**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**
**ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ**
**ΤΟΜΕΑΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΕΠΙΧΕΙΡΗΣΙΑΚΗΣ ΕΡΕΥΝΑΣ**


**Διπλωματική Εργασία**

# Χρονοπρογραμματισμός Πολλαπλών Έργων με Περιορισμένους Πόρους και Πολλαπλούς Τρόπους Εκτέλεσης με τη Μέθοδο Βελτιστοποίησης Αποικίας Μυρμηγκιών

Δερμιτζάκης Εμμανουήλ

Υπεύθυνος Καθηγητής: Κωνσταντίνος Κηρυττόπουλος,
Επίκουρος Καθηγητής Σχολής Μηχανολόγων Μηχανικών ΕΜΠ

**Αθήνα, 2013**

# Ευχαριστίες

Με την παρούσα διπλωματική εργασία ολοκληρώνονται οι σπουδές μου στη σχολή των Μηχανολόγων Μηχανικών. Θέλω να ευχαριστήσω ιδιαίτερα τον επιβλέποντα καθηγητή μου, κύριο Κυρηττόπουλο Κωνσταντίνο για την άριστη συνεργασία και υποστήριξη όποτε κι αν το χρειαζόμουν. Επίσης την Δρ. Ρόκου Έλενα για την βοήθεια, την καθοδήγηση, την υπομονή και την εμπιστοσύνη που μου έδειξε κατά τη διάρκεια εκπόνησης της παρούσας εργασίας, της οποίας η ολοκλήρωση θα ήταν αδύνατη χωρίς τη βοήθειά της. Με την ευκαιρία που μου δίνεται θέλω ακόμη να ευχαριστήσω όλους τους συμφοιτητές και φίλους που με στήριξαν κατά τη διάρκεια των σπουδών μου και όλους τους καθηγητές μου, οι οποίοι με βοήθησαν να διευρύνω τους ορίζοντες μου, τόσο σε γνωσιακό όσο και σε ηθικοπνευματικό και πολιτισμικό επίπεδο. Τέλος, θέλω να ευχαριστήσω την οικογένειά μου για όλα όσα μου έχει προσφέρει όλα αυτά τα χρόνια, την αγάπη και την αυτοθυσία τους.

# Περιεχόμενα

**Κατάλογος Πινάκων**

# Κατάλογος Σχημάτων

# Κατάλογος Αλγορίθμων

***Έχω διαβάσει και κατανοήσει τους κανόνες για τη λογοκλοπή και τον τρόπο σωστής αναφοράς των πηγών που περιέχονται στον Οδηγό συγγραφής Διπλωματικών εργασιών. Δηλώνω ότι, από όσα γνωρίζω, το περιεχόμενο της παρούσας Διπλωματικής εργασίας είναι προϊόν δικής μου δουλειάς και υπάρχουν αναφορές σε όλες τις πηγές που χρησιμοποίησα.***

***Δερμιτζάκης Εμμανουήλ***

# Έποψη

Στην παρούσα διπλωματική εργασία παρουσιάζεται ένας εξελικτικός αλγόριθμος για τον προγραμματισμό πολλαπλών έργων, με πολλαπλούς τρόπους εκτέλεσης και περιορισμένους πόρους (Multi-project M-RCPSP), βασισμένος στην βελτιστοποίηση με αποικία μυρμηγκιών (Ant Colony Optimization - ACO).

Η δομή του προβλήματος αποτελείται από ένα σύνολο παράλληλων έργων και ένα σύνολο κοινών για όλα τα έργα πόρων, που περιέχει ανανεώσιμους και μη-ανανεώσιμους πόρους. Οι πόροι έχουν περιορισμένη διαθεσιμότητα ανά περίοδο. Τα έργα μπορεί να είναι ανεξάρτητα μεταξύ τους ή και όχι. Κάθε έργο έχει ένα σύνολο δραστηριοτήτων, οι οποίες μπορούν να εκτελεστούν με έναν ή περισσότερους τρόπους, όπου κάθε τρόπος αντιστοιχεί σε διαφορετικό τύπο και απαιτούμενη ποσότητα πόρων και οδηγεί σε διαφορετική διάρκεια δραστηριότητας.

Η συνήθης μέθοδος όσον αφορά το πρόβλημα προγραμματισμού πολλαπλών έργων είναι η μετατροπή του προβλήματος πολλαπλών έργων σε πρόβλημα ενός έργου μέσω συγχώνευσης των δικτύων των έργων. Η μέθοδος αυτή μπορεί να οδηγήσει σε δυσεπίλυτα δίκτυα των οποίων η επίλυση μπορεί να είναι ιδιαίτερα χρονοβόρα. Εδώ παρουσιάζεται μια νέα μέθοδος, βασισμένη στη βελτιστοποίηση με αποικία μυρμηγκιών (ACO), η οποία διαφοροποιείται με βάση την ύπαρξη ή όχι συσχετίσεων μεταξύ μιας ή περισσότερων δραστηριοτήτων ενός έργου με ένα ή περισσότερα άλλα έργα. Έτσι, παράγονται δύο σύνολα έργων, ένα σύνολο ανεξάρτητων έργων και ένα σύνολο έργων με εξωτερικές εξαρτήσεις. Το σύνολο των έργων με εξωτερικές εξαρτήσεις διαιρείται σε υποσύνολα με βάση τον αριθμό των εξαρτήσεων, με σκοπό τη δημιουργία δύο ή τριών υποσυνόλων: υψηλής, μεσαίας και χαμηλής εξάρτησης. Τα έργα έπειτα προγραμματίζονται σύμφωνα με το σύνολο στο οποίο ανήκουν. Τα έργα που ανήκουν στο σύνολο υψηλής εξάρτησης και έχουν υψηλότερες απαιτήσεις σε πόρους κατηγοριοποιούνται και προγραμματίζονται πρώτα, ακολουθούν εκείνα που έχουν ακόμη υψηλό βαθμό εξωτερικών εξαρτήσεων αλλά χαμηλότερες απαιτήσεις σε πόρους και ούτω καθεξής. Η διαδικασία προγραμματισμού χρησιμοποιεί έναν Διαχειριστή (Moderator) ACO που χειρίζεται τη σειρά προγραμματισμού και συνακόλουθα την κατηγοριοποίηση των συνόλων και την τυχαιοποίηση των έργων του ιδίου συνόλου, και την εκχώρηση των τρόπων εκτέλεσης των δραστηριοτήτων και έναν δεύτερο ACO αλγόριθμο για το RCPSP.

Τα σύνολα δεδομένων πολλαπλών τρόπων (multi-mode data sets) της διαδικτυακής βιβλιοθήκης PSP-Lib συνδυάστηκαν για να δώσουν μια ποικιλία συνόλων πολλαπλών έργων και χρησιμοποιήθηκαν για να αποδειχθεί η αποδοτικότητα και η αποτελεσματικότητα της προτεινόμενης προσέγγισης.

## Synopsis

An evolutionary algorithm for the multi-project, multi-mode resource constrained project scheduling (M-RCPSP), based on the combination of ant colony optimization (ACO) and genetic algorithm (GA), is proposed.

The problem setting consists of several parallel projects and a number of shared resource pools containing renewable and non-renewable resources. All the resources are supplied limitedly. The projects can be independent with each other or not. Each project has a number of activities that can be executed in one or more modes (multi-mode) where each mode corresponds to different resource type and amount requirements and leads to different activity duration.

The common method about the multi-project scheduling problem is converting the multiple projects problem into a single project problem through the merger of a number of network plans. This method can lead to cumbersome networks that their solution can be very time consuming. In this paper a new method based on the combination of ant colony optimization (ACO) and genetic algorithm is proposed.

The process is differentiated based on the existence of relations between one or more activities of a project with the activities of one or more other projects or not. This way two project sets are generated, a set of independent projects and a set of projects with outer dependencies. The set of projects with outer dependencies is divided in subsets based on the number of outer dependencies in order to form two or three subsets: of low, medium and high dependency. The projects are then ordered and scheduled based on the set that they belong. This way, all the projects that belong to the high dependency set and have higher resource requirements will be prioritized and scheduled up front, then those that still have high degree of outer dependencies but low resource requirements and so on. The scheduling process uses an adapted ACO algorithm for the M-RCPSP scheduling and the genetic algorithm handles the scheduling order and therefore the prioritization of the sets and the randomization of the projects in the same set.

The multi-mode PSP Lib data sets were combined to give a variety of multi-project data sets and used to prove the efficiency and effectiveness of the proposed approach.

# 1. Εισαγωγή

Ο χρονικός προγραμματισμός των διεργασιών ενός έργου αποτελεί έναν από τους σημαντικότερους τομείς της διοίκησης έργων. Κατά κανόνα περιλαμβάνει διαδικασίες καθορισμού των δραστηριοτήτων που πρέπει να εκτελεστούν ώστε να ολοκληρωθεί το έργο, καθορισμού της σειράς με την οποία πρέπει αυτές να γίνουν, εκτίμησης και προσδιορισμού των απαιτούμενων πόρων για την πραγματοποίηση κάθε μιας από αυτές, εκτίμησης του απαιτούμενου χρόνου με βάση τους διαθέσιμους πόρους και τέλος τις διαδικασίες κατασκευής και παρακολούθησης του επακόλουθου χρονικού προγράμματος. Στόχος της παρούσας διπλωματικής εργασίας είναι ο χρονικός προγραμματισμός πολλαπλών παράλληλων έργων και η βελτιστοποίηση των παραγόμενων χρονοπρογραμμάτων, λαμβάνοντας υπόψη τους περιορισμούς όσον αφορά τη διαθεσιμότητα των πόρων και τις σχέσεις προτεραιότητας μεταξύ δραστηριοτήτων του ίδιου έργου ή και διαφορετικών έργων. Πιο αναλυτικά, αποσκοπείται η ελαχιστοποίηση της διάρκειας του συνόλου των έργων αλλά και κάθε έργου ξεχωριστά, σύμφωνα με τις προτεραιότητες για την εκτέλεση των δραστηριοτήτων και τους περιορισμούς των διαθέσιμων πόρων.

Ένα έργο είναι ένα σύνολο από δραστηριότητες το οποίο έχει σαφώς καθορισμένη αρχή και συγκεκριμένο τέλος και στοχεύει στην επίτευξη ενός ορισμένου σκοπού κάνοντας χρηση καθορισμένων πόρων. Ένα χρονοπρόγραμμα συνήθως αποσκοπεί στον προγραμματισμό των δραστηριοτήτων του έργου, δηλαδή στον καθορισμό της έναρξής τους, έτσι ώστε να ικανοποιούνται οι σχέσεις προτεραιότητας και οι χρησιμοποιούμενοι πόροι να μην υπερβαίνουν την διατιθέμενη ποσότητα. Βασικά συστατικά ενός χρονοπρογράμματος είναι τα ακόλουθα:

- Δραστηριότητες με χαρακτηριστικά γνωρίσματα το αναγνωριστικό (ID) και τον τρόπο εκτέλεσης κάθε μίας, ο οποίος αφορά τη διάρκεια της δραστηριότητας και το είδος και την ποσότητα πόρων που απαιτούνται για την εκτέλεσή της.
- Σχέσεις προτεραιότητας που καθορίζουν ποιές δραστηριότητες πρέπει να ολοκληρωθούν προτού η υπό εξέταση δραστηριότητα μπορεί να αρχίσει να εκτελείται.
- Πόροι, οι οποίοι χαρακτηρίζονται από τη λειτουργική τους χρήση και τη διαθέσιμη ποσότητα. Μπορούν να είναι δύο ειδών: α) ανανεώσιμοι, δηλαδή να έχουν περιορισμένη διαθεσιμότητα ανά περίοδο του έργου (πχ. εργάτες) β) μη ανανεώσιμοι, δηλαδή να έχουν συγκεκριμένη ποσότητα διαθέσιμη για το σύνολο του έργου (πχ. προϋπολογισμός έργου) .

Το πρόβλημα του χρονικού προγραμματισμού έργων υπό περιορισμένους πόρους, το οποίο αναφέρεται στη βιβλιογραφία ως Resource Constrained Project Scheduling Problem ή εν συντομία με το ακρονύμιο RCPSP, ορίζεται ως *«η μέθοδος προγραμματισμού δραστηριοτήτων δοθέντων συγκεκριμένων ποσοτήτων διαθέσιμων πόρων για κάθε περίοδο της διάρκειας του έργου με στόχο την ελαχιστοποίηση της συνολικής διάρκειάς του»* (Davis, 1973). Το πρόβλημα αυτό έχει αποδειχθεί ότι είναι πρόβλημα τύπου δυσκολίας NP-hard, και έχει μελετηθεί εκτενώς στα πλαίσια της επιστήμης της επιχειρησιακής έρευνας, δεδομένου του ενδιαφέροντος και της κρισιμότητάς του για την επιτυχή οργάνωση, εκτέλεση και ολοκλήρωση ενός έργου. Με το πέρασμα των χρόνων και την όλο και μεγαλύτερη ενασχόληση των ερευνητών με το συγκεκριμένο πρόβλημα, αναπτύχθηκαν πολλές επεκτάσεις του, με στόχο την όσο το δυνατόν πιο ρεαλιστική απεικόνιση της πραγματικότητας στην μοντελοποίηση του προβλήματος. Στο πλαίσιο αυτό, στην παρούσα διπλωματική αντιμετωπίζεται η επέκταση του RCPSP γνωστή ως Multi-project Multi-mode RCPSP. Είναι ουσιαστικά διπλή επέκταση του αρχικού προβλήματος, όπως απεικονίζεται και στο Σχήμα 1.1: Πρώτον, στόχος δεν είναι ο προγραμματισμός ενός έργου αλλά ενός συνόλου έργων, ανεξάρτητων ή και εξαρτημένων μεταξύ τους, που πρέπει να προγραμματιστούν παράλληλα (Multi-project) και δεύτερον, οι δραστηριότητες κάθε έργου μπορούν να εκτελεστούν με έναν ή περισσότερους τρόπους (Multi-mode). Ακόμη υπάρχει ένα σύνολο κοινών για όλα τα έργα πόρων, που περιέχει ανανεώσιμους και μη-ανανεώσιμους πόρους, των οποίων η διαθεσιμότητα είναι περιορισμένη ανά περίοδο ή για το σύνολο της διάρκειας όλων των έργων, αντίστοιχα.


**Σχήμα 1.1 Σχηματική αναπαράσταση του προς επίλυση προβλήματος**

Για την επίλυση του προβλήματος πολλαπλών έργων η συνήθης μέθοδος είναι να μετατρέπεται το πρόβλημα των πολλαπλών έργων σε πρόβλημα ενός έργου, με συγχώνευση των δικτύων των έργων. Πραγματοποιείται δηλαδή μια μοντελοποίηση του προβλήματος ως ένα, πρακτικά υπερμεγέθες RCPSP, που λόγω της NP-δυσκολίας του πρωτότυπου προβλήματος, έχει σαν αποτέλεσμα την δημιουργία πολύπλοκων, δυσεπίλυτων και δυσκίνητων δικτύων, των οποίων η επίλυση είναι ιδιαίτερα χρονοβόρα και πιθανά μη αποδοτική. Εδώ παρουσιάζεται μια

νέα μέθοδος, βασισμένη στην κατηγοριοποίηση των έργων σε σύνολα, σύμφωνα με τις αλληλοεξαρτήσεις μεταξύ τους, την έπειτα φθίνουσα κατάταξη τους εντός των δημιουργημένων συνόλων σύμφωνα με τις απαιτήσεις τους σε πόρους, και τελικά τον προγραμματισμό και εκτέλεσή τους με βάση αυτή την προκαθορισμένη, διπλού επιπέδου ιεράρχηση. Για την επίλυση του προβλήματος χρησιμοποιήθηκε η μέθοδος των αλγορίθμων Βελτιστοποίησης με Αποικία Μυρμηγκιών (Ant Colony Optimization), οι οποίοι ανήκουν στους Εξελικτικούς Αλγόριθμους (Evolutionary Algorithms) και παράλληλα είναι βασισμένοι στην Τεχνητή Νοημοσύνη (Artificial Intelligence) και ειδικότερα στη Νοημοσύνη των Σμηνών (Swarm Intelligence). Παρουσιάστηκαν πρώτη φορά το 1992 από τον Marco Dorigo στο πλαίσιο της διδακτορικής του διατριβής. Έκτοτε έχει αναπτυχθεί και παρουσιαστεί πλήθος εξελίξεών τους και έχουν εφαρμοσθεί σε πολλά προβλήματα Συνδυαστικής Βελτιστοποίησης (Combinatorial Optimization), με ιδιαίτερα ενθαρρυντικά αποτελέσματα.

