guard" x="8" y="560">glt>Target</label> <label kind="assignment" x="-128" y="592">targetDeviation=Late*(glt-Target), cost+=targetDeviation</label> <nail x="160" y="584"/> </transition> <transition> <source ref="id16"/> <target ref="id1"/> <label kind="guard" x="-24" y="448">glt<=Target</label> <label kind="assignment" x="-144" y="408">targetDeviation=Early*(Target-glt), cost+=targetDeviation</label> <nail x="-40" y="440"/> </transition> <transition> <source ref="id2"/> <target ref="id17"/> <label kind="guard" x="228" y="182">(glc>best)</label> <nail x="304" y="256"/> </transition> <transition> <source ref="id5"/> <target ref="id17"/> <label kind="guard" x="352" y="182">(glc>best)</label> <nail x="552" y="392"/> <nail x="336" y="392"/> </transition> <transition> <source ref="id3"/> <target ref="id4"/> <label kind="guard" x="416" y="504">busy[m5]</label> <nail x="440" y="504"/> </transition> <transition> <source ref="id3"/> <target ref="id2"/> <label kind="synchronisation" x="272" y="512">machOn[m5]!</label> <label kind="assignment" x="280" y="528">c=0</label> <nail x="304" y="600"/> </transition> <transition> <source ref="id4"/> <target ref="id3"/> <label kind="synchronisation" x="352" y="488">StepEnd?</label> <nail x="376" y="496"/> </transition> <transition> <source ref="id5"/> <target ref="id4"/> <label kind="guard" x="448" y="408">(c>=d4) and (glc<=best)</label> <label kind="synchronisation" x="448" y="440">machOff[m4]!</label> </transition> <transition> <source ref="id6"/> <target ref="id5"/> <label kind="synchronisation" x="528" y="520">machOn[m4]!</label> <label kind="assignment" x="528" y="536">c=0</label> <nail x="552" y="600"/> </transition> <transition> <source ref="id6"/> <target ref="id7"/> <label kind="guard" x="712" y="480">busy[m4]</label> <nail x="712" y="488"/> </transition> <transition> <source ref="id7"/> <target ref="id6"/> <label kind="synchronisation" x="620" y="465">StepEnd?</label> <nail x="656" y="480"/> </transition> <transition> <source ref="id15"/> <target ref="id7"/> <label kind="guard" x="752"

y="288">(c>=d3) and
(glc<=best)</label> <label kind="synchronisation" x="744" y="328">machOff[m3]!</label> <nail
x="744" y="216"/> <nail x="744" y="440"/> </transition> <transition> <source ref="id2"/> <target
ref="id16"/> <label kind="guard" x="200" y="448">(c>=d5) and
(glc<=best)</label> <label kind="synchronisation" x="192" y="416">machOff[m5]!</label>
</transition> <transition> <source ref="id8"/> <target ref="id9"/> <label kind="guard" x="-232"
y="184">glc==0</label> <label kind="synchronisation" x="-248" y="168">machOn[m0]!</label> <label
kind="assignment" x="-224" y="128">c=0</label> <nail x="-256" y="160"/> <nail x="-176" y="160"/>
</transition> <transition> <source ref="id9"/> <target ref="id10"/> <label kind="guard" x="-88"
y="168">(c>=d0) and
(glc<=best)</label> <label kind="synchronisation" x="-104" y="224">machOff[m0]!</label>
</transition> <transition> <source ref="id11"/> <target ref="id12"/> <label kind="guard" x="176"
y="176">(c>=d1) and
(glc<=best)</label> <label kind="synchronisation" x="160" y="216">machOff[m1]!</label>
</transition> <transition> <source ref="id13"/> <target ref="id14"/> <label kind="guard" x="456"
y="176">(c>=d2) and
(glc<=best)</label> <label kind="synchronisation" x="456" y="216">machOff[m2]!</label>
</transition> <transition> <source ref="id9"/> <target ref="id17"/> <label kind="guard" x="-176"
y="8">(glc>best)</label> <nail x="-136" y="-16"/> </transition> <transition> <source ref="id11"/>
<target ref="id17"/> <label kind="guard" x="140" y="66">(glc>best)</label> </transition> <transition>
<source ref="id13"/> <target ref="id17"/> <label kind="guard" x="284" y="66">(glc>best)</label>
</transition> <transition> <source ref="id15"/> <target ref="id17"/> <label kind="guard" x="640"
y="48">(glc>best)</label> <nail x="680" y="-16"/> </transition> <transition> <source ref="id18"/>
<target ref="id9"/> <label kind="synchronisation" x="-184" y="320">machOn[m0]!</label> <label
kind="assignment" x="-152" y="336">c=0</label> <nail x="-136" y="368"/> </transition> <transition>
<source ref="id19"/> <target ref="id11"/> <label kind="synchronisation" x="96"
y="312">machOn[m1]!</label> <label kind="assignment" x="120" y="336">c=0</label> <nail x="128"
y="368"/> </transition> <transition> <source ref="id21"/> <target ref="id15"/> <label
kind="synchronisation" x="632" y="320">machOn[m3]!</label> <label kind="assignment" x="664"
y="344">c=0</label> <nail x="680" y="368"/> </transition> <transition> <source ref="id20"/> <target
ref="id13"/> <label kind="synchronisation" x="376" y="312">machOn[m2]!</label> <label
kind="assignment" x="400" y="336">c=0</label> <nail x="416" y="368"/> </transition> <transition>
<source ref="id10"/> <target ref="id19"/> <label kind="synchronisation" x="-52"
y="281">StepEnd?</label> <nail x="-32" y="344"/> </transition> <transition> <source ref="id12"/>
<target ref="id20"/> <label kind="synchronisation" x="212" y="281">StepEnd?</label> <nail x="240"
y="344"/> </transition> <transition> <source ref="id14"/> <target ref="id21"/> <label
kind="synchronisation" x="500" y="285">StepEnd?</label> <nail x="528" y="336"/> </transition>
<transition> <source ref="id8"/> <target ref="id18"/> <label kind="synchronisation" x="-348"
y="281">StepEnd?</label> <nail x="-320" y="344"/> </transition> <transition> <source ref="id18"/>
<target ref="id8"/> <label kind="guard" x="-288" y="320">busy[m0]</label> <nail x="-256" y="352"/>
</transition> <transition> <source ref="id19"/> <target ref="id10"/> <label kind="guard" x="16"
y="320">busy[m1]</label> <nail x="40" y="344"/> </transition> <transition> <source ref="id20"/> <target
ref="id12"/> <label kind="guard" x="272" y="320">busy[m2]</label> <nail x="296" y="352"/>
</transition> <transition> <source ref="id21"/> <target ref="id14"/> <label kind="guard" x="544"
y="320">busy[m3]</label> <nail x="584" y="336"/> </transition> </template> <template> <name x="5"
y="5">Machine</name> <parameter>const int no, const int startupCost</parameter>
<declaration>//Insert local declarations of clocks, variables and constants.
clock cIdle;</declaration> <location id="id22" x="560" y="184"> <committed/> </location> <location
id="id23" x="72" y="96"> <name x="40" y="64">ColdStart</name> </location> <location id="id24"
x="416" y="-40"> <committed/> </location> <location id="id25" x="256" y="-40"> <name x="246" y="-
70">Off</name> <committed/> </location> <location id="id26" x="256" y="96"> <name x="246"
y="66">Busy</name> </location> <location id="id27" x="560" y="-40"> <name x="550" y="-
70">Idle</name> </location> <location id="id28" x="416" y="184"> <committed/> </location> <init
ref="id23"/> <transition> <source ref="id22"/> <target ref="id27"/> <label kind="synchronisation" x="512"
y="56">iTimerOn[no]!</label> </transition> <transition> <source ref="id28"/> <target ref="id22"/> <label
kind="synchronisation" x="456" y="160">StepEnd!</label> </transition> <transition> <source
ref="id23"/> <target ref="id26"/> <label kind="synchronisation" x="136" y="72">machOn[no]?</label>
<label kind="assignment" x="136" y="96">busy[no]=true</label> </transition> <transition> <source
ref="id25"/> <target ref="id26"/> </transition> <transition> <source ref="id24"/> <target ref="id26"/>
<label kind="guard" x="288" y="72">cIdle<=startupCost</label> <label kind="synchronisation" x="304"
y="96">iTimerOff[no]!</label> <nail x="416" y="96"/> </transition> <transition> <source ref="id24"/>
<target ref="id25"/> <label kind="guard" x="280" y="-64">cIdle>startupCost</label> <label
kind="synchronisation" x="296" y="-40">iTimerOff[no]!</label> </transition> <transition> <source
ref="id27"/> <target ref="id24"/> <label kind="synchronisation" x="448" y="-64">machOn[no]?</label>
<label kind="assignment" x="448" y="-40">busy[no]=true</label> </transition> <transition> <source
ref="id26"/> <target ref="id28"/> <label kind="synchronisation" x="288" y="160">machOff[no]?</label>
<label kind="assignment" x="280" y="184">busy[no]=false</label> <nail x="256" y="184"/> </transition>