Η διπλωματική αυτή εργασία περιλαμβάνει ένα αρχικό κεφάλαιο όπου εισάγονται τα βασικά στοιχεία του περιεχομένου της και στη συνέχεια διαρθρώνεται ως εξής:

Στο Κεφάλαιο 2 παρουσιάζεται η μέθοδος έρευνας που ακολουθήθηκε για την προσέγγιση του προβλήματος.

Στο Κεφάλαιο 3 τοποθετείται το πρόβλημα σε σχέση με τη βιβλιογραφία και γίνεται ανάλυση όλων των στοιχείων με τα οποία σχετίζεται το αντικείμενο της παρούσας εργασίας.

Στο Κεφάλαιο 4 γίνεται η προσέγγιση του προβλήματος, όπου περιγράφεται το ευρύτερο πλαίσιο του προς επίλυση προβλήματος, ορίζονται οι στόχοι του και ο τρόπος που αυτοί επηρεάζονται από το περιβάλλον του προβλήματος και αναλύεται η δομή των στοιχείων που απαρτίζουν το πρόβλημα.

Στο Κεφάλαιο 5 παρουσιάζεται η εννοιολογική και μαθηματική μοντελοποίηση του προβλήματος.

Στο Κεφάλαιο 6 γίνεται αναλυτική περιγραφή της προτεινόμενης διαδικασίας επίλυσης, καθώς και της δομής και λειτουργίας του αλγορίθμου επίλυσης του προβλήματος. Στο πρώτο υποκεφάλαιο γίνεται επισκόπηση του αλγορίθμου βελτιστοποίησης με αποικία μυρμηγκιών και ανάλυση των χαρακτηριστικών του. Στο δεύτερο υποκεφάλαιο παρουσιάζεται η διαδικασία επίλυσης και ο τρόπος λειτουργίας του αλγορίθμου.

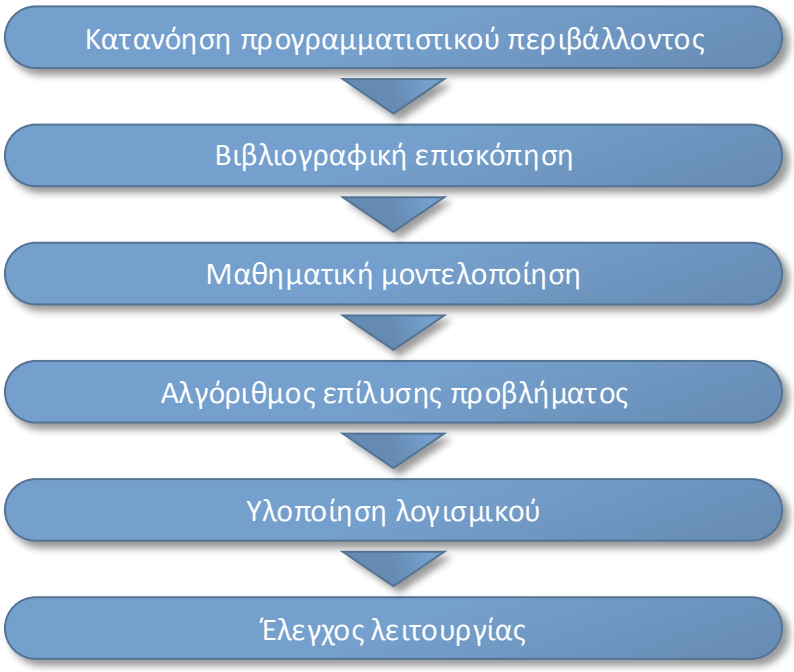
Στο Κεφάλαιο 7 παρουσιάζονται οι πειραματικές διατάξεις και τα αποτελέσματα που επιτεύχθηκαν με χρήση του προτεινόμενου μοντέλου και

αλγορίθμου και συγκρίνονται με τις καλύτερες λύσεις, όπως αυτές προκύπτουν από τη βιβλιογραφία.

Τέλος, στο Κεφάλαιο 8 γίνεται παρουσίαση των συμπερασμάτων που προέκυψαν από την έρευνα, αξιολόγηση του προτεινόμενου αλγορίθμου και προτάσεις για περεταίρω ερευνητική μελέτη.

## 2. Μέθοδος έρευνας

Η διαδικασία που ακολουθήθηκε κατά τη διάρκεια εκπόνησης της εργασίας εμφανίζονται στο Σχήμα 2.1. Αρχικά έγινε μελέτη και κατανόηση του προγραμματιστικού περιβάλλοντος, στο οποίο κατασκευάστηκε το πρόγραμμα του αλγορίθμου επίλυσης του προβλήματος. Μετά την εμπέδωση του προγραμματιστικού περιβάλλοντος, ακολούθησε βιβλιογραφική επισκόπηση επί του θέματος του προγραμματισμού έργων, με μεγαλύτερη βαρύτητα στο θέμα της διπλωματικής εργασίας. Συγκεκριμένα, εντοπίστηκε το υπό μελέτη πρόβλημα, οι προτεινόμενες λυσεις στη βιβλιογραφία, καθώς και συγκριτική μελέτη των υφιστάμενων τρόπων μοντελοποίησης των αλγορίθμων επίλυσης. Έπειτα ορίστηκε αρχικά το εννοιολογικό μοντέλο του προβλήματος και εν συνεχεία μοντελοποιήθηκε μαθηματικά. Μετά από τα παραπάνω βήματα, σχεδιάστηκε ο αλγόριθμος επίλυσης του προβλήματος. Τέλος, ύστερα από την υλοποίηση του μαθηματικού μοντέλου και του αλγορίθμου επίλυσης, ένα σύνολο από δεδομένα διαθέσιμα στη βιβλιογραφία χρησιμοποιήθηκε για να ρυθμιστεί και να βελτιωθεί περεταίρω ο υλοποιημένος αλγόριθμος και έπειτα για να αποδειχθεί η εγκυρότητα και η αποδοτικότητά του.




**Σχήμα 2.1 Βήματα εργασίας**

# 3. Βιβλιογραφική Επισκόπηση

## 3.1 Διοίκηση Έργων

Ο τομέας της διοίκησης έργων έχει κάνει ιδιαίτερα μεγάλα βήματα προς τα εμπρός τις τελευταίες δεκαετίες. Στο ανταγωνιστικό περιβάλλον του σήμερα είναι ζωτικής σημασίας να παραδίδει κανείς ποιοτικά προϊόντα εμπρόθεσμα και εντός προϋπολογισμού. Συνεπώς δεν πρέπει να προκαλεί έκπληξη το γεγονός ότι η διοίκηση έργων έχει γίνει ένα τόσο φλέγον ζήτημα.

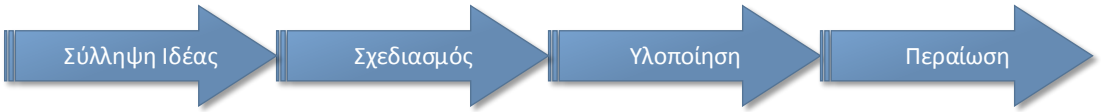
Η λέξη “έργο” σήμερα χρησιμοποιείται πολύ συχνά και έχει διαφορετικό νόημα για διαφορετικούς ανθρώπους, συναρτήσει του πλαισίου στο οποίο τοποθετείται κάθε φορά. Ένας γενικά αποδεκτός ορισμός έχει δοθεί από το ISO όπου ως “έργο” ορίζεται *«μια μοναδική διαδικασία, αποτελούμενη από ένα σύνολο από συντονισμένες και ελεγχόμενες δραστηριότητες με ημερομηνίες έναρξης και λήξης, που έχει αναληφθεί για να επιτύχει ένα στόχο συμβιβαζόμενη σε συγκεκριμένες απαιτήσεις συμπεριλαμβανομένων περιορισμών σε χρόνο, κόστος και πόρους»* (ISO, 2012). Ένας άλλος ορισμός δίνεται από το Project Management Book of Knowledge όπου ένα έργο ορίζεται ως *«ένα προσωρινό εγχείρημα με σκοπό τη δημιουργία ενός μοναδικού προϊόντος , μιας υπηρεσίας ή ενός αποτελέσματος»* (PMI, 2012).

Αναλύοντας τον ορισμό του έργου, η προσωρινή του φύση υποδεικνύεται από την ύπαρξη καθορισμένης αρχής και τέλους του. Το τέλος σηματοδοτείται όταν οι στόχοι του έχουν επιτευχθεί ή όταν τερματίζεται γιατί οι στόχοι του δεν μπορούν ή δε θα μπορούν στο μέλλον να επιτευχθούν ή γιατί η ανάγκη για το έργο δεν υπάρχει πιά. Η διάρκεια του είναι πάντοτε πεπερασμένη και καθορίζεται από το χρονικό διάστημα μεταξύ αρχής και τέλους του. Ακόμη, η προσωρινή φύση του έργου υποδεικνύει οτι μια συγκεντρωμένη χρήση πόρων θα χρειαστεί για την εκτέλεσή του. Η μοναδική φύση των έργων εκφράζει το γεγονός οτι κάθε έργο δημιουργεί ένα μοναδικό προϊόν, υπηρεσία ή αποτέλεσμα που το διαφοροποιεί από άλλα προϊόντα, υπηρεσίες ή αποτελέσματα (PMI, 2012). Επιτυχημένο έργο είναι εκείνο που τελειώνει εμπρόθεσμα, εντός προϋπολογισμού και σύμφωνα με τις αρχικες προδιαγραφές. Συνοψίζοντας, όλα τα έργα έχουν τα παρακάτω κοινά στοιχεία:

- Ενα στόχο ή επιδίωξη: Ένα ορισμένο τελικό προϊόν, αποτέλεσμα ή έξοδο που ορίζεται συνήθως ως προς το κόστος, την ποιότητα και το χρόνο εξαγωγής του από τις δραστηριότητες του έργου.
- Μοναδικότητα: Ένα έργο γίνεται μία φορά και δεν επαναλαμβάνεται. Αν το τελικό προϊόν, η υπηρεσία ή το αποτέλεσμα ενός έργου έχει επαναληφθεί σε άλλα έργα ή ακόμα και αν το προϊόν, η υπηρεσία ή το αποτέλεσμα του ίδιου του έργου είναι

επαναλαμβανόμενο, αυτές οι επαναλήψεις δεν αλλάζουν τη μοναδικότητα των εργασιών του έργου, οι οποίες μπορεί να έχουν ειδοποιές διαφορές όσον αφορά τους απαιτούμενους πόρους, τους τρόπους εκτέλεσης ή και το ίδιο το περιβάλλον στο οποίο το έργο πραγματοποιείται.

- Πολυπλοκότητα: Οι σχέσεις μεταξύ των διαφόρων εργασιών που πρέπει να πραγματοποιηθούν για να επιτευχθούν οι στόχοι του έργου μπορούν να χαρακτηρίζονται από μεγάλο βαθμό πολυπλοκότητας.
- Προσωρινή φύση: Τα έργα έχουν καθορισμένους χρόνους έναρξης και λήξης, κάτι το οποίο συνήθως σημαίνει οτι μια συγκεντρωμένη χρήση πόρων απαιτείται για την πραγματοποίηση τους.
- Αβεβαιότητα: Τα έργα προγραμματίζονται πριν εκτελεστούν και σαν επακόλουθο αυτού ενέχουν το στοιχείο του κινδύνου.
- Κύκλο ζωής: Ένα έργο διανύει ένα κύκλο ζωής που αποτελείται από διαφορετικές φάσεις. Ξεκινά με την σύλληψη της ιδέας του έργου, κατά την οποία ο οργανισμός συνειδητοποιεί οτι μπορεί να χρειάζεται ένα έργο ή δέχεται αίτημα από ένα πελάτη για να προτείνει ένα πλάνο για την πραγματοποίηση ενός έργου. Έπειτα έρχεται η φάση του ορισμού, στην διάρκεια της οποίας οι επιδιώξεις, οι στόχοι και το περιεχόμενο του έργου ορίζονται και αποφασίζεται ο τρόπος με τον οποίο ο οργανισμός θα πετύχει τους στόχους και θα ανταπεξέλθει στα διάφορα μέτρα απόδοσης. Όταν το έργο καθοριστεί και εγκριθεί, ξεκινά η φάση της οργάνωσης που αφορά τη διαίρεση του έργου σε διαχειρίσιμα πακέτα εργασίας που αποτελούνται από συγκεκριμένες δραστηριότητες που πρέπει να εκτελεστούν προκειμένου να πραγματοποιηθούν οι στόχοι του έργου. Πρέπει να εκτιμηθούν οι διάρκειες των δραστηριοτήτων, και να καθοριστούν με επαρκή ακρίβεια τα είδη και οι ποσότητες των αναγκαίων πόρων για την εκτέλεση κάθε δραστηριότητας, καθώς και οι σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων. Το έργο τότε εισέρχεται στην φάση του προγραμματισμού, που περιλαμβάνει την κατασκευή ενός εφικτού απο άποψη προτεραιότητας δραστηριοτήτων και χρήσης πόρων βασικού σχεδίου ή χρονοπρογράμματος, που ορίζει τους χρόνους έναρξης και ολοκλήρωσης κάθε μιας δραστηριότητας. Το έργο τότε πρέπει να πραγματοποιηθεί. Κατά την εκτέλεση του έργου, η πρόοδός του πρέπει να καταγράφεται και πρέπει να γίνονται διορθωτικές κινήσεις όταν είναι αναγκαίο. Τελικά, το έργο μπαίνει στην φάση τερματισμού του, όπου και γίνεται η παράδοση του αποτελέσματος του έργου (προϊόντων ή/και υπηρεσιών).




**Σχήμα 3.1 Κύκλος ζωής έργου (PMI, 2012)**

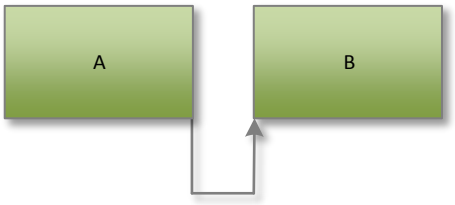
Ένα έργο αποτελείται από ένα σύνολο συμβάντων (events/milestones) και δραστηριοτήτων που πρέπει να εκτελεστούν, σύμφωνα με ένα σύνολο σχέσεων προτεραιότητας. Κάθε συμβάν αναφέρεται σε ένα στάδιο ολοκλήρωσης δραστηριοτήτων, συνδέεται με ένα συγκεκριμένο χρονικό σημείο και έχει μηδενική διάρκεια και απαιτήσεις σε πόρους. Κάθε δραστηριότητα έχει χρονική διάρκεια και απαιτεί συγκεκριμένη ποσότητα από έναν ή περισσότερους πόρους για να εκτελεστεί.

Η *διάρκεια* (*duration*) κάθε δραστηριότητας μπορεί να είναι συγκεκριμένη ή στοχαστική. Στην πρώτη περίπτωση η διάρκεια έχει μια σταθερή τιμή, που υπολογίζεται εκτιμώντας τον μέσο χρόνο που χρειάζεται για να ολοκληρωθεί, εξαιρώντας μη ελεγχόμενα απρόοπτα ή τη στοχαστικότητα. Σε πολύ ειδικές περιπτώσεις, που ενέχεται ο κίνδυνος μάλλον ανακρίβειας παρά αβεβαιότητας, η διάρκεια των δραστηριοτήτων μπορεί να εκφραστεί με χρήση ασαφών αριθμών (fuzzy numbers).

Μια δραστηριότητα μπορεί να είναι *πολλαπλών τρόπων εκτέλεσης* (*multi-mode*), δηλαδή μπορεί να εκτελεστεί με αρκετούς, ξεκάθαρα διαφορετικούς τρόπους εκτέλεσης, με κάθε έναν να ορίζει διαφορετική διάρκεια, είδος και ποσότητα απαιτούμενων πόρων ή να έχει μια διάρκεια και ένα σύνολο απαιτήσεων σε πόρους και τότε καλείται *απλού τρόπου εκτέλεσης* (*single-mode*).

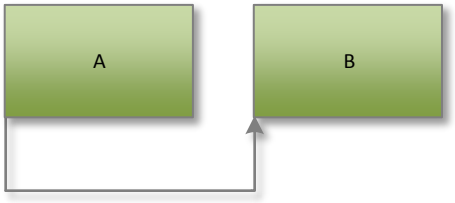
Μεταξύ των δραστηριοτήτων αναπτύσσονται τεσσάρων ειδών *σχέσεις προτεραιότητας:*

*Σχέση τέλους - αρχής (finish to start, FS)*: Δηλώνει οτι η έναρξη της εκτέλεσης μιας δραστηριότητας προαπαιτεί την ολοκλήρωση των δραστηριοτήτων που προηγούνται αυτής στη σειρά προτεραιότητας. Δηλαδή, η αρχή μιας δραστηριότητας καθορίζεται από το τέλος των προκατόχων (predecessors) της. Όταν η χρονική καθυστέρηση (lag) μεταξύ δραστηριοτήτων που συνδέονται με σχέση τέλους-αρχής είναι μηδενική, η σχέση αυτή ταυτίζεται με τη σχέση δραστηριοτήτων στη μέθοδο του κρίσιμου δρόμου (CPM).



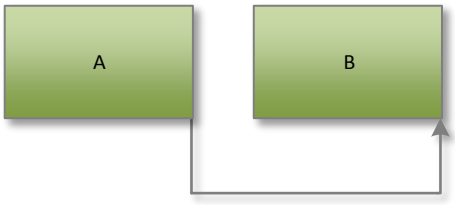

**Σχήμα 3.2 Σχέση τέλους – αρχής (FS)**

*Σχέση αρχής - αρχής (start to start, SS)*: Δηλώνει οτι αν δύο δραστηριότητες συνδέονται με αυτή τη σχέση, η δραστηριότητα που ακολουθεί μπορεί να ξεκινήσει μόνο μετά την έναρξη εκείνης που προηγείται. Με τη σχέση αυτή προσπαθεί να αποδοθεί το πρακτικό φαινόμενο οι απαιτήσεις και οι περιορισμοί για την εκκίνηση μιας δραστηριότητας να πληρούνται πριν την ολοκλήρωση του συνόλου των προκατόχων της.




**Σχήμα 3.3 Σχέση αρχής - αρχής (SS)**

*Σχέση τέλους - τέλους (finish to finish, FF)*: Σύμφωνα με αυτή τη σχέση, η ολοκλήρωση μιας δραστηριότητας προϋποθέτει την ολοκλήρωση εκείνων των δραστηριοτήτων που προηγούνται αυτής. Κατά αυτόν τον τρόπο επιτρέπεται η ταυτόχρονη εκτέλεση των δύο δραστηριοτήτων ή ακόμα και η εκκίνηση μιας δραστηριότητας πριν από την εκκίνηση προκατόχου της, εφόσον δεν παραβιάζεται η συνθήκη που ορίζει η σχέση τέλους-τέλους.




**Σχήμα 3.4 Σχέση τέλους – τέλους (FF)**

*Σχέση αρχής – τέλους (start to finish, SF)*: Δηλώνει οτι αν δύο δραστηριότητες συνδέονται με σχέση αρχής – τέλους τότε εκείνη που ακολουθεί μπορεί να τελειώσει

μόνο αν έχει ξεκινήσει να εκτελείται η προκάτοχός της. Εδώ, η λήξη της διαδόχου καθορίζεται από την έναρξη της προκατόχου.


**Σχήμα 3.5 Σχέση αρχής – τέλους (SF)**

Οι *πόροι* μπορούν να είναι διαφορετικών τύπων, περιλαμβάνοντας οικονομικούς πόρους, ανθρώπινο δυναμικό, μηχανήματα, εξοπλισμό, υλικά, ενέργεια, χώρο κτλ. Κατηγοριοποιούνται σε *ανανεώσιμους, μη ανανεώσιμους* και *διπλά περιορισμένους*.

Οι *ανανεώσιμοι πόροι* είναι διαθέσιμοι από περίοδο σε περίοδο, όπου μόνο η συνολική ποσότητα του πόρου εντός κάθε περιόδου είναι περιορισμένη, ενώ η διαθεσιμότητά του ανά περίοδο $k$ είναι $R_k^{\rho}$. Τυπικά παραδείγματα ανανεώσιμων πόρων είναι το ανθρώπινο δυναμικό, τα μηχανήματα, ο εξοπλισμός κτλ. Οι ανά περίοδο ανανεώσιμες μονάδες του πόρου $k$ που απαιτούνται από τη δραστηριότητα $i$, $r_{ik}^{\rho}$ μπορεί να είναι σταθερός αριθμός για όλη τη διάρκεια της δραστηριότητας, μεταβλητός σε σχέση με το στάδιο εκτέλεσης της δραστηριότητας ή στοχαστική μεταβλητή.

Οι *μη ανανεώσιμοι πόροι* είναι διαθέσιμοι στη βάση του συνολικού έργου, με περιορισμένη διαθεσιμότητα ως προς το σύνολο του έργου. Τυπικό παράδειγμα μη ανανεώσιμων πόρων είναι τα χρήματα, οι πρώτες ύλες και η ενέργεια. Το σύνολο των μη ανανεώσιμων πόρων συμβολίζεται ως $R^{\nu}$. Η διαθεσιμότητα του μη ανανεώσιμου πόρου $k$ συμβολίζεται ως $R_k^{\nu}$ και η απαιτούμενη ποσότητα για την εκτέλεση της δραστηριότητας $i$ είναι $r_{ik}^{\nu}$ και μπορεί να είναι συγκεκριμένη, σταθερή ή μεταβλητή, ή στοχαστκή.

Οι *διπλά περιορισμένοι πόροι* είναι περιορισμένοι και ανά περίοδο και για το σύνολο του έργου. Το κεφάλαιο με περιορισμένες ανά περίοδο ταμειακές ροές και ορισμένο συνολικό χρηματικό ποσό είναι ένα τυπικό παράδειγμα. Επίσης, οι ανθρωποώρες ανά ημέρα σε συνυασμό με περιορισμένο σύνολο ανθρωποωρών διαθέσιμων για το σύνολο του έργου είναι ένα ακόμη.

Ένα έργο μπορεί να αναπαρασταθεί σαν ένα δίκτυο έργου, ένα διάγραμμα Gantt (Clark, 1942), σαν γραμμή προγραμματισμού (track planning) (Herroelen et al., 1998) και σαν γραμμή ευστάθειας (line of balance) (Lumsden, 1968). Ένα δίκτυο

έργου μπορεί να περιγραφεί σαν την γραφική απεικόνιση συμβάντων, δραστηριοτήτων και σχέσεων προτεραιότητας. Είναι ένας γράφος $G = (N, A)$ αποτελούμενος από ένα σύνολο κόμβων $N$ και ένα σύνολο τόξων $A$. Υπάρχουν δύο πιθανοί τρόποι απεικόνισης ενός δικτύου έργου: η *δραστηριότητα-στο-τόξο* απεικόνιση ($Activity - on - Arc$ ή $AoA$) που χρησιμοποιεί το σύνολο των τόξων $A$ για απεικόνιση των δραστηριοτήτων και το σύνολο των κόμβων $N$ για απεικόνιση των συμβάντων, και η *δραστηριότητα-στον-κόμβο* απεικόνιση ($Activity - on - Node$ ή $AoN$) που χρησιμοποιεί το σύνολο των κόμβων $N$ για απεικόνιση των δραστηριοτήτων και το σύνολο των τόξων $A$ για απεικόνιση των σχέσεων προτεραιότητας (Demeulemeester and Herroelen, 1992).

Έχοντας ορίσει τί είναι το έργο και ποιά τα βασικά του στοιχεία μπορούμε να προχωρήσουμε στην διοίκηση έργου, που ουσιαστικά είναι ένα σύνολο διαδικασιών με στόχο την επιτυχή ολοκλήρωση του έργου. Ο επίσημος ορισμός σύμφωνα με το PMBOK είναι ο εξής:

*«Η διοίκηση του έργου είναι η εφαρμογή της γνώσης, των δεξιοτήτων, των εργαλείων, και των διαδικασιών στις δραστηριότητές του, με σκοπό να επιτευχθούν οι στόχοι του.*

*Περιλαμβάνει τον σχεδιασμό, τον προγραμματισμό και τον έλεγχο των δραστηριοτήτων του έργου για την επίτευξη της απόδοσης, του κόστους και του χρόνου που έχουν τεθεί ως στόχος, χρησιμοποιώντας τους πόρους όσο το δυνατόν πιο αποδοτικά και αποτελεσματικά»* (PMI, 2012).

Ο σχεδιασμός καλείται να ορίσει μια κατηγοριοποίηση των δραστηριοτήτων που πρέπει να εκτελεστούν με τις συνακόλουθες απαιτήσεις από τους διάφορους πόρους και τις εκτιμήσεις για τη διάρκεια και τα κόστη των διαφόρων δραστηριοτήτων. Ο προγραμματισμός είναι η έκθεση όλων των δραστηριοτήτων του έργου στη χρονική σειρά με την οποία πρέπει να εκτελεστούν. Με τον τρόπο αυτό υπολογίζονται οι απαιτούμενοι πόροι σε κάθε βήμα του έργου και οι αναμενόμενοι χρόνοι ολοκλήρωσης κάθε μιας από της δραστηριότητες. Τέλος, ο έλεγχος επικεντρώνεται στη διαφορά μεταξύ του προγράμματος και της πραγματικής εκτέλεσης του έργου, από τη στιγμή της έναρξής του και έπειτα. Μέσω του ελέγχου και του συντονισμού διευκολύνεται η ροή των δραστηριοτήτων και η επικοινωνία μεταξύ του προσωπικού και εντοπίζονται, καταγράφονται και επιλύονται τα προβλήματα που ανακύπτουν κατά την εκτέλεση του έργου (Lewis, 1998). Όλες οι διαδικασίες της διοίκησης έργου εντάσονται στους δέκα τομείς γνώσης της διοίκησης έργου (Project Management Knowledge Areas) (PMI, 2012):

- Η Διαχείριση Αρμοδιοτήτων (Scope Management) αναφέρεται στις διαδικασίες ελέγχου και καθοδήγησης του γενικού συνόλου ενός έργου, με κατεύθυνση την επίτευξη ενός συγκεκριμένου στόχου.
- Η Διοίκηση Ποιότητας (Quality Management) στοχεύει στην εξασφάλιση ότι η εκτέλεση ενός έργου συμβιβάζεται με τις προδιαγραφές που έχουν τεθεί από τους επενδυτές και τους συμμετέχοντες στο έργο.
- Η Διαχείριση Χρόνου/Προγράμματος (Schedule/Time Management) αφορά την αποτελεσματική και αποδοτική χρήση του χρόνου για τη διευκόλυνση της εκτέλεσης ενός έργου.
- Η Διαχείριση Κόστους (Cost Management) ασχολείται με μεθόδους που έχουν στόχο την συγκράτηση του έργου εντός προϋπολογισμού.
- Η Διαχείριση Κινδύνων (Risk Management) είναι η διαδικασία ταυτοποίησης, ανάλυσης και αναγνώρισης των διαφόρων κινδύνων και αβεβαιοτήτων που μπορεί να επηρρεάσουν ένα έργο.
- Η Διοίκηση Ανθωπίνου Δυναμικού (Human Resources Management) έχει αντικείμενο τη λειτουργία της διοίκησης του ανθρωπίνου δυναμικού καθ' όλη τη διάρκεια του κύκλου ζωής του έργου.
- Η Διοίκηση Προμηθειών (Procurement/Contract Management) ασχολείται τη διαδικασία απόκτησης των απαραίτητων προμηθειών ώστε να ολοκληρωθούν με επιτυχία οι στόχοι του έργου.
- Η Διοίκηση Επικοινωνιών (Communications Management) αφορά την απόκτηση των κατάλληλων δεξιοτήτων με στόχο την επικοινωνία με τους κατάλληλους ανθρώπους τη σωστή στιγμή, με την πρέπουσα οργάνωση, καθοδήγηση και έλεγχο των πληροφοριών.
- Η Διαχείριση Ολοκλήρωσης Έργου (Project Integration Management) περιλαμβάνει τις διαδικασίες που απαιτούνται για τη διασφάλιση οτι τα διάφορα στοιχεία του έργου είναι κατάλληλα και ορθά συντονισμένα.
- Η Διαχείριση Επενδύσεων (Stakeholder Management) ασχολείται με τέσσερις διαδικασίες: την αναγνώριση επενδύσεων, το σχεδιασμό της διαχείρισης των επενδύσεων, τη διαχείριση των επενδυτικών δεσμεύσεων και τον έλεγχο των επενδυτικών δεσμεύσεων.

Από αυτό το σημείο και μετά δίνεται βαρύτητα στη Διαχείριση Χρόνου/Προγράμματος, που αναγνωρίζεται συχνά ως ένας από τους πιο συνήθεις λόγους αποτυχίας των έργων. Περιλαμβάνει την αποδοτική και αποτελεσματική χρήση του χρόνου για την διευκόλυνση της εκτέλεσης ενός έργου και η αποδοτικότητά της αντανακλάται στην εκτέλεση του προγράμματος, καθώς

υπολογίζεται συγκρίνοντας την πραγματική πορεία ή/και το κόστος του έργου με το αρχικό χρονοπρόγραμμα.

## 3.2 Προγραμματισμός Έργων

Το πρόβλημα του προγραμματισμού έργων μελετάται απο τα μέσα του προηγούμενου αιώνα, ωθούμενο από την ανάγκη εξέλιξης και διευκόλυνσης στην παράδοση νέων, κυρίως στρατιωτικής φύσεως, έργων. Στο ανταγωνιστικό περιβάλλον της σύγχρονης αγοράς η αποτελεσματικότητα μιας επιχείρησης στον προγραμματισμό των έργων είναι βασικός παράγοντας για την επιβίωση, την αξιοπιστία και την εξέλιξή της. Εφαρμογές του προγραμματισμού έργων απαντώνται σήμερα σε μεγάλο εύρος της βιομηχανίας, από τον κατασκευαστικό τομέα μέχρι την ανάπτυξη λογισμικού και ούτω καθεξής. Είναι ένα θέμα ιδιαίτερα ελκυστικό στους ερευνητές, καθώς οι παραλλαγές στα προβλήματα του χρονοπρογραμματισμού είναι ιδιαίτερα πολλές και άρα και δύσκολες στην επίλυση.