</template> <template> <name>CostTimer</name> <location id="id29" x="-280" y="-136"> <name x="-290" y="-166">idle</name> <label kind="invariant" x="-290" y="-121">cost'==15</label> </location> <init ref="id29"/> </template> <template> <name>Timer</name> <declaration>clock c;</declaration> <location id="id30" x="-328" y="-112"> <label kind="invariant" x="-336" y="-144">c<=2</label> </location> <location id="id31" x="-480" y="-40"> <label kind="invariant" x="-490" y="-25">c<=1</label> </location> <init ref="id31"/> <transition> <source ref="id30"/> <target ref="id31"/> <label kind="guard" x="-376" y="-64">c>1</label> <label kind="assignment" x="-408" y="-40">c=0 & glt++</label> <nail x="-328" y="-40"/> </transition> <transition> <source ref="id31"/> <target ref="id30"/> <label kind="guard" x="-408" y="-136">c>0</label> <label kind="assignment" x="-456" y="-136">glt++</label> <nail x="-480" y="-112"/> </transition> </template> <template> <name>IdleTimer</name> <parameter>const int[0,5] no</parameter> <declaration>clock c;</declaration> <location id="id32" x="-704" y="96"> <label kind="invariant" x="-714" y="111">cost'==1</label> </location> <location id="id33" x="-880" y="96"> </location> <init ref="id33"/> <transition> <source ref="id32"/> <target ref="id33"/> <label kind="synchronisation" x="-848" y="-8">iTimerOff[no]?</label> <nail x="-704" y="16"/> <nail x="-880" y="16"/> </transition> <transition> <source ref="id33"/> <target ref="id32"/> <label kind="synchronisation" x="-840" y="72">iTimerOn[no]?</label> </transition> </template> <system>//Insert process assignments.
machine0 = Machine(0,20);
machine1 = Machine(1,30);
machine2 = Machine(2,15);
machine3 = Machine(3,22);
machine4 = Machine(4,35);
machine5 = Machine(5,9);
job0=Job(1,4, 3,5, 0,8, 2,6, 4,4, 5,8, 70,1,20);
job1=Job(3,6, 1,2, 5,9, 0,11, 2,7, 4,6, 20,1,25);
job2=Job(1,4, 4,7, 3,4, 2,7, 0,2, 5,5, 36,1,21);
job3=Job(5,8, 2,5, 3,6, 1,5, 4,9, 0,4, 11,2,15);
//Edit system definition.
system machine0, machine1, machine2, machine3, machine4, machine5, job0, job1, job2, job3, CostTimer, IdleTimer;
</system> </nta>

Μεταβάσεις πειράματος παραγράφου 8.3 σε raw format

job1.s0->job1.t0 { glc == 0, machOn[m0]!, c := 0 } machine3.ColdStart->machine3.Busy { 1, machOn[no]?, busy[no] := 1 }
job0.s0->job0.t0 { glc == 0, machOn[m0]!, c := 0 } machine1.ColdStart->machine1.Busy { 1, machOn[no]?, busy[no] := 1 }
job0.t0->job0.s1 { c >= d0 && glc <= best, machOff[m0]!, 1 } machine1.Busy->machine1._id28 { 1, machOff[no]?, busy[no] := 0 }
machine1._id28->machine1._id22 { 1, StepEnd!, 1 } job0.s1->job0.c1 { 1, StepEnd?, 1 } job2.s0->job2.c0 { 1, StepEnd?, 1 } job3.s0->job3.c0 { 1, StepEnd?, 1 }
machine1._id22->machine1.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(1)._id33->IdleTimer(1)._id32 { 1, iTimerOn[no]?, 1 }
job3.c0->job3.t0 { 1, machOn[m0]!, c := 0 } machine5.ColdStart->machine5.Busy { 1, machOn[no]?, busy[no] := 1 }
job2.c0->job2.t0 { 1, machOn[m0]!, c := 0 } machine1.Idle->machine1._id24 { 1, machOn[no]?, busy[no] := 1 }
machine1._id24->machine1.Busy { cIdle <= startupCost, iTimerOff[no]!, 1 } IdleTimer(1)._id32->IdleTimer(1)._id33 { 1, iTimerOff[no]?, 1 }
job0.c1->job0.s1 { busy[m1], tau, 1 }
job1.t0->job1.s1 { c >= d0 && glc <= best, machOff[m0]!, 1 } machine3.Busy->machine3._id28 { 1, machOff[no]?, busy[no] := 0 }
machine3._id28->machine3._id22 { 1, StepEnd!, 1 } job0.s1->job0.c1 { 1, StepEnd?, 1 } job1.s1->job1.c1 { 1, StepEnd?, 1 }
machine3._id22->machine3.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(3)._id33->IdleTimer(3)._id32 { 1, iTimerOn[no]?, 1 }
job0.c1->job0.t1 { 1, machOn[m1]!, c := 0 } machine3.Idle->machine3._id24 { 1, machOn[no]?, busy[no] := 1 }
machine3._id24->machine3.Busy { cIdle <= startupCost, iTimerOff[no]!, 1 } IdleTimer(3)._id32->IdleTimer(3)._id33 { 1, iTimerOff[no]?, 1 }
job1.c1->job1.s1 { busy[m1], tau, 1 }
job2.t0->job2.s1 { c >= d0 && glc <= best, machOff[m0]!, 1 } machine1.Busy->machine1._id28 { 1, machOff[no]?, busy[no] := 0 }
machine1._id28->machine1._id22 { 1, StepEnd!, 1 } job1.s1->job1.c1 { 1, StepEnd?, 1 } job2.s1->job2.c1 { 1, StepEnd?, 1 }
machine1._id22->machine1.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(1)._id33->IdleTimer(1)._id32 { 1, iTimerOn[no]?, 1 }
job2.c1->job2.t1 { 1, machOn[m1]!, c := 0 } machine4.ColdStart->machine4.Busy { 1, machOn[no]?, busy[no] := 1 }
job1.c1->job1.t1 { 1, machOn[m1]!, c := 0 } machine1.Idle->machine1._id24 { 1, machOn[no]?, busy[no] := 1 }
machine1._id24->machine1.Busy { cIdle <= startupCost, iTimerOff[no]!, 1 } IdleTimer(1)._id32->IdleTimer(1)._id33 { 1, iTimerOff[no]?, 1 }
job1.t1->job1.s2 { c >= d1 && glc <= best, machOff[m1]!, 1 } machine1.Busy->machine1._id28 { 1, machOff[no]?, busy[no] := 0 }
machine1._id28->machine1._id22 { 1, StepEnd!, 1 } job1.s2->job1.c2 { 1, StepEnd?, 1 }
machine1._id22->machine1.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(1)._id33->IdleTimer(1)._id32 { 1, iTimerOn[no]?, 1 }
job1.c2->job1.s2 { busy[m2], tau, 1 }
job0.t1->job0.s2 { c >= d1 && glc <= best, machOff[m1]!, 1 } machine3.Busy->machine3._id28 { 1, machOff[no]?, busy[no] := 0 }
machine3._id28->machine3._id22 { 1, StepEnd!, 1 } job0.s2->job0.c2 { 1, StepEnd?, 1 } job1.s2->job1.c2 { 1, StepEnd?, 1 }
machine3._id22->machine3.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(3)._id33->IdleTimer(3)._id32 { 1, iTimerOn[no]?, 1 }
job1.c2->job1.s2 { busy[m2], tau, 1 }
job0.c2->job0.t2 { 1, machOn[m2]!, c := 0 } machine0.ColdStart->machine0.Busy { 1, machOn[no]?, busy[no] := 1 }
job3.t0->job3.s1 { c >= d0 && glc <= best, machOff[m0]!, 1 } machine5.Busy->machine5._id28 { 1, machOff[no]?, busy[no] := 0 }
machine5._id28->machine5._id22 { 1, StepEnd!, 1 } job1.s2->job1.c2 { 1, StepEnd?, 1 } job3.s1->job3.c1 { 1, StepEnd?, 1 }
machine5._id22->machine5.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(5)._id33->IdleTimer(5)._id32 { 1, iTimerOn[no]?, 1 }
job3.c1->job3.t1 { 1, machOn[m1]!, c := 0 } machine2.ColdStart->machine2.Busy { 1, machOn[no]?, busy[no] := 1 }
job1.c2->job1.t2 { 1, machOn[m2]!, c := 0 } machine5.Idle->machine5._id24 { 1, machOn[no]?, busy[no] := 1 }