Ο χρονοπρογραμματισμός περιλαμβάνει την κατασκευή ενός βασικού προγράμματος έργου που ορίζει για κάθε δραστηριότητα τους αποδεκτούς χρόνους έναρξης και ολοκλήρωσης, από άποψη προτεραιοτήτων και χρήσης πόρων, τις ποσότητες των διάφορων ειδών πόρων που θα χρειαστούν για κάθε χρονική περίοδο και σαν αποτέλεσμα τον αντίστοιχο απαιτούμενο προϋπολογισμό για την εκτέλεση του έργου (Brucker et al., 1999). Η δημιουργία ενός ρεαλιστικού χρονοπρογράμματος είναι κριτικής σημασίας για την επιτυχή ολοκλήρωση ενός έργου. Θεμελιώδες ζήτημα είναι η παραγωγή ενός προγράμματος που είναι όχι μόνο εφικτό, από θέμα προτεραιοτήτων δραστηριοτήτων και χρήσης πόρων, έχοντας καλύψει τους αρχικούς περιορισμούς, αλλά επίσης και στιβαρό, ελαχιστοποιώντας τις επιδράσεις πιθανών διαταραχών στη διάρκεια των δραστηριοτήτων και στη διαθεσιμότητα των πορων κατά την εκτέλεση του έργου. Το πρόγραμμα δημιουργείται με βάση έναν ή περισσότερους συνήθεις στόχους (π.χ. τη διάρκεια του έργου) ή μη συνήθεις στόχους (π.χ. την καθαρή παρούσα αξία του έργου).

Ο χρονικός προγραμματισμός περιλαμβάνει τις διαδικασίες εκείνες που απαιτούνται για τη διασφάλιση της ολοκλήρωσης ενός έργου στον προκαθορισμένο χρόνο, προϋπολογισμό και ποιότητα (PMI, 2012). Οι διαδικασίες αυτές συνοψίζονται ως εξής:

- Καθορισμός των προγραμματισμένων δραστηριοτήτων του έργου.
- Διαγραμματική οργάνωση των δραστηριοτήτων, καθορίζοντας τις σχέσεις προτεραιότητας μεταξύ τους.

- Εκτίμηση του είδους και της ποσότητας των πορων που χρειάζονται για την ολοκλήρωση κάθε δραστηριότητας και την επίτευξη των στόχων.
- Εκτίμηση της χρονικής διάρκειας περάτωσης κάθε μιας από τις δραστηριότητες του έργου.
- Σχεδιασμός του χρονοπρογράμματος σύμφωνα με τα παραπάνω στοιχεία.
- Έλεγχος του χρονοπρογράμματος και διενέργεια των απαραίτητων διορθωτικών κινήσεων.


**Σχήμα 3.6 Οι διαδικασίες χρονοπρογραμματισμού έργων**

Στην πράξη, είναι αρκετά πιθανό όχι ένα, αλλά περισσότερα αλληλοεξαρτώμενα έργα να πρέπει να προγραμματιστούν ταυτόχρονα. Αν υπάρχουν πόροι που χρησιμοποιούνται από κοινού από τα διαφορετικά έργα, μπορεί να ανακύψουν δύσκολα προβλήματα προγραμματισμου αν τα έργα πρέπει να προγραμματιστούν παράλληλα. Σε συγκεκριμένες περιπτώσεις, όλα τα δίκτυα έργων μπορούν να συνδυαστούν σε ένα υπερ-δίκτυο με την προσθήκη ενός βοηθητικού υπερ-κόμβου αρχής και ενός βοηθητικού υπερ-κόμβου τέλους. Προθεσμίες και καταληκτικές ημερομηνίες μπορούν τότε να εισαχθούν στις δραστηριότητες τέλους καθενός από τα απλά έργα. Παρόμοια, χρόνοι εκκίνησης μπορούν να εισαχθούν στους κόμβους αρχής κάθε ξεχωριστού έργου.

## 3.3 Κατηγοριοποίηση προβλημάτων Προγραμματισμού Έργων

Οι αναπτυσσόμενες ερευνητικές προσπάθειες στον χώρο του προγραμματισμού έργων έχουν οδηγήσει σε μια ευρεία και συνεχώς αυξανόμενη ποικιλία από τύπους προβλημάτων. Το γεγονός αυτό οδήγησε στην ανάγκη εισαγωγής ενός σχεδίου κατηγοριοποίησης όλων αυτών των προβλημάτων. Το εκτενές σχήμα που χρησιμοποιείται συνήθως στον χρονοπρογραμματισμό έργων (Herroelen et al., 1999) μοιάζει με το κλασικό σχέδιο για το πρόβλημα προγραμματισμού των μηχανών. Στα προβλήματα προγραμματισμού των μηχανών (Graham et al., 1979, Blazewicz et al., 1983) υπάρχει ένα σύστημα κατηγοριοποίησης τριών πεδίων, α|β|γ, όπου το πρώτο πεδίο περιγράφει το

μηχανικό περιβάλλον, το δεύτερο τα χαρακτηριστικά των πόρων και των εργασιών που πρέπει να γίνουν και το τρίτο πεδίο δηλώνει το κριτήριο βελτιστοποίησης (μέτρο απόδοσης). Στον προγραμματισμό έργων, το πεδίο $\alpha$ χρησιμοποιείται για να περιγράψει τα χαρακτηριστικά των πόρων. Περιέχει το πολύ τέσσερα στοιχεία $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. Οι μηχανικές διαδικασίες καθορίζονται από την παράμετρο $\alpha_1$. Όταν όμως έχουμε να κάνουμε με το καθαρό χρονοπρογραμματισμό έργων ο καθορισμός των κατασκευαστικών πόρων δεν σχετίζεται με το πρόβλημα. Αυτό συμβολίζεται με το κενό σύμβολο $\alpha_1 = \circ$. Ο αριθμός των πόρων ενός προβλήματος χρονοπρογραμματισμού έργων (εκτός των μηχανών) καθορίζεται με την παράμετρο $a_2 = \{\circ, 1, m\}$. Όταν κανένας πόρος δεν είναι διαθέσιμος $\alpha_2 = \circ$, όταν μόνο ένας πόρος είναι διαθέσιμος $\alpha_2 = 1$ και στην περίπτωση πολλαπλών πόρων είναι $\alpha_2 = m$, με το $m$ να αναπαριστά τον αριθμό των διαθέσιμων πόρων. Η παράμετρος $\alpha_3$ δηλώνει το είδος των διαφόρων πόρων που χρησιμοποιούνται ($\circ$ καθόλου πόροι, 1 ανανεώσιμοι, $T$ μη ανανεώσιμοι, $1T$ μαζί ανανεώσιμοι και μη ανανεώσιμοι, $v$ μερικώς (μη) ανανεώσιμοι κτλ. ) Τέλος, το $\alpha_4$ περιγράφει τα χαρακτηριστικά διαθεσιμότητας των πορων ($\circ$ συνεχής αυθαίρετη χρήση, $k$ σταθερή ποσότητα από $k$ μονάδες, $v$ μεταβλητή ποσότητα συναρτήσει του χρόνου κτλ. ).

Το δεύτερο πεδίο $\beta$ ορίζει τα χαρακτηριστικά των δραστηριοτήτων ενός προβλήματος προγραμματισμού έργου. Περιέχει το μέγιστο εννέα στοιχεία $\beta_1$, $\beta_2$, $\beta_3$, $\beta_4$, $\beta_5$, $\beta_6$, $\beta_7$, $\beta_8$, $\beta_9$. Η παράμετρος $\beta_1$ δείχνει την πιθανότητα της διακοπής στην εκτέλεση των δραστηριοτήτων ($\circ$, $pmtn$ για διακοπή και συνέχεια σε άλλη χρονική στιγμή, $pmtn - rep$ για διακοπή και συνέχεια από το ίδιο χρονικό σημείο). Η δεύτερη παράμετρος $\beta_2$ αφορά τους περιορισμούς στις σχέσεις προτεραιότητας και το αν είναι δυνατό να υπάρχουν ελάχιστες ή/και μέγιστες χρονικές καθυστερήσεις. Η τρίτη παράμετρος $\beta_3$ περιγράφει τους χρόνους εκκίνησης των δραστηριοτήτων. Η παράμετρος $\beta_4$ περιγράφει τον τύπο της χρονικής διάρκειας των δραστηριοτήτων, αν είναι διακριτή, συνεχής, στοχαστική ή ασαφής. Η παράμετρος $\beta_5$ αφορά την ύπαρξη ή όχι προθεσμιών χρόνου για το έργο και τις δραστηριότητες. Η παράμετρος $\beta_6$ δείχνει τη φύση των απαιτήσεων σε πόρους από τις δραστηριότητες του έργου (σταθερές, μεταβλητές κτλ.). Ο τύπος και ο αριθμός των πιθανών τρόπων εκτέλεσης του έργου ορίζεται από την παράμετρο $\beta_7$. Η παράμετρος $\beta_8$ χρησιμοποιείται για τον καθορισμό των οικονομικών χαρακτηριστικών των δραστηριοτήτων του έργου συσχετίζοντας τις χρηματορροές με τις δραστηριότητες. Η παράμετρος $\beta_9$ χρησιμοποιείται για την περιγραφή των χρόνων μετάβασης (μηδενικοί χρόνοι

μετάβασης, εξαρτώμενοι από τη σειρά εκτέλεσης των δραστηριοτήτων, στοχαστικοί ή ασαφείς) που είναι οι χρόνοι που χρειάζονται για το πέρασμα από την εκτέλεση δραστηριότητας $i$ με χρήση του πόρου $r_x$ στην εκτέλεση δραστηριότητας $j$ με χρήση του ίδιου ή διαφορετικού πόρου.

Το πεδίο $\gamma$, το τρίτο δηλαδή πεδίο, κρατείται για να ορίσει τα κριτήρια βελτιστοποίησης. Υπάρχουν τα συνήθη (πρώιμα) κριτήρια απόδοσης, τα οποία αφορούν συναρτήσεις ποινής (penalty functions) που είναι μη-μειούμενες όσον αφορά τους χρόνους ολοκλήρωσης των δραστηριοτήτων, δηλαδή αν οι χρόνοι ολοκλήρωσης των δραστηριοτήτων αυξηθούν, η αξία της αντικειμενικής συνάρτησης θα μειωθεί. Βασικά παραδείγματα είναι η ελαχιστοποίηση της διάρκειας του έργου (makespan), της αργοπορίας ή καθυστέρησής του, του συνόλου του άμεσου και του έμμεσου κόστους του κτλ. Υπάρχουν ωστόσο και περιπτώσεις προβλημάτων που απαιτούν χρήση των μη συνηθισμένων (ελεύθερων) κριτηρίων απόδοσης, όπου η κατάσταση που ορίστηκε ακριβώς πριν δεν ισχύει, δηλαδή μπορεί η καθυστέρηση δραστηριοτήτων να βελτιώσει την απόδοση του έργου, ακόμη κι αν μια τέτοια καθυστέρηση δεν είναι υποχρεωτικό λόγων περιορισμών να συμβεί. Η μεγιστοποίηση την καθαρής παρούσας αξίας ενός έργου είναι ένα τυπικό παράδειγμα τέτοιου προβλήματος.

## 3.4 Προγραμματισμός Έργων Ύπο Περιορισμένους Πόρους

Η ανάπτυξη των τεχνικών CPM (Critical Path Method) και PERT (Program Evaluation Research Technique) στα μέσα του προηγούμενου αιώνα έδωσε τη δυνατότητα τα έργα να μπορούν να περιγραφούν με διαγράμματα δικτύων, όπου είτε οι δραστηριότητες παρουσιάζονταν ως κόμβοι και οι σχέσεις μεταξύ τους καθορίζονταν από τη δομή του δικτύου (Activity-on-Node - AoE) ή οι δραστηριότητες αναπαριστώνταν ως τόξα (Activity-on-Arc – AoA). Με τον τρόπο αυτό, όμως, ήταν δυνατό να αντιμετωπιστεί μόνο από άποψη χρόνου το πρόβλημα, υποθέτοντας ανυπαρξία περιορισμών σε πόρους. Στην πράξη είναι σπάνιο να μπορεί να τηρηθεί ένα πρόγραμμα δημιουργημένο είτε με CPM είτε με PERT, λόγω έλλειψης διαθεσιμότητας πόρων (Icmeli and Erenguc, 1996a).

Το συγκεκριμένο πρόβλημα του προγραμματισμού των δραστηριοτήτων ενός έργου που υπόκεινται σε σχέσεις προτεραιότητας και περιορισμένους πόρους είναι γνωστό σαν το «πρόβλημα προγραμματισμού δραστηριοτήτων υπό περιορισμένους πόρους» (RCPSP) στη βιβλιογραφία. Προτάθηκε πρώτη φορά από τον Kelley το 1963 και είναι ένα πολύ γενικό πρόβλημα προγραμματισμού που μπορεί να

χρησιμοποιηθεί για τη μοντελοποίηση πολλών πρακτικών εφαρμογών όπως η διαδικασία παραγωγής, η κατασκευή σχολικού χρονοπρογράμματος και τα κατασκευαστικά έργα και είναι ένα πρόβλημα συνδυαστικής βελτιστοποίησης NP-δύσκολο (NP-hard).

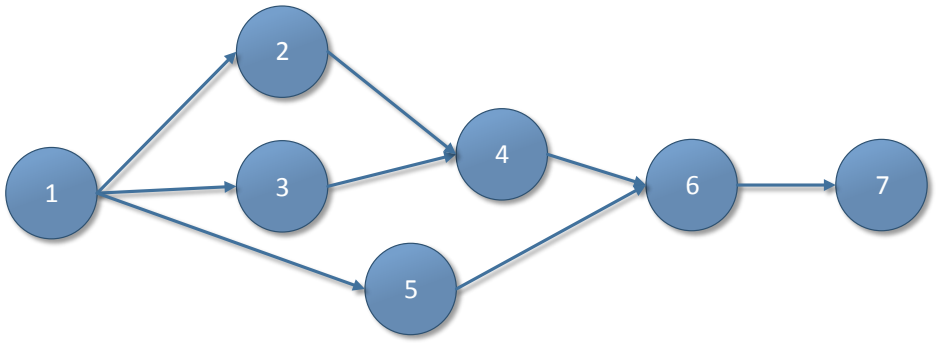
Οι κύριοι άξονες του προβλήματος είναι τρεις: οι δραστηριότητες, οι πόροι και τα κριτήρια απόδοσης. Ένα έργο αποτελείται από απλές δραστηριότητες, για την εκτέλεση των οποίων χρειάζονται πόροι από μια προκαθορισμένη συνολική διαθέσιμη ποσότητα και έτσι είναι περιορισμένο από άποψη πόρων, ενώ ένα ή περισσότερα κριτήρια απόδοσης χρησιμοποιούνται για τη σύγκριση των παραγόμενων προγραμμάτων και τα κριτήρια αυτά αποτελούν τους στόχους βελτιστοποίησης του προβλήματος (Liu et al., 2009).