machine5._id24->machine5.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(5)._id32->IdleTimer(5)._id33 { 1, iTimerOff[no]?, 1 }
machine5.Off->machine5.Busy { 1, tau, 1 }
job2.t1->job2.s2 { c >= d1 && glc <= best, machOff[m1]!, 1 } machine4.Busy->machine4._id28 { 1, machOff[no]?, busy[no] := 0 }
machine4._id28->machine4._id22 { 1, StepEnd!, 1 } job2.s2->job2.c2 { 1, StepEnd?, 1 }
machine4._id22->machine4.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(4)._id33->IdleTimer(4)._id32 { 1, iTimerOn[no]?, 1 }
job2.c2->job2.t2 { 1, machOn[m2]!, c := 0 } machine3.Idle->machine3._id24 { 1, machOn[no]?, busy[no] := 1 }
machine3._id24->machine3.Busy { cIdle <= startupCost, iTimerOff[no]!, 1 } IdleTimer(3)._id32->IdleTimer(3)._id33 { 1, iTimerOff[no]?, 1 }
job3.t1->job3.s2 { c >= d1 && glc <= best, machOff[m1]!, 1 } machine2.Busy->machine2._id28 { 1, machOff[no]?, busy[no] := 0 }
machine2._id28->machine2._id22 { 1, StepEnd!, 1 } job3.s2->job3.c2 { 1, StepEnd?, 1 }
machine2._id22->machine2.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(2)._id33->IdleTimer(2)._id32 { 1, iTimerOn[no]?, 1 }
job3.c2->job3.s2 { busy[m2], tau, 1 }
job0.t2->job0.s3 { c >= d2 && glc <= best, machOff[m2]!, 1 } machine0.Busy->machine0._id28 { 1, machOff[no]?, busy[no] := 0 }
machine0._id28->machine0._id22 { 1, StepEnd!, 1 } job0.s3->job0.c3 { 1, StepEnd?, 1 } job3.s2->job3.c2 { 1, StepEnd?, 1 }
machine0._id22->machine0.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(0)._id33->IdleTimer(0)._id32 { 1, iTimerOn[no]?, 1 }
job3.c2->job3.s2 { busy[m2], tau, 1 }
job0.c3->job0.t3 { 1, machOn[m3]!, c := 0 } machine2.Idle->machine2._id24 { 1, machOn[no]?, busy[no] := 1 }
machine2._id24->machine2.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(2)._id32->IdleTimer(2)._id33 { 1, iTimerOff[no]?, 1 }
machine2.Off->machine2.Busy { 1, tau, 1 }
job2.t2->job2.s3 { c >= d2 && glc <= best, machOff[m2]!, 1 } machine3.Busy->machine3._id28 { 1, machOff[no]?, busy[no] := 0 }
machine3._id28->machine3._id22 { 1, StepEnd!, 1 } job2.s3->job2.c3 { 1, StepEnd?, 1 } job3.s2->job3.c2 { 1, StepEnd?, 1 }
machine3._id22->machine3.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(3)._id33->IdleTimer(3)._id32 { 1, iTimerOn[no]?, 1 }
job2.c3->job2.s3 { busy[m3], tau, 1 }
job3.c2->job3.t2 { 1, machOn[m2]!, c := 0 } machine3.Idle->machine3._id24 { 1, machOn[no]?, busy[no] := 1 }
machine3._id24->machine3.Busy { cIdle <= startupCost, iTimerOff[no]!, 1 } IdleTimer(3)._id32->IdleTimer(3)._id33 { 1, iTimerOff[no]?, 1 }
job1.t2->job1.s3 { c >= d2 && glc <= best, machOff[m2]!, 1 } machine5.Busy->machine5._id28 { 1, machOff[no]?, busy[no] := 0 }
machine5._id28->machine5._id22 { 1, StepEnd!, 1 } job1.s3->job1.c3 { 1, StepEnd?, 1 } job2.s3->job2.c3 { 1, StepEnd?, 1 }
machine5._id22->machine5.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(5)._id33->IdleTimer(5)._id32 { 1, iTimerOn[no]?, 1 }
job1.c3->job1.t3 { 1, machOn[m3]!, c := 0 } machine0.Idle->machine0._id24 { 1, machOn[no]?, busy[no] := 1 }
machine0._id24->machine0.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(0)._id32->IdleTimer(0)._id33 { 1, iTimerOff[no]?, 1 }
machine0.Off->machine0.Busy { 1, tau, 1 }
job2.c3->job2.s3 { busy[m3], tau, 1 }
job0.t3->job0.s4 { c >= d3 && glc <= best, machOff[m3]!, 1 } machine2.Busy->machine2._id28 { 1, machOff[no]?, busy[no] := 0 }
machine2._id28->machine2._id22 { 1, StepEnd!, 1 } job0.s4->job0.c4 { 1, StepEnd?, 1 } job2.s3->job2.c3 { 1, StepEnd?, 1 }
machine2._id22->machine2.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(2)._id33->IdleTimer(2)._id32 { 1, iTimerOn[no]?, 1 }
job0.c4->job0.t4 { 1, machOn[m4]!, c := 0 } machine4.Idle->machine4._id24 { 1, machOn[no]?, busy[no] := 1 }
machine4._id24->machine4.Busy { cIdle <= startupCost, iTimerOff[no]!, 1 } IdleTimer(4)._id32->IdleTimer(4)._id33 { 1, iTimerOff[no]?, 1 }
job2.c3->job2.t3 { 1, machOn[m3]!, c := 0 } machine2.Idle->machine2._id24 { 1, machOn[no]?, busy[no] := 1 }
machine2._id24->machine2.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(2)._id32->IdleTimer(2)._id33 { 1, iTimerOff[no]?, 1 }