Το πρώτο κριτήριο βελτιστοποίησης που χρησιμοποιήθηκε για το RCPSP ήταν ο χρονικός ορίζοντας (makespan) του έργου, δηλαδή η συνολική του διάρκεια. Το αντίστοιχο πρόβλημα βελτιστοποίησης ορίστηκε ως «εύρεση εφικτών, από θέμα σχέσεων προτεραιότητας και περιορισμών πόρων, χρόνων έναρξης για όλες τις δραστηριότητες, τέτοιων ώστε ο χρονικός ορίζοντας του έργου να ελαχιστοποιηθεί» (Davis, 1973). Αργότερα, το RCPSP ορίστηκε με μεγαλύτερη λεπτομέρεια σαν το πρόβλημα που συνίσταται από τον προγραμματισμό όλων των δραστηριοτήτων ενός έργου με τρόπο τέτοιο ώστε να ελαχιστοποιηθεί η συνολική του διάρκεια, υποκείμενο σε μηδενικής καθυστέρησης (zero-lag) εφικτές σχέσεις προτεραιότητας του τύπου PERT/CPM και σταθερούς περιορισμούς διαθεσιμότητας στο απαιτούμενο σύνολο ανανεώσιμων πόρων (Herroelen et al., 1998).

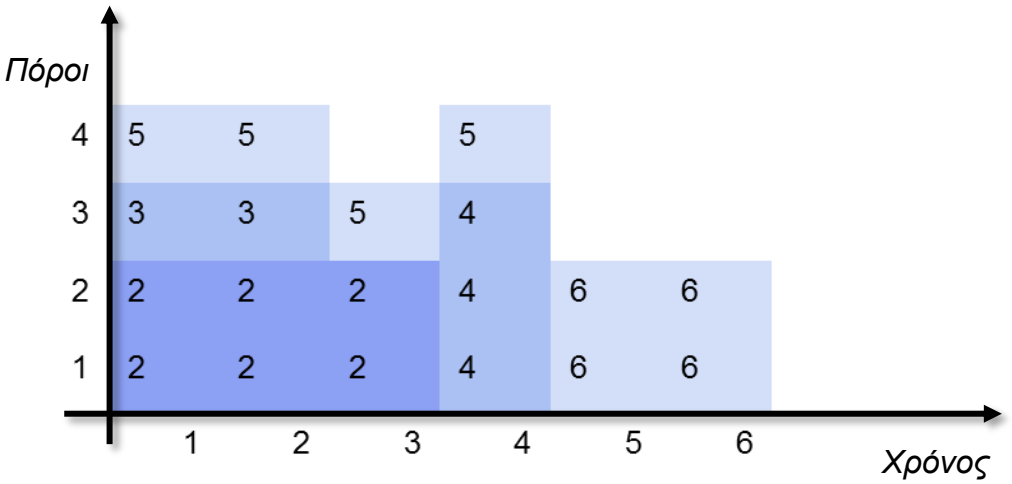
Συνοψίζοντας, το RCPSP αφορά τον προγραμματισμό των δραστηριοτήτων ενός έργου, υποκείμενες σε σχέσεις προτεραιότητας τέλους-αρχής χωρίς χρονικές καθυστερήσεις και σταθερούς περιορισμούς σε πόρους, με σκοπό να ελαχιστοποιηθεί η χρονική διάρκεια του έργου. Οι δραστηριότητες έχουν μοναδικό τρόπο εκτέλεσης, διάρκεια με ορισμένη ακέραια τιμή, η διακοπή κατά την εκτέλεση δέν επιτρέπεται και οι απαιτήσεις σε ανανεώσιμους πόρους κατά τη διάρκεια εκτέλεσης μιας δραστηριότητας είναι σταθερές. Το πρόβλημα αυτό συμβολίζεται σαν $m,1|cpm|C_{max}$ χρησιμοποιώντας την κατηγοριοποίηση που παρουσιάστηκε στο προηγούμενο κεφάλαιο (Herroelen and Demeulemeester, 1996).

**Πίνακας 3.1 RCPSP : Δραστηριότητες, σχέσεις προτεραιότητας, διάρκεια και απαιτήσεις σε πόρους**

| Δραστηριότητα | Προαπαιτούμενες δραστηριότητες | Διάρκεια | Απαίτηση σε πόρο |
|---|---|---|---|
| 1 | - | 0 | 0 |
| 2 | 1 | 3 | 2 |
| 3 | 1 | 2 | 1 |
| 4 | 2, 3 | 1 | 3 |
| 5 | 1 | 4 | 1 |
| 6 | 4, 5 | 2 | 2 |
| 7 | 6 | 0 | 0 |




**Σχήμα 3.7 RCPSP: Αναπαράσταση του δικτύου του έργου με τη μορφή δραστηριότητας-στον-κόμβο (AoN)**




**Σχήμα 3.8 RCPSP: Προκύπτον πρόγραμμα όταν η διαθεσιμότητα του πόρου είναι τρεις μονάδες**

Το πρόβλημα αυτό επομένως ασχολείται με το βέλτιστο χρονικό καταμερισμό των πόρων και καταλήγει στον ορισμό του ποιές δραστηριότητες θα εκτελεστούν σε

ποιό συγκεκριμένο χρόνο. Ο χρονικός καταμερισμός των πόρων έχει αποτελέσει θέμα εκτεταμένης έρευνας από τις πρώτες κιόλας μέρες της επιχειρισιακής έρευνας στα μέσα του 1950 (Tavares, 2002). Το αποτέλεσμα είναι η ύπαρξη σήμερα μιας ευρείας και ιδιαίτερα ογκώδους βιβλιογραφίας πάνω στο αντικείμενο και ένα σημαντικό κενό ανάμεσα στην θεωρία χρονοπρογραμματισμού και την πρακτική εφαρμογή του. Εκείνοι που βρίσκονται στον τομέα της πρακτικής εφαρμογής συχνά ρίχνουν ευθύνες τους θεωρητικούς του χρονοπρογραμματισμού ότι μελετούν πλασματικά προβλήματα και οτι υπεραπλουστεύουν την πραγματικότητα. Από την άλλη, οι θεωρητικοί του χρονοπρογραμματισμού τους κατηγορούν για απροθυμία να εφαρμόσουν τις σύγχρονες εξελίξεις στην πράξη και να υπάρξει έτσι ανάδραση στα αποτελέσματα των ερευνών (Demeulemeester et al., 1994). Παρά τα έγκυρα επιχειρήματα όσον αφορά την απλοποιημένη μορφοποίησή του, το RCPSP έχει πλέον γίνει ένα κλασικό πρόβλημα του χρονοπρογραμματισμού στη βιβλιογραφία.

### *3.4.1 Ορισμός Προβλήματος*

Το πρόβλημα προγραμματισμού δραστηριοτήτων υπό περιορισμένους πόρους (RCPSP) μπορεί να αναλυθεί ως εξής (Christofides et al., 1987, Demeulemeester et al., 1994):

- Ορίζεται ένα μοναδικό έργο με $n$ δραστηριότητες $i = 1, \dots, n$ συν μια βοηθητική δραστηριότητα αρχής $i = 0$ που αναπαριστά την έναρξη του έργου και μια βοηθητική δραστηριότητα τέλους $i = n + 1$ που αναπαριστά την λήξη του, και οι δύο με μηδενική διάρκεια και χωρίς απαιτήσεις σε πόρους.
- Κάθε δραστηριότητα $i$ έχει διάρκεια $d_i$ χρονικές μονάδες. Οι χρόνοι προετοιμασίας κάθε δραστηριότητας δεν λαμβάνονται υπόψη ξεχωριστά, αλλά προσμετρώνται στη συνολική της διάρκεια.
- Οι περιορισμοί στο πρόβλημα είναι δύο ειδών, περιορισμοί σε πόρους και περιορισμοί στις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων.
- Η εκτέλεση των εργασιών γίνεται με την ακριβή σειρά που ορίζουν οι σχέσεις προτεραιότητας, δηλαδή κάθε δραστηριότητα $i$ μπορεί να εκτελεστεί μόνο αφότου ολοκληρωθούν όλες οι διαδικασίες που προηγούνται αυτής άμεσα. Οι περιορισμοί αυτοί συμβολίζονται ως $i \rightarrow j$, που σημαίνει οτι η δραστηριότητα $j$ δεν μπορεί να εκτελεστεί προτού ολοκληρωθεί η δραστηριότητα $i$.
- Όταν το έργο παρουσιάζεται με τη μορφή δικτύου δραστηριότητας-στον-κόμβο $(AoN)$ $G = (V, A)$, όπου το διάνυσμα $V = \{0,1, \dots, n, n + 1\}$ περιέχει όλες τις δραστηριότητες και το σύνολο των τόξων $A = \{(i, j) | i, j \in V; i \rightarrow j\}$ αναπαριστά

τις σχέσεις προτεραιότητας. Για κάθε δραστηριότητα $i$ ορίζεται το σύνολο των προκατόχων της (predecessors) ως $Pred(i) = \{j|(j,i) \in A\}$.

- Η εκτέλεση μιας δραστηριότητας απαιτεί πόρους των οποίων η διαθεσιμότητα είναι περιορισμένη. Ορίζεται σύνολο $K$ ανανεώσιμων πόρων με $k = 1, \dots, r$, και κάθε τύπος πόρου έχει μέγιστη διαθεσιμότητα $R_k$ η οποία είναι σταθερά διαθέσιμη σε κάθε χρονική στιγμή. Η δραστηριότητα $i$ για να πραγματοποιηθεί χρειάζεται $r_{ik}$ μονάδες από πόρο $k$. Οι απαιτούμενοι πόροι δεν καταναλώνονται αλλά χρησιμοποιούνται κατά τη διάρκεια εκτέλεσης της δραστηριότητας και έπειτα επιστρέφουν στην αρχική τους κατάσταση.
- Οι δραστηριότητες θεωρείται οτι δεν είναι διακοπτόμενες, δηλαδή από την στιγμή που θα ξεκινήσει η εκτέλεση τους δεν μπορεί να σταματήσει, μέχρις ότου ολοκληρωθούν.
- Όλα τα δεδομένα θεωρούνται γνωστά εξ' αρχής και μη αμφισβητήσιμα.
- Στόχος είναι ο καθορισμός των χρόνων έναρξης $S_i$ όλων των δραστηριοτήτων $i = 1, \dots, n$ έτσι ώστε:
    - Σε κάθε στιγμη $t$ η συνολική ζήτηση σε πόρους να είναι μικρότερη ή ίση με την διαθεσιμότητα κάθε πόρου $k = 1, \dots, r$.
    - Η δεδομένες σχέσεις προτεραιότητας να ικανοποιούνται, έτσι ώστε κάθε δραστηριότητα να ξεκινά μετά την ολοκλήρωση όλων των προκατόχων της.
    - Ο ορίζοντας του έργου, δηλαδή η συνολική του διάρκεια, ο οποίος είναι ο χρόνος ολοκλήρωσης της βοηθητικής δραστηριότητας τέλους, που αναπαριστά τη λήξη του έργου, να ελαχιστοποιείται.
- Το διάνυσμα $S = \overrightarrow{S_i}$ ορίζει ένα πρόγραμμα για το έργο, υπό την προϋπόθεση οτι δεν επιτρέπεται η διακοπή δραστηριοτήτων την ώρα της εκτέλεσής τους. Ένα πρόγραμμα $S$ καλείται εφικτό όταν όλοι οι περιορισμοί πόρων και προτεραιοτήτων ικανοποιούνται.

Το RCPSP μπορεί να διατυπωθεί εννοιολογικά (Christofides et al., 1987, Demeulemeester and Herroelen, 1992) με τον ακόλουθο τρόπο:

(3.1) $\min f_{n+1}$

(3.2) $S_i + d_i \leq S_j \;\; j = 1, \dots, n \,, \;\; i \in Pred(j)$

(3.3) $\sum_{j \in Act(t)} r_{jk} \leq R_k \;\; Act(t) = \{j | j = 1, \dots, n \; S_{j+1} \leq t \leq S_j \,, \;\; k \in K$

(3.4) $S_j \geq 0 \;\; j = 0, \dots., n+1$

Η αντικειμενική συνάρτηση της εξίσωσης (3.1) ελαχιστοποιεί το χρόνο ολοκλήρωσης της δραστηριότητας τέλους του έργου και συνεπώς το χρονικό ορίζοντα του έργου. Οι περιορισμοί που ορίζονται στην εξίσωση (3.2) περιγράφουν τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, που είναι του τύπου «τέλους-αρχής χωρίς χρονική καθυστέρηση». Οι περιορισμοί που τίθενται στην εξίσωση (3.3) αφορούν την χρήση των πόρων ανά περίοδο, ορίζοντας τη διαθέσιμη ποσότητα καθενός.

### *3.4.2 Παραλλαγές & Επεκτάσεις του Προβλήματος*

Το RCPSP είναι εκ φύσεως ένα ισχυρό μοντέλο, όμως δεν μπορεί να περιγράψει όλες τις περιπτώσεις προβλημάτων που απαντώνται στην πράξη. Έτσι, πολλοί ερευνητές ανέπτυξαν πιο σύνθετα και γενικά μοντέλα επίλυσης προβλημάτων προγραμματισμού έργων, έχοντας συχνά ως βάση το RCPSP. Τα τελευταία χρόνα έχουν προταθεί πεκτάσεις στην έννοια των δραστηριοτήτων, στις σχέσεις προτεραιότητας και στα χαρακτηριστικά του δικτύου έργου, στην έννοια των πόρων και στον αριθμό και το είδος των στόχων (Hartmann and Briskorn, 2010).

#### *3.4.2.1 Προεκχωρητικός Προγραμματισμός (Preemptive scheduling)*

Στο πρόβλημα χρονοπρογραμματισμού έργου με περιορισμένους πόρους και δυνατότητα διακοπής των δραστηριοτήτων (PRCPSP) επιτρέπεται στις δραστηριότητες να διακοπεί η λειτουργία τους σε οποιαδήποτε ακέραια χρονική στιγμή και να συνεχιστεί αργότερα χωρίς κάποια επίπτωση (Bianco et al., 1998, Brucker et al., 1999, Debels and Vanhoucke, 2005, Demeulemeester and Herroelen, 1992). Το πρόβλημα αυτό συμβολίζεται ως $m,1|pmtn,cpm|C_{max}$ χρησιμοποιώντας τον τρόπο κατηγοριοποίησης που παρουσιάστηκε στο κεφάλαιο 3.3 (Herroelen et al., 1999). Εδώ, η διάρκεια $d_i$ μιας δραστηριότητας $i$ μπορεί να χωριστεί μέχρι και σε $d_i$ δραστηριότητες, κάθε μια από τις οποίες θα έχει διάρκεια μια ακέραια μονάδα χρόνου. Σε κάθε μονάδα χρόνου $j = 1,2,\dots,d_i$ της δραστηριότητας $i$ εκχωρείται ένας χρόνος τέλους $f_{ij}$. Για απλοποίηση στην διατύπωση χρησιμοποιείται μια μεταβλητή $f_{i,0}$ που συμβολίζει τη νωρίτερη χρονική στιγμή που μια δραστηριότητα $i$ μπορεί να ολοκληρωθεί. Στο PRCPSP επιτρέπονται μόνο σχέσεις «αρχής – τέλους» (FS) και μηδενική χρονική καθυστέρηση (zero time – lag). Επομένως, η $f_{i,0}$ ισούται με τον αργότερο χρόνο ολοκλήρωσης του συνόλου των προκατόχων της δραστηριότητας $i$. Μια δραστηριότητα $i$ ανήκει στο σύνολο των δραστηριοτήτων σε εξέλιξη τη χρονική στιγμή $t$, $Act(t)$ αν και μόνο αν μια από τις μονάδες διάρκειάς της τελειώνει σε χρόνο $t$.