machine2.Off->machine2.Busy { 1, tau, 1 }
job3.t2->job3.s3 { c >= d2 && glc <= best, machOff[m2]!, 1 } machine3.Busy->machine3._id28 { 1, machOff[no]?, busy[no] := 0 }
machine3._id28->machine3._id22 { 1, StepEnd!, 1 } job3.s3->job3.c3 { 1, StepEnd?, 1 }
machine3._id22->machine3.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(3)._id33->IdleTimer(3)._id32 { 1, iTimerOn[no]?, 1 }
job3.c3->job3.t3 { 1, machOn[m3]!, c := 0 } machine1.Idle->machine1._id24 { 1, machOn[no]?, busy[no] := 1 }
machine1._id24->machine1.Busy { cIdle <= startupCost, iTimerOff[no]!, 1 } IdleTimer(1)._id32->IdleTimer(1)._id33 { 1, iTimerOff[no]?, 1 }
job0.t4->job0.s5 { c >= d4 && glc <= best, machOff[m4]!, 1 } machine4.Busy->machine4._id28 { 1, machOff[no]?, busy[no] := 0 }
machine4._id28->machine4._id22 { 1, StepEnd!, 1 } job0.s5->job0.c5 { 1, StepEnd?, 1 }
machine4._id22->machine4.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(4)._id33->IdleTimer(4)._id32 { 1, iTimerOn[no]?, 1 }
job3.t3->job3.s4 { c >= d3 && glc <= best, machOff[m3]!, 1 } machine1.Busy->machine1._id28 { 1, machOff[no]?, busy[no] := 0 }
machine1._id28->machine1._id22 { 1, StepEnd!, 1 } job3.s4->job3.c4 { 1, StepEnd?, 1 }
machine1._id22->machine1.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(1)._id33->IdleTimer(1)._id32 { 1, iTimerOn[no]?, 1 }
job3.c4->job3.t4 { 1, machOn[m4]!, c := 0 } machine4.Idle->machine4._id24 { 1, machOn[no]?, busy[no] := 1 }
machine4._id24->machine4.Busy { cIdle <= startupCost, iTimerOff[no]!, 1 } IdleTimer(4)._id32->IdleTimer(4)._id33 { 1, iTimerOff[no]?, 1 }
job0.c5->job0.t5 { 1, machOn[m5]!, c := 0 } machine5.Idle->machine5._id24 { 1, machOn[no]?, busy[no] := 1 }
machine5._id24->machine5.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(5)._id32->IdleTimer(5)._id33 { 1, iTimerOff[no]?, 1 }
machine5.Off->machine5.Busy { 1, tau, 1 }
job2.t3->job2.s4 { c >= d3 && glc <= best, machOff[m3]!, 1 } machine2.Busy->machine2._id28 { 1, machOff[no]?, busy[no] := 0 }
machine2._id28->machine2._id22 { 1, StepEnd!, 1 } job2.s4->job2.c4 { 1, StepEnd?, 1 }
machine2._id22->machine2.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(2)._id33->IdleTimer(2)._id32 { 1, iTimerOn[no]?, 1 }
job1.t3->job1.s4 { c >= d3 && glc <= best, machOff[m3]!, 1 } machine0.Busy->machine0._id28 { 1, machOff[no]?, busy[no] := 0 }
machine0._id28->machine0._id22 { 1, StepEnd!, 1 } job1.s4->job1.c4 { 1, StepEnd?, 1 }
machine0._id22->machine0.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(0)._id33->IdleTimer(0)._id32 { 1, iTimerOn[no]?, 1 }
job2.c4->job2.t4 { 1, machOn[m4]!, c := 0 } machine0.Idle->machine0._id24 { 1, machOn[no]?, busy[no] := 1 }
machine0._id24->machine0.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(0)._id32->IdleTimer(0)._id33 { 1, iTimerOff[no]?, 1 }
machine0.Off->machine0.Busy { 1, tau, 1 }
job1.c4->job1.t4 { 1, machOn[m4]!, c := 0 } machine2.Idle->machine2._id24 { 1, machOn[no]?, busy[no] := 1 }
machine2._id24->machine2.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(2)._id32->IdleTimer(2)._id33 { 1, iTimerOff[no]?, 1 }
machine2.Off->machine2.Busy { 1, tau, 1 }
job2.t4->job2.s5 { c >= d4 && glc <= best, machOff[m4]!, 1 } machine0.Busy->machine0._id28 { 1, machOff[no]?, busy[no] := 0 }
machine0._id28->machine0._id22 { 1, StepEnd!, 1 } job2.s5->job2.c5 { 1, StepEnd?, 1 }
machine0._id22->machine0.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(0)._id33->IdleTimer(0)._id32 { 1, iTimerOn[no]?, 1 }
job2.c5->job2.s5 { busy[m5], tau, 1 }
job1.t4->job1.s5 { c >= d4 && glc <= best, machOff[m4]!, 1 } machine2.Busy->machine2._id28 { 1, machOff[no]?, busy[no] := 0 }
machine2._id28->machine2._id22 { 1, StepEnd!, 1 } job1.s5->job1.c5 { 1, StepEnd?, 1 } job2.s5->job2.c5 { 1, StepEnd?, 1 }
machine2._id22->machine2.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(2)._id33->IdleTimer(2)._id32 { 1, iTimerOn[no]?, 1 }
job3.t4->job3.s5 { c >= d4 && glc <= best, machOff[m4]!, 1 } machine4.Busy->machine4._id28 { 1, machOff[no]?, busy[no] := 0 }
machine4._id28->machine4._id22 { 1, StepEnd!, 1 } job3.s5->job3.c5 { 1, StepEnd?, 1 }
machine4._id22->machine4.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(4)._id33->IdleTimer(4)._id32 { 1, iTimerOn[no]?, 1 }
job0.t5->job0._id16 { c >= d5 && glc <= best, machOff[m5]!, 1 } machine5.Busy->machine5._id28 { 1,