Με βάση τα παραπάνω, μοντελοποιούμε το PRCPSP ακολούθως:

(3.5) $\min f_{n,0}$

(3.6) $f_{i,d_i} \leq f_{j,0} \; \forall \; (i,j) \in A \; , \; i \in Preds(j)$

(3.7) $f_{i,j-1} + 1 \leq f_{i,j} \quad i = 0, \dots, n \; , \; j = 1, \dots, d_i$

(3.8) $f_{1,0} = 0$

(3.9) $\sum_{i \in Act(t)} r_{ik} \leq a_k \quad k = 1, \dots, m \quad t = 1, \dots, f_{n,0}$

Η αντικειμενική συνάρτηση 3.5 ελαχιστοποιεί το χρονικό ορίζοντα του έργου, ελαχιστοποιώντας τον νωρίτερο χρόνο έναρξης της βοηθητικής δραστηριότητας τέλους, η οποία έχει μηδενική διάρκεια. Η σχέση 3.6 φροντίζει ώστε όλες οι σχέσεις προτεραιότητας να ικανοποιούνται, απαιτώντας ο νωρίτερος χρόνος έναρξης κάθε δραστηριότητας $j$ να είναι μεγαλύτερος από τον χρόνο λήξης της τελευταίας μονάδας διάρκειας κάθε προκατόχου της $i$. Η 3.7 σχέση καθορίζει οτι ο χρόνος τέλους κάθε μονάδας διάρκειας μιας δραστηριότητας πρέπει να είναι τουλάχιστον μια μονάδα χρόνου μεγαλύτερος από το χρόνο τέλους που έχει δοθεί στην προηγούμενη μονάδα διάρκειας της δραστηριότητας αυτής. Η εξίσωση 3.8 δίνει στην βοηθητική δραστηριότητα αρχής σαν νωρίτερο χρόνο έναρξης τη χρονική στιγμή μηδέν. Τέλος, η σχέση 3.9 διασφαλίζει οτι θα τηρηθούν οι περιορισμοί πόρων, απαιτώντας η συνολική ποσότητα των χρησιμοποιούμενων πόρων σε κάθε χρονική στιγμή να είναι μικρότερη ή ίση με τη διαθέσιμη ποσότητα ανά τύπο πόρου.

Στην πράξη δεν είναι όλες οι δραστηριότητες ενός έργου ικανές να διακοπούν, ενώ συχνά κάτι τέτοιο είναι εφικτό μόνο σε συγκεκριμένα χρονικά σημεία και υπάρχει μια ελάχιστη, αλλά όχι μοναδιαία, διάρκεια για κάθε υπό-δραστηριότητα.

#### *3.4.2.2 Πολλαπλοί τρόποι εκτέλεσης δραστηριοτήτων (Multiple Mode)*

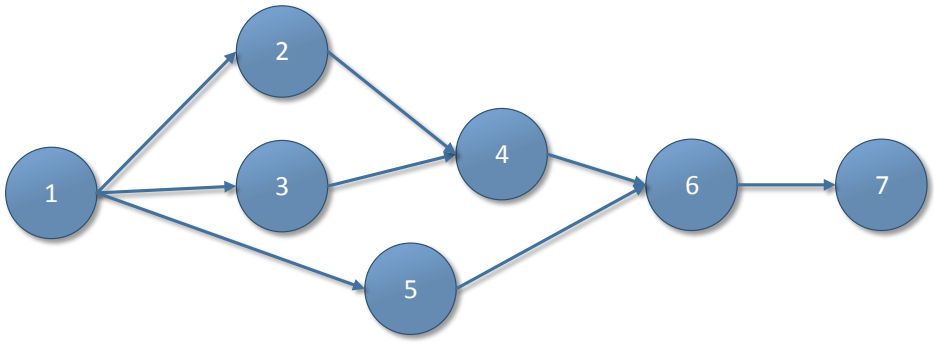
Το κλασικό RCPSP υποθέτει οτι μια δραστηριότητα μπορεί να εκτελεστεί με μοναδικό τρόπο, ο οποίος ορίζεται από συγκεκριμένη διάρκεια και συγκεκριμένες απαιτήσεις σε πόρους. Η έννοια της δραστηριότητας όπως δίνεται στο κλασικό RCPSP εδώ επεκτείνεται, επιτρέποντας αρκετούς διαφορετικούς τρόπους (modes) με τους οποίους μπορεί να εκτελεστεί μια δραστηριότητα. Στο αποκαλούμενο πρόβλημα χρονοπρογραμματισμού έργου με περιορισμένους πόρους και πολλαπλούς τρόπους εκτέλεσης (MRCPSP), κάθε δραστηριότητα μπορεί να εκτελεστεί με έναν από όλους τους δυνατούς τρόπους εκτέλεσης (Elmaghraby, 1964). Κάθε τρόπος εκτέλεσης εκφράζεται σαν διαφορά στην διάρκεια της δραστηριότητας και στις απαιτήσεις της σε πόρους. Οι διαφορετικοί τρόποι εκτέλεσης

δίνουν τη δυνατότητα, με χρήση είτε περισσότερων πόρων του ίδιου τύπου ή με χρήση πιο αποδοτικών τύπων πόρων ανά περίπτωση, να προκύψει μικρότερος χρόνος εκτέλεσης της δραστηριότητας.

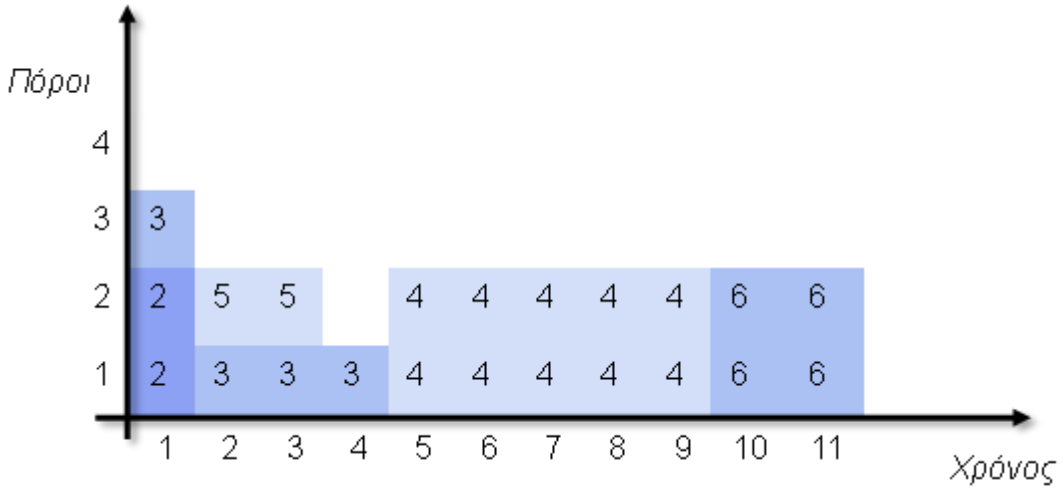
Στο MRCPSP κάθε διαφορετικός τρόπος (mode) εκτέλεσης μιας δραστηριότητας $i$ συμβολίζεται ως $m_i$ και ισχύει $1 < m_i < M_i$ όπου $M_i$ το σύνολο των διαφορετικών τρόπων εκτέλεσης της δραστηριότητας. Η διάρκεια που ορίζεται από τον τρόπο $m_i$ συμβολίζεται με $d_{im_i}$. Από τη στιγμή που μια δραστηριότητα ξεκινήσει να εκτελείται με έναν συγκεκριμένο τρόπο, ο τρόπος αυτός δεν μπορεί να αλλάξει, ούτε η εκτέλεση να διακοπεί.

**Πίνακας 3.2 MRCPSP: Δραστηριότητες, διάρκεια και απαιτήσεις σε πόρους για κάθε δραστηριότητα**

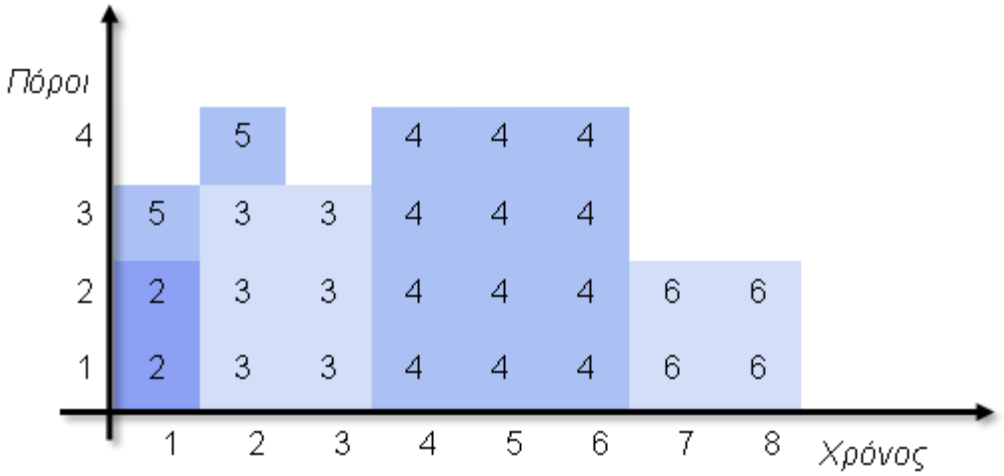
| Δραστηριότητα | Προαπαιτούμενες Δραστηριότητες | Τρόπος Εκτέλεσης | Διάρκεια | Πόρος |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 |
| 2 | 1 | 1 | 1 | 2 |
| | | **2** | **3** | **1** |
| 3 | 1 | 1 | 4 | 1 |
| | | **2** | **2** | **3** |
| 4 | 2,3 | 1 | 5 | 2 |
| | | **2** | **3** | **4** |
| 5 | 1 | 1 | 2 | 1 |
| | | **2** | **1** | **4** |
| 6 | 4,5 | 1 | 4 | 1 |
| | | **2** | **2** | **2** |
| 7 | 6 | 1 | 0 | 0 |




**Σχήμα 3.9 MRCPSP: Αναπαράσταση του δικτύου του έργου με τη μορφή δραστηριότητας-στον-κομβο (AoN)**




**Σχήμα 3.9 MRCPSP: Προκύπτον πρόγραμμα όταν η διαθεσιμότητα πόρου είναι τέσσερις μονάδες και η ανάθεση τρόπων εκτέλεσης είναι {1, 1, 1, 1, 2, 1}**




**Σχήμα 3.11 MRCPSP: Προκύπτον πρόγραμμα όταν η διαθεσιμότητα του πόρου είναι τέσσερις μονάδες και η ανάθεση τρόπων εκτέλεσης είναι {1, 1, 2, 2, 1, 2, 1}**

Τα είδη των πόρων που χρησιμοποιούνται είναι *ανανεώσιμοι, μη ανανεώσιμοι* και *διπλά περιορισμένοι πόροι*, όπως ορίστηκε στο κεφάλαιο 3.1. Οι διπλά περιορισμένοι πόροι όμως δεν αντιμετωπίζονται σαν ξεχωριστή κατηγορία, καθώς μπορούν να μοντελοποιηθούν από έναν ανανεώσιμο και έναν μη ανανεώσιμο πόρο, όπως και γίνεται στην περίπτωση αυτή. Το σύνολο των ανανεώσιμων πόρων συμβολίζεται ως $R^{\rho}$. Για κάθε πόρο $k \in R^{\rho}$ ορίζεται η διαθεσιμότητά του ανά περίοδο $\alpha_k^{\rho}$. Το σύνολο των μη ανανεώσιμων πόρων ορίζεται ως $R^{\nu}$. Για κάθε μη ανανεώσιμο πόρο $l \in R^{\nu}$ η συνολική διαθεσιμότητά του για όλο το έργο συμβολίζεται ως $\alpha_l^{\nu}$. Κάθε δραστηριότητα $i$ σε τρόπο εκτέλεσης $m_i$ απαιτεί την κατανάλωση $r_{im_ik}^{\rho}$ μονάδων από τον ανανεώσιμο πόρο $k \in R^{\rho}$ και την κατανάλωση $r_{im_ik}^{\nu}$ μονάδων από τον μη ανανεώσιμο πόρο $l \in R^{\nu}$. Στόχος του MRCPSP είναι να βρεθεί το χρονοπρόγραμμα με την ελάχιστη διάρκεια όπου καθορίζονται ο χρονισμός των δραστηριοτήτων και η εκχώρηση τρόπων εκτέλεσης σε κάθε μια από αυτές, ούτως ώστε να είναι το πρόγραμμα εφικτό από άποψη περιορισμών πόρων και προτεραιοτήτων. Η μαθηματική μοντελοποίησή του μπορεί να γίνει ως εξής (Hartmann, 2001):

(3.10) $mins_{n+1}$

(3.11) $s.t.\ s_i + d_{im_i} \leq s_j \ \forall\, i \in Pred(j)$

(3.12) $\sum_{i \in Act(t)} r_{im_ik}^{\rho} \leq a_k^{\rho} \ \forall k \in R^{\rho}\,, \forall m_i \in M_i$

(3.13) $\sum_{i=1}^{n} r_{im_il}^{\nu} \leq a_l^{\nu} \ \forall l \in R^{\nu}\,, \forall m_i \in M_i$

(3.14) $s_0 = 0$

(3.15) $s_i \in int^{+}\ \forall i$

Η αντικειμενική συνάρτηση 3.10 ελαχιστοποιεί τον χρονικό ορίζοντα του έργου. Οι περιορισμοί που τίθενται από τη σχέση 3.11 αφορούν την τήρηση των σχέσεων προτεραιότητας τέλους-αρχής (FS) μεταξύ των δραστηριοτήτων, με ελάχιστη χρονική καθυστέρηση το μηδέν. Οι σχέσεις 3.12 και 3.13 αφορούν τους περιορισμούς σε ανανεώσιμους και μη ανανεώσιμους πόρους αντίστοιχα. Η εξίσωση 3.13 δίνει στην βοηθητική δραστηριότητα αρχής του έργου χρονική διάρκεια μηδέν και η 3.15 φροντίζει ώστε οι χρόνοι έναρξης όλων των δραστηριοτήτων να είναι μη αρνητικοί ακέραιοι αριθμοί (Kolisch, 1996).

Προτού ξεκινήσει η διαδικασία επίλυσης του MRCPSP, πραγματοποιείται μια διαδικασία απλοποίησης των δεδομένων εισόδου στο πρόβλημα, με σκοπό τη μείωση του όγκου των δεδομένων και την επιτάχυνση στην εύρεση λύσης από τον αλγόριθμο (Sprecher et al., 1997). Η διαδικασία αυτή αφαιρεί τους μη αποδοτικούς ή τους μη εκτελέσιμους τρόπους και τους πόρους που είναι πλεονασματικοί. Μη

αποδοτικός τρόπος εκτέλεσης $m_i$ είναι εκείνος που σε σχέση με κάποιον άλλο τρόπο εκτέλεσης για την ίδια δραστηριότητα $m'_i$ έχει μεγαλύτερη διάρκεια $d_{im'_i} \leq d_{im_i}$, και όχι μικρότερες απαιτήσεις σε ανανεώσιμους $r^{\rho}_{im'_ik} \leq r^{\rho}_{im_ik}$ και μη ανανεώσιμους πόρους $r^{\nu}_{im'_il} \leq r^{\nu}_{im_il}$. Μη εκτελέσιμος τρόπος $m_i$ είναι εκείνος που παραβιάζει κάποιον από τους περιορισμούς για κάποιον ανανεώσιμο $r^{\rho}_{im_ik} > a^{\rho}_k$ ή κάποιον μη ανανεώσιμο πόρο $\sum_{j\neq i} \min(r^{\nu}_{jm_jl}) + r^{\nu}_{im_il} \leq a^{\nu}_l$. Ένας πόρος $r^{\nu}_l$ καλείται πλεονάζων αν το σύνολο των μέγιστων δυνατών απαιτήσεων για αυτό τον μη ανανεώσιμο πόρο δεν ξεπερνά την αρχική διαθεσιμότητά του $\sum_j max(r^{\nu}_{jm_jl}) \leq a^{\rho}_l$. Η διαδικασία αυτή διευκολύνει την επίλυση του προβλήματος, χωρίς να επηρεάζει την επίλυση του προβλήματος ή τα αποτελέσματά της σε κανένα επίπεδο.