machOff[no]?, busy[no] := 0 }
machine5._id28->machine5._id22 { 1, StepEnd!, 1 }
machine5._id22->machine5.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(5)._id33->IdleTimer(5)._id32 { 1, iTimerOn[no]?, 1 }
job0._id16->job0._id1 { glt <= Target, tau, targetDeviation := Early * (Target - glt), cost += targetDeviation }
job0._id1->job0.end { 1, tau, 1 }
job1.c5->job1.t5 { 1, machOn[m5]!, c := 0 } machine4.Idle->machine4._id24 { 1, machOn[no]?, busy[no] := 1 }
machine4._id24->machine4.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(4)._id32->IdleTimer(4)._id33 { 1, iTimerOff[no]?, 1 }
machine4.Off->machine4.Busy { 1, tau, 1 }
job2.c5->job2.t5 { 1, machOn[m5]!, c := 0 } machine5.Idle->machine5._id24 { 1, machOn[no]?, busy[no] := 1 }
machine5._id24->machine5.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(5)._id32->IdleTimer(5)._id33 { 1, iTimerOff[no]?, 1 }
machine5.Off->machine5.Busy { 1, tau, 1 }
job3.c5->job3.t5 { 1, machOn[m5]!, c := 0 } machine0.Idle->machine0._id24 { 1, machOn[no]?, busy[no] := 1 }
machine0._id24->machine0.Off { cIdle > startupCost, iTimerOff[no]!, 1 } IdleTimer(0)._id32->IdleTimer(0)._id33 { 1, iTimerOff[no]?, 1 }
machine0.Off->machine0.Busy { 1, tau, 1 }
job3.t5->job3._id16 { c >= d5 && glc <= best, machOff[m5]!, 1 } machine0.Busy->machine0._id28 { 1, machOff[no]?, busy[no] := 0 }
machine0._id28->machine0._id22 { 1, StepEnd!, 1 }
machine0._id22->machine0.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(0)._id33->IdleTimer(0)._id32 { 1, iTimerOn[no]?, 1 }
job1.t5->job1._id16 { c >= d5 && glc <= best, machOff[m5]!, 1 } machine4.Busy->machine4._id28 { 1, machOff[no]?, busy[no] := 0 }
machine4._id28->machine4._id22 { 1, StepEnd!, 1 }
machine4._id22->machine4.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(4)._id33->IdleTimer(4)._id32 { 1, iTimerOn[no]?, 1 }
job1._id16->job1._id1 { glt <= Target, tau, targetDeviation := Early * (Target - glt), cost += targetDeviation }
job1._id1->job1.end { 1, tau, 1 }
job2.t5->job2._id16 { c >= d5 && glc <= best, machOff[m5]!, 1 } machine5.Busy->machine5._id28 { 1, machOff[no]?, busy[no] := 0 }
machine5._id28->machine5._id22 { 1, StepEnd!, 1 }
machine5._id22->machine5.Idle { 1, iTimerOn[no]!, 1 } IdleTimer(5)._id33->IdleTimer(5)._id32 { 1, iTimerOn[no]?, 1 }
job3._id16->job3._id1 { glt <= Target, tau, targetDeviation := Early * (Target - glt), cost += targetDeviation }
job3._id1->job3.end { 1, tau, 1 }
job2._id16->job2._id1 { glt <= Target, tau, targetDeviation := Early * (Target - glt), cost += targetDeviation }
job2._id1->job2.end { 1, tau, 1 }