Το MRCPSP υπό τον στόχο ελαχιστοποίησης της χρονικής διάρκειας του έργου με σχέσεις τέλους – αρχής (FS) μεταξύ των δραστηριοτήτων μπορεί να συμβολιστεί ως $m,1T|cpm,disc,mu|C_{max}$ (Herroelen et al., 1997). Είναι ένα NP - δύσκολο πρόβλημα και στην περίπτωση ύπαρξης δύο μη ανανεώσιμων πόρων, το πρόβλημα εύρεσης απλά εφικτής λύσης είναι NP - ολόκληρο (NP - complete) (Kolisch and Drexl, 1997).

#### *3.4.2.3 Προγραμματισμός με Γενικευμένους Χρονικούς Περιορισμούς (Generalized Temporal Constraints)*

Στο απλό RCPSP για να ξεκινήσει η εκτέλεση μιας δραστηριότητας πρέπει πρώτα να έχουν ολοκληρωθεί όλοι οι προκάτοχοί της. Με εφαρμογή των γενικευμένων σχέσεων προτεραιότητας (Generalized Precedence Relationships – GPRs) και μέγιστων και ελάχιστων χρονικών καθυστερήσεων (maximal-minimal lags) οι σχέσεις μεταξύ των δραστηριοτήτων επεκτείνονται. Οι τέσσερις τύποι GPRs είναι οι σχέσεις προτεραιότητας «τέλους-αρχής» (FS), «αρχής-αρχής» (SS), «τέλους-τέλους» (FF) και «αρχής-τέλους» (SF), όπως περιγράφηκαν στο κεφάλαιο 3.1.

Οι ελάχιστες χρονικές καθυστερήσεις στη σχέση (FS) εισάγουν μια ελάχιστη χρονική περίοδο $t$ μεταξύ του χρόνου ολοκλήρωσης μιας δραστηριότητας $i$ που προηγείται και του χρόνου έναρξης μιας δραστηριότητας $j$ που την ακολουθεί. Επιτρέποντας αρνητικές ελάχιστες χρονικές καθυστερήσεις δίνεται η δυνατότητα σε δύο δραστηριότητες να επικαλύπτονται. Αντιστοίχως, οι μέγιστες χρονικές καθυστερήσεις στη σχέση (FS) εισάγουν μια μέγιστη χρονική περίοδο $t$ μεταξύ του χρόνου ολοκλήρωσης μιας δραστηριότητας $i$ που προηγείται και του χρόνου έναρξης μιας δραστηριότητας $j$ που την ακολουθεί. Για παράδειγμα, μια προθεσμία (deadline) που αφορά μια δραστηριότητα $j$ μπορεί να προσομοιωθεί με μια σχέση (FF) και την

ύπαρξη μέγιστης χρονικής καθυστέρησης μεταξύ της βοηθητικής δραστηριότητας αρχής και της δραστηριότητας $j$. Οι GPRs είναι αρκετά χρήσιμες στην πράξη, όπως σε περιπτώσεις που δραστηριότητες απαιτούν συγκεκριμένες ή ταυτόχρονες εκκινήσεις ή λήξεις, εκτέλεση χωρίς καθυστέρηση, υποχρεωτικές επικαλύψεις με άλλες δραστηριότητες, απαιτήσεις σε πόρους μεταβαλλόμενες στο χρόνο, προθεσμίες κτλ. (Herroelen et al., 1999).

Το πρόβλημα χρονοπρογραμματισμού έργων με περιορισμένους πόρους και γενικευμένες σχέσεις προτεραιοτήτων αναφέρεται ως RCPSP-GPR ή RCPSP/max και επεκτείνει το RCPSP επιτρέποντας σχέσεις προτεραιότητας τέλους-αρχής, αρχής-αρχής, τέλους-τέλους και αρχής-τέλους και μαζί ελάχιστες και μέγιστες χρονικές καθυστερήσεις μεταξύ των δραστηριοτήτων. Μπορεί να συμβολιστεί ως $m,1|gpr|C_{max}$. Επιπλέον, οι σχέσεις GPR καθώς μπορούν να μοντελοποιήσουν προθεσμίες δραστηριοτήτων καθώς και μεταβαλλόμενες απαιτήσεις και διαθεσιμότητες σε πόρους μπορούν να οδηγήσουν σε ιδιαίτερα μεγάλη γενίκευση του προβλήματος RCPSP, συμβολιζόμενη ως $m,1,va|gpr,\rho_j,\delta_j,vr|C_{max}$ (Herroelen et al., 1999).

#### *3.4.2.4 Γενικευμένοι Τύποι Πόρων και Μεταβλητές Διαθεσιμότητες*

Το απλό RCPSP αναφέρεται σε ένα μόνο τύπο πόρων, τους ανανεώσιμους, που είναι διαθέσιμοι κάθε χρονική περίοδο, στην πλήρη τους διαθεσιμότητα.

Στο MRCPSP εισάγονται τρεις διαφορετικοί τύποι πόρων, οι ανανεώσιμοι, οι μη ανανεώσιμοι και οι διπλά περιορισμένοι. Οι ανανεώσιμοι πόροι είναι περιορισμένοι ανα περίοδο, όπως για παράδειγμα τα μηχανήματα και το εργατικό δυναμικό. Οι μη ανανεώσιμοι πόροι έχουν περιορισμένη διαθεσιμότητα ως προς το σύνολο του έργου. Παράδειγμα τέτοιου πόρου είναι τα χρήματα, εάν ο προϋπολογισμός του έργου είναι περιορισμένος. Οι διπλά περιορισμένοι πόροι έχουν όρια τόσο ανά περίοδο όσο και στο σύνολο του έργου, όπως για παράδειγμα τα χρήματα στην περίπτωση που είναι περιορισμένος και ο προϋπολογισμός και οι ταμειακές ροές ανα περίοδο.

Μια πιο σπάνια χρησιμοποιούμενη κατηγορία είναι οι μερικώς ανανεώσιμοι πόροι που αποτελούν γενίκευση των ανανεώσιμων και μη ανανεώσιμων πόρων με καθορισμό διαφορετικών διαθεσιμοτήτων ανά περίοδο αλλά και για το σύνολο του έργου μαζί. Επιπλέον, υπάρχουν περιπτώσεις όπου απαιτείται η συνεχής αντί για τη διακριτή διαθεσιμότητα, όπως για παράδειγμα όταν υπάρχουν πόροι όπως η ενέργεια, πρώτες ύλες υγρής μορφής κτλ.

Στο RCPSP οι διαθεσιμότητες των πόρων λαμβάνονται ως σταθερές ανά περίοδο χρόνο, κάτι που όμως δεν είναι πολύ κοντά στην πραγματικότητα. Στην πράξη είναι γεγονός οτι αλλάζει η διαθεσιμότητα σε εργατικό δυναμικό λόγω διακοπών, ασθενειών ή αδειών μητρότητας, όπως και ο διαθέσιμος εξοπλισμός, λόγω συντήρησης ή βλαβών, γεγονότα δηλαδή που βρίσκονται στο καθημερινό πρόγραμμα. Το πρόβλημα αυτό μπορεί να μοντελοποιηθεί με χρήση μέγιστων και ελάχιστων χρονικών καθυστερήσεων (Bartusch et al., 1988). Η σταθερή διαθεσιμότητα μπορεί να οριστεί ως το μέγιστο της χρονικά εξαρτώμενης διαθεσιμότητας κατά την πάροδο του χρόνου, ενώ για κάθε διάστημα μειωμένης διαθεσιμότητας μια τεχνητή δραστηριότητα εισάγεται για κατάλληλη μείωση της διαθεσιμότητας. Κάθε τεχνητή δραστηριότητα τοποθετείται στο επιθυμητό χρονικό διάστημα ορίζοντας για αυτή μια ελάχιστη και μια μέγιστη χρονική καθυστέρηση.

#### *3.4.2.5 Εναλλακτικοί Στόχοι*

Ενώ η ελαχιστοποίηση της χρονικής διάρκειας του έργου βρίσκεται ανάμεσα στους πιο δημοφιλείς στόχους του χρονοπρογραμματισμού, υπάρχουν και αρκετοί άλλοι στόχοι που επιλέγονται προς επίτευξη ή βελτιστοποίηση. Μπορούν να διακριθούν σε στόχους με βάση το χρόνο, με βάση την ευρωστία των παραγόμενων προγραμμάτων, την ικανότητα επαναπρογραμματισμού κατά την εκτέλεση του έργου, με βάση την κατανάλωση και την εξομάλυνση της κατανάλωσης σε ανανεώσιμους ή μη ανανεώσιμους πόρους, το κόστος ή την καθαρή παρούσα αξία ή τέλος ένα σύνολο των παραπάνω, στην περίπτωση των πολλαπλών παράλληλων στόχων.

*Στόχοι με βάση το χρόνο (time-based)*

Πέραν της ελαχιστοποίησης του χρονικού ορίζοντα του έργου (makespan) $C_{max}$ , μπορεί κανείς να θέσει άλλες αντικειμενικές συνάρτησεις προς βελτιστοποίηση με βάση την ελαχιστοποίηση του χρόνου ολοκλήρωσης των δραστηριοτήτων, όπως ο συνολικός χρόνος ροής $\sum_{i=1}^{n} C_i$ή πιο γενικά ο σταθμισμένος (συνολικός) χρόνος ροής $\sum_{i=1}^{n} w_i C_i$ . Άλλοι στόχοι βασίζονται στις ημερομηνίες λήξης (due dates) $dd_i$ που σχετίζονται με τις δραστηριότητες, όπως είναι η μέγιστη καθυστέρηση $L_{max} = max_{i=1}^{n} L_i$ , $L_i = C_i - dd_i$, ο αριθμός των καθυστερημένων δραστηριοτήτων $\sum_{i=1}^{n} U_i$ , $U_i = 0\ if\ C_i \leq dd_i\ otherwise\ U_i = 1$ ή η συνολική αργοπορία του έργου $\sum_{i=1}^{n} T_i$ , $T_i = max\{0, C_i - dd_i\}$. Πιο σύνηθες στην τελευταία περίπτωση είναι να μελετάται η ελαχιστοποίηση της σταθμισμένης (συνολικής) αργοπορίας του έργου (Kolisch, 2000, Viana and Pinho de Sousa, 2000, Ballestín and Trautmann, 2008).

Σε όλες τις παραπάνω περιπτώσεις η αντικειμενική συνάρτηση είναι κανονική, άρα μονότονη, μη μειούμενη ως προς αυξανόμενους χρόνους ολοκλήρωσης. Σε άλλες περιπτώσεις όμως, όπως όταν στόχος του προβλήματος είναι η μέγιστη πρωϊμότητα $max_{i=1}^{n} E_i$ , $E_i \coloneqq max\{0, dd_i - C_i\}$, η αντικειμενική συνάρτηση είναι μη κανονική (Vanhoucke et al., 2003, Lorenzoni et al., 2006). Μια άλλη μη κανονική αντικειμενική συνάρτηση που χρησιμοποιείται αρκετά συχνά (Kimms, 2001, Mika et al., 2005, Vanhoucke et al., 2008) είναι η ελαχιστοποίηση της καθαρής παρούσας αξίας (NPV).

*Στόχοι με βάση την ευρωστία (robustness)*

Κατά τη διάρκεια της εκτέλεσης ενός έργου, μπορεί να προκύψουν καθυστερήσεις που μπορεί να μην είχαν προβλεφθεί κατά τον καθορισμό του προγράμματος. Άρα ο διαχειριστής του έργου (project manager) ενδιαφέρεται για ένα εύρωστο χρονοπρογράμμα, δηλαδή ένα πρόγραμμα που οι καθυστερήσεις να έχουν μικρή επίδραση στο αποτέλεσμά του. Αυτή η προσέγγιση αναφέρεται και ως ενεργητικός προγραμματισμός (proactive scheduling) (Al-Fawzan and Haouari, 2005, Kobylański and Kuchta, 2007).

*Στόχοι με βάση τους πόρους (resources)*

Οι στόχοι με βάση τους πόρους χωρίζονται στο πρόβλημα της επένδυσης πόρων (resource investment – RIP) και στο πρόβλημα της εξομάλυνσης του χρονικού προφίλ χρήσης των πόρων (resource leveling – RLP).

Στο πρώτο πρόβλημα, το RIP (Neumann and Zimmermann, 2000, Drexl and Kimms, 2001), οι διαθεσιμότητες πόρων $R_k$ δεν δίνονται ως δεδομένα αλλά πρέπει να υπολογιστούν σαν επιπλέον μεταβλητές απόφασης με δεδομένο το κόστος ανά μονάδα κάθε τύπου πόρου $c_k$ και μια τιμή-στόχο για το σύνολο των πόρων που θα χρησιμοποιηθούν $Y_k$ . Αντικειμενικός στόχος είναι να βρεθεί ένα πρόγραμμα με χρονική διάρκεια μικρότερη από τη δεδομένη προθεσμία $T$ για το έργο και το ελάχιστο δυνατό κόστος πόρων.

Στο δεύτερο πρόβλημα, το RLP (Davis, 1973, Neumann and Zimmermann, 2000, Viana and Pinho de Sousa, 2000), μετράται η μεταβολή ή η απόκλιση στη χρήση πόρων ανά το χρόνο. Στα προβλήματα απόκλισης με δεδομένο ένα προφίλ πόρων, όπου $r_k^S(t)$ είναι η χρήση σε πόρο $k$ τη χρονική περίοδο $t \in \{1, \dots, T\}$, ο στόχος μπορεί να είναι η ελαχιστοποίηση της απόκλισης από ένα δεδομένο επίπεδο χρήσης πόρων, της υπερφόρτωσης ή της τετραγωνικής απόκλισης. Στα προβλήματα μεταβολής από την άλλη, στόχος είναι η χρήση πόρων να μην μεταβάλλεται

ουσιαστικά κατά την πάροδο του χρόνου. Αυτό μπορεί να γίνει ελαχιστοποιώντας την ανα περίοδο μεταβολή, τη μεγιστη απόκλιση ή την τετραγωνική ανά περίοδο απόκλιση.