Παράρτημα 2: Λίστα δημοσιεύσεων

Α. Δημοσιεύσεις σε διεθνή περιοδικά με κριτές

[1] Panopoulos D., Koussouris S., Kokkinakos P., Markaki O., Askounis D., Psarras J., (2013 – under review) A Knowledge-Enhanced Model-Based Approach for Managing Dynamic Manufacturing Networks, Journal of Intelligent Manufacturing, ISSN 0956-5515

[2] Kokkinakos, P., Koussouris, S., Panopoulos, D., Askounis, D., Ramfos, A., Georgousopoulos, C., Wittern, E. (2012) Citizens collaboration and co-creation in public service delivery: The COCKPIT project, International Journal of Electronic Government Research, 8 (3), pp. 44-62.

[3] Panopoulos, D., Sachpazidis, I., Rizou, D., Menary, W., Cardenas, J., Psarras, J. (2009) MEDNET: Telemedicine via satellite combining improved access to health-care services with enhanced social cohesion in rural Peru, Information Systems Development: Towards a Service Provision Society, pp. 933-940.

[4] Kirytopoulos, K., Voulgaridou, D., Panopoulos, D., Leopoulos, V. (2009) Project termination analysis in SMEs: Making the right call, International Journal of Management and Decision Making, 10 (1-2), pp. 69-90

[5] Zacharias, O., Panopoulos, D., Askounis, D.T. (2008) Large scale program risk analysis using a risk breakdown structure, European Journal of Economics, Finance and Administrative Sciences, (12), pp. 170-181

[6] Panopoulos, D., Metaxiotis, K. (2006) Solving production scheduling problems using advanced model checking tools, International Journal of Computer Applications in Technology, 26 (1-2), pp. 37-48

[7] Ptochos, D., Panopoulos, D., Metaxiotis, K., Askounis, D. (2004) Using internet GIS technology for early warning, response and controlling the quality of the public health sector, International journal of electronic healthcare, 1 (1), pp. 78-102

[8] D. Panopoulos, D. Ptochos, A. Oikonomou, D. Th, Askounis, J. Psarras (2003) Combining Neural Networks and CLP for Production Scheduling, WSEAS Transactions on Circuits and Systems, Issue 4, Vol. 2, pp. 826-831.

Β. Δημοσιεύσεις σε πρακτικά συνεδρίων

[1] Markaki, O., Kokkinakos, P., Panopoulos, D., Koussouris, S., Askounis, D. (2013) Benefits and risks in Dynamic Manufacturing Networks, IFIP Advances in Information and Communication Technology, 398 (PART 2), pp. 438-445

[2] Kokkinakos, P., Markaki, O., Panopoulos, D., Koussouris, S., Askounis, D. (2013) Dynamic Manufacturing Networks monitoring and governance, IFIP Advances in Information and Communication Technology, 398 (PART 2), pp. 446-453.

[3] Markaki, O., Panopoulos, D., Kokkinakos, P., Koussouris, S., Askounis, D. (2013) Towards adopting dynamic manufacturing networks for future

manufacturing: Benefits and risks of the IMAGINE DMN end-to-end management methodology, Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, art. no. 6570632, pp. 305-310.

[4] Georgiadou, A., Kokkinakos, P., Panopoulos, D., Koussouris, S., Askounis, D. (2013) A multicriteria methodology for the selection and prioritisation of public services, IFIP Advances in Information and Communication Technology, 399, pp. 325-337

[5] Bellos, Evangelos; Voulgaridou, Dimitra; Kirytopoulos, Konstantinos; Panopoulos, Dimitrios (2010) An MCDA Approach for Project Selection in Public Sector, PM-05 - Advancing Project Management for the 21st Century “Concepts, Tools & Techniques for Managing Successful Projects” 29-31 May 2010, Heraklion, Crete, Greece

ΣΥΝΟΠΤΙΚΟ ΒΙΟΓΡΑΦΙΚΟ ΣΗΜΕΙΩΜΑ

ΔΗΜΗΤΡΙΟΣ Π. ΠΑΝΟΠΟΥΛΟΣ

Ο κ. Δημήτριος Πανόπουλος γεννήθηκε στις 21 Φεβρουαρίου 1977 στην Αθήνα. Το 2001 έγινε δεκτός στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π., ως Υποψήφιος Διδάκτορας στο Εργαστήριο Συστημάτων Αποφάσεων και Διοίκησης, υπό την επίβλεψη του Καθηγητή κ. Ι.Ψαρρά. Ο κ. Πανόπουλος είναι κάτοχος Διπλώματος Μηχανολόγου Μηχανικού από το Ε.Μ.Π. και μεταπτυχιακού Διπλώματος Ειδίκευσης στα Τεχνο-οικονομικά Συστήματα στην κατεύθυνση της Διοίκησης Συστημάτων Παραγωγής.

Το ερευνητικό του έργο επικεντρώνεται στις περιοχές των Συστημάτων Παραγωγής, των Πληροφοριακών Συστημάτων, της Ηλεκτρονικής Διακυβέρνησης και της Επιχειρησιακής Αναδιοργάνωσης. Έχει συμμετάσχει σε πλήθος ερευνητικών έργων χρηματοδοτούμενων τόσο από τη ΓΓΕΤ όσο και από την Ε.Ε. στο πλαίσιο των προγραμμάτων FP6 και FP7.

Στις παραπάνω περιοχές ο κ. Πανόπουλος διαθέτει επίσης σημαντική επαγγελματική εμπειρία παρέχοντας συμβουλευτικές υπηρεσίες σε οργανισμούς και επιχειρήσεις.

Κατά τη διάρκεια εκπόνησης της διδακτορικής του διατριβής, ο κ. Πανόπουλος παρείχε επικουρική διδασκαλία στα εξής μαθήματα:

- Διοίκηση Παραγωγής & Συστημάτων Υπηρεσιών (Προπτυχιακό ΣΗΜΜΥ): 2002-2007
- Παίγνια Αποφάσεων (Προπτυχιακό ΣΗΜΜΥ): 2002 – 2013
- Διοίκηση Παραγωγής (Μεταπτυχιακό Τεχνο-οικονομικά Συστήματα): 2007-2013

Επιπρόσθετα έχει συμμετάσχει στην επίβλεψη 32 διπλωματικών εργασιών για το προπτυχιακό πρόγραμμα της ΣΗΜΜΥ και για το διατμηματικό μεταπτυχιακό πρόγραμμα «Τεχνοοικονομικά Συστήματα», με επιβλέποντες καθηγητές τους κ. Ιωάννη Ψαρρά (Καθ.ΕΜΠ) και Δημήτριο Ασκούνη (Αναπλ.Καθ.ΕΜΠ).