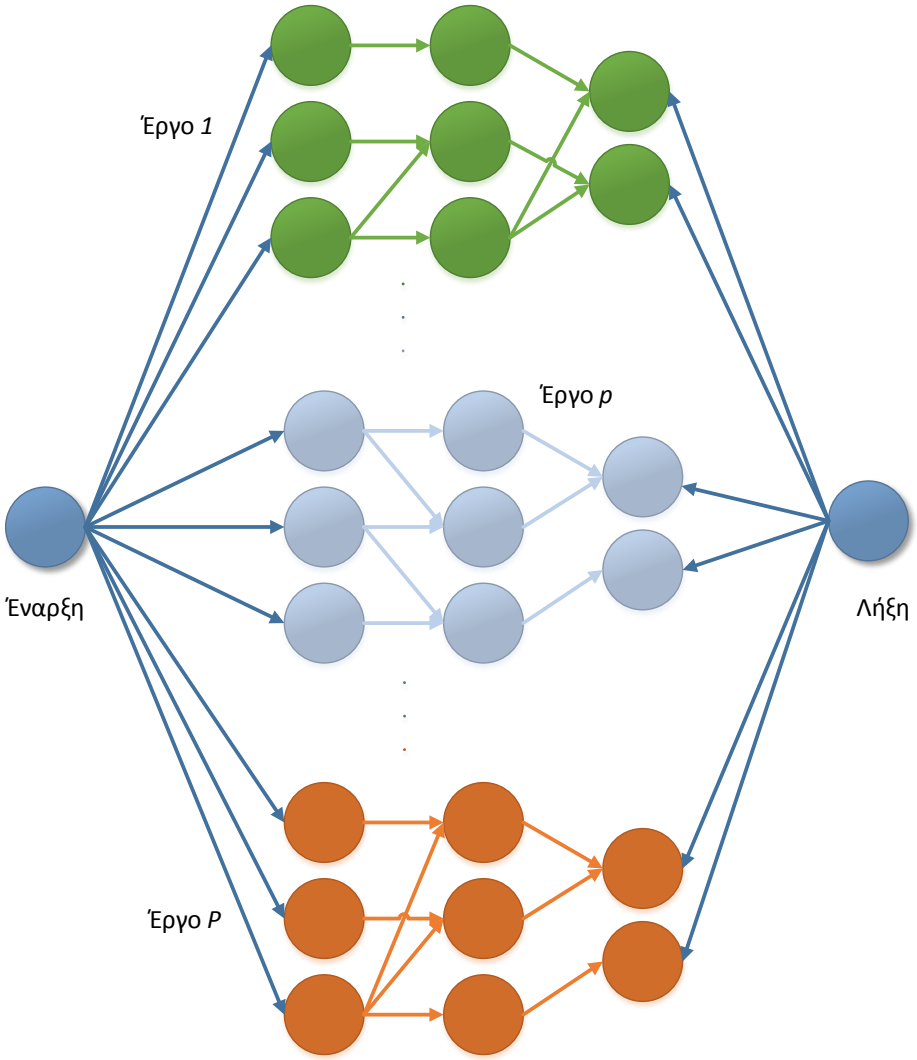
*Στόχοι με βάση τον επαναπρογραμματισμό (rescheduling)*

Ο επαναπρογραμματισμός είναι χρήσιμος όταν το έργο βρίσκεται ήδη σε εξέλιξη, αλλά λόγω απρόβλεπτων γεγονότων, όπως για παράδειγμα καθυστερήσεων, το πρόγραμμα που είχε δημιουργηθεί πριν την έναρξη του έργου πλέον δεν είναι έγκυρο. Σε τέτοια περίπτωση, τα χαρακτηριστικά του έργου αλλάζουν: Κάποιες δραστηριότητες μπορεί να έχουν ήδη τελειώσει και δεν έχουν κάποια σημασία πια, άλλες μπορεί να βρίσκονται σε εξέλιξη και πρέπει να θεωρηθούν μη δυνάμενες να μεταβληθούν και η διαθεσιμότητα σε πόρους μπορεί να έχει αλλάξει και πιθανά να έχει μετατραπεί από χρονικά αμετάβλητη σε συνάρτηση του χρόνου. Σε αντίθεση με τον ενεργητικό (proactive) προγραμματισμό, οπου αναμένονται εκ των προτέρων διαταραχές και στόχος είναι η κατασκευή ενός προγράμματος εύρωστου, που θα τις αντέξει, εδώ στην περίπτωση αυτή κάποια διαταραχή έχει ήδη συμβεί και ένα νέο πρόγραμμα πρέπει να καθοριστεί με όσο το δυνατόν μικρότερη διαφοροποίηση από το αυθεντικό-αρχικό πρόγραμμα. Η περίπτωση αυτή αναφέρεται ως αντενεργητικός (reactive) προγραμματισμός (Calhoun et al., 2002, Vanhoucke et al., 2002).

### *3.4.2.6 Προγραμματισμός Πολλαπλών Έργων (Multiple-Project Scheduling)*

*Δίκτυα πολλαπλών έργων*

Στην πράξη, συχνά όχι ένα αλλά αρκετά και αλληλοεξαρτώμενα έργα πρέπει να προγραμματιστούν ταυτόχρονα, όπως αναφέρεται και στο κεφάλαιο 3.2. Αυτό το γεγονός είναι σημαντικό στην περίπτωση που υπάρχουν δύο ή περισσότερα έργα που είναι πιθανό να εκτελεστούν παράλληλα και μοιράζονται τουλάχιστον ένα πόρο. Ο πιο συνήθης τρόπος αντιμετώπισης πολλαπλών έργων είναι να συμπεριληφθούν τα δίκτυά τους σε ένα υπερδίκτυο, όπως αυτό απεικονίζεται στο Σχήμα 3.10, με την προσθήκη ενός βοηθητικού υπερκόμβου-αρχής και ενός βοηθητικού υπερκόμβου-τέλους (Pritsker et al., 1969). Το πλεονέκτημα της ενσωμάτωσης των πολλαπλών έργων σε ένα μοναδικό δίκτυο είναι οτι έτσι παρέχεται η βάση για την εφαρμογή των μεθόδων προγραμματισμού που χρησιμοποιούνται για απλά έργα και στην περίπτωση των πολλαπλών έργων (Herroelen, 2005).




**Σχήμα 3.10 Αναπαράσταση υπερδικτύου έργων**

Μετά την ενοποίηση μπορούν να προστεθούν επιπλέον περιορισμοί στο πρόβλημα, όπως προθεσμίες για τις βοηθητικές δραστηριότητες κάθε ξεχωριστού έργου, ελάχιστα και μέγιστα χρονικά περιθώρια στις δραστηριότητες και πολλαπλοί τρόποι εκτέλεσης δραστηριοτήτων.

*Στόχοι προγραμματισμού πολλαπλών έργων*

Οι στόχοι κατά τον προγραμματισμό πολλαπλών έργων μπορούν να είναι αντίστοιχα ποικίλοι με τους στόχους του κλασικού RCPSP. Σε προβλήματα πολλαπλών έργων, σε κάθε ένα από τα οποία αντιστοιχεί μια προθεσμία, η αργοπορία βρίσκεται ανάμεσα στα πιο δημοφιλή μέτρα απόδοσης. Μπορεί να χρησιμοποιηθεί και η σταθμισμένη αργοπορία, σε συνδυασμό με ελαχιστοποίηση του κόστους που υπερβαίνει τον προϋπολογισμό κάθε έργου ξεχωριστά καθώς και του

συνόλου τους (Chen, 1994). Άλλοι στόχοι που έχουν χρησιμοποιηθεί είναι η καθαρή παρούσα αξία, η εξομάλυνση του προφίλ των πόρων και η βέλτιστη επένδυση πόρων, η ελαχιστοποίηση της συνολικής σταθμισμένης αργοπορίας και πρωϊμότητας, η ελαχιστοποίηση του μέσου χρονικού ορίζοντα (average makespan) των έργων, ο χρονικός ορίζοντας του «υπερ-έργου» και τέλος η συνολική καθυστέρηση, όπου η καθυστέρηση κάθε έργου υπολογίζεται ως η διαφορά μεταξύ του χρόνου ολοκλήρωσης του έργου στο πρόγραμμα και του χρόνου ολοκλήρωσης του έργου στην περίπτωση που δεν υπάρχουν περιορισμοί στους πόρους (Kurtulus and Davis, 1982). Η τελευταία περίπτωση μπορεί να επεκταθεί με τον ορισμό συναρτήσεων που εκχωρούν διαφορετικές ποινές σε κάθε έργο με βάση το χρόνο καθυστέρησής του (Kurtulus and Narula, 1985).

*Επιπλέον Περιπτώσεις στον Προγραμματισμό Πολλαπλών Έργων*

Εκτός από την περίπτωση που έχει ήδη περιγραφεί, μια άλλη προσέγγιση του προβλήματος είναι η πρόταση οτι το πρόγραμμα κάθε έργου είναι καθορισμένο. Συνεπώς, κάθε έργο μπορεί να αντιμετωπιστεί σαν μια δραστηριότητα με χρονικά μεταβαλλόμενες απαιτήσεις σε πόρους (Heimerl and Kolisch, 2010). Μια άλλη περίπτωση είναι το πρόβλημα προγραμματισμού με ένα από τα υπο-έργα να επαναλαμβάνεται αρκετές φορές, όπου λαμβάνονται υπόψη φαινόμενα εκμάθησης που εφαρμόζονται στην περίπτωση ανθρωπίνων πόρων όταν εκτελούνται αρκετές φορές οι ίδιες δραστηριότητες οπότε είναι πιθανό να μειωθεί ο χρόνος ολοκλήρωσης ενός έργου (Shtub et al., 1996).

Επίσης, θα μπορούσε ένα υποσύνολο των έργων προς προγραμματισμό να χρησιμοποιεί πόρους από εξωτερικούς προμηθευτές, όπου πλέον κατά τον προγραμματισμό του κυρίως έργου κάθε τέτοια περίπτωση υπο-έργου αντιμετωπίζεται σαν μια δραστηριότητα με διάρκεια αντίστοιχη με τον χρονικό ορίζοντα του υπο-έργου (Tukel and Wasti, 2001). Τέλος, υπάρχει η περίπτωση ένα μόνο έργο να πρέπει να επαναλαμβάνεται αρκετές φορές. Τα έργα που προκύπτουν αναπαριστώνται με ένα «υπερδίκτυο» όπως έχει περιγραφεί παραπάνω και δίνεται στον βοηθητικό «υπερ-κόμβο» τέλους μια προθεσμία. Κάθε δραστηριότητα έχει τις ίδιες απαιτήσεις σε πόρους σε κάθε έργο, και στόχος είναι η ελαχιστοποίηση του χρόνου μη χρησιμοποίησης (idle time) των πόρων (Vanhoucke, 2006).

### *3.4.3 Πολυπλοκότητα Αλγορίθμου*

Τα προβλήματα χρονοπρογραμματισμού είναι προβλήματα βελτιστοποίησης και ένα από τα βασικά θέματα που εγείρονται είναι η πολυπλοκότητα των προβλημάτων αυτών, στην συγκεκριμένη περίπτωση η πολυπλοκότητα του προγραμματισμού έργων υπό περιορισμένους πόρους και των επεκτάσεών του.

Είναι γνωστό οτι κάποια υπολογιστικά προβλήματα είναι αρκετά ευκολότερα στην επίλυση από κάποια άλλα. Για παράδειγμα, για κάποια προβλήματα προγραμματισμού υπάρχουν αλγόριθμοι γνωστοί εδώ και δεκαετίες, ικανοί να λύσουν παραλλαγές με χιλιάδες δραστηριότητες, ενώ από την άλλη, για κάποια άλλα προβλήματα, ακόμη και οι καλύτεροι αλγόριθμοι είναι ικανοί να διαχειριστούν μόνο ένα πολύ μικρό αριθμό εργασιών. Το πλαίσιο που χρησιμοποιείται για την κατηγοριοποίηση των υπολογιστικών προβλημάτων παρέχεται από την θεωρία πολυπλοκότητας (Karp, 1975, Garey and Johnson, 1979, Graham et al., 1979, Shmoys and Tardos, 1993).

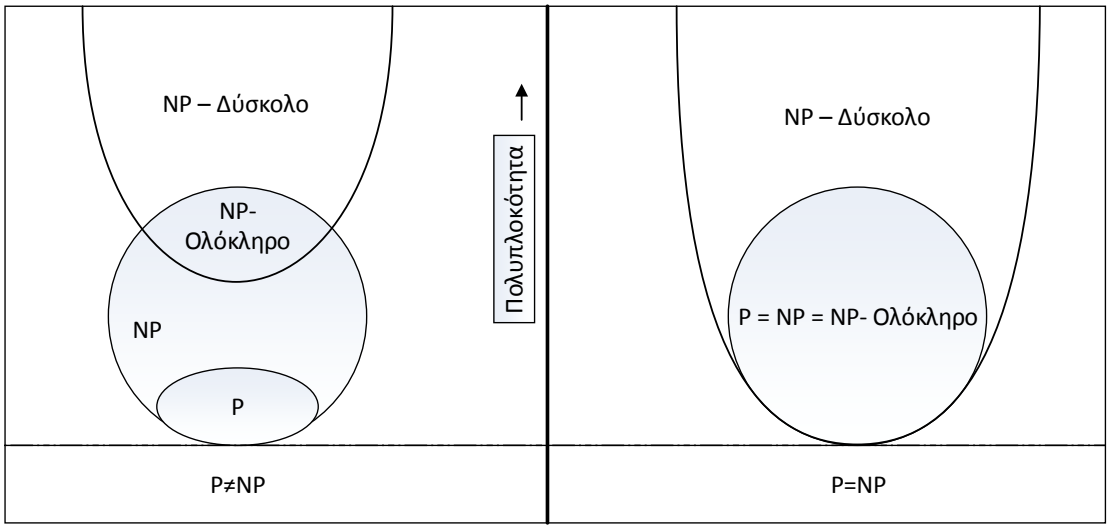
Η θεωρία πολυπλοκότητας είναι ένας κλάδος της θεωρίας υπολογισμού στην θεωρητική υπολογιστική επιστήμη και τα μαθηματικά, ο οποίος επικεντρώνεται στην κατηγοριοποίηση υπολογιστικών προβλημάτων ανάλογα με την έμφυτη δυσκολία τους και στην συσχέτιση των κατηγοριών αυτών μεταξύ τους. Ένα υπολογιστικό πρόβλημα μπορεί να οριστεί σαν μια συνάρτηση $f$ που για κάθε είσοδο $x$ από ένα δεδομένο πεδίο ορισμού δίνει μια έξοδο $f(x)$ σε ένα δεδομένο σύνολο τιμών. Η θεωρία πολυπλοκότητας ενδιαφέρεται για τη μελέτη του χρόνου που χρειάζεται ένας αλγόριθμος να υπολογίσει την $f(x)$ σαν συνάρτηση της έκτασης της κωδικοποίησης της εισόδου $x$, που συμβολίζεται ως $|x|$. Η θεωρία πολυπλοκότητας χρησιμοποιεί μια μηχανή Turing (αφηρημένος υπολογιστής - abstract computer) σαν ένα μαθηματικό μοντέλο ενός αλγορίθμου, αλλά για πιο καλή κατανόηση μπορεί κανείς να σκεφτεί ένα υπολογιστικό πρόγραμμα γραμμένο σε κάποια γλώσσα προγραμματισμού σαν ένα μοντέλο ενός αλγορίθμου. Η αποδοτικότητα με την οποία ο αλγόριθμος υπολογίζει την $f(x)$ για την είσοδο $x$ μετράται με ένα άνω όριο $T(n)$ στον αριθμό των βημάτων που κάνει ο αλγόριθμος για κάθε είσοδο $x$ με $|x| = n$. Στις περισσότερες περιπτώσεις είναι δύσκολο να υπολογίσει κανείς με ακρίβεια την συνάρτηση $T$, για αυτό το ενδιαφέρον δίνεται στην ασυμπτωτική της συμπεριφορά. Έτσι, ορίζεται οτι $T(n) = O(p(n))$ αν υπάρχουν σταθερές $c > 0$ και ένας μη αρνητικός ακέραιος $n_0$ τέτοιοι ώστε $T(n) \leq O(p(n))$ για κάθε ακέραιο $n \geq n_0$. Ένα πρόβλημα $\Pi$ θεωρείται «εύκολο» αν υπάρχει αλγόριθμος $A$ για την επίλυσή του που έχει χρόνο εκτέλεσης $T(n) = O(n^k)$ για κάποια σταθερά $k$, δηλαδή η $T(n)$ οριοθετείται από μια πολυωνυμική συνάρτηση του $n$. Με άλλα λόγια, ένας *πολυωνυμικού χρόνου*

*(πολυωνυμικός) αλγόριθμος* είναι εκείνος του οποίου η συνάρτηση χρόνου πολυπλοκότητας είναι $O(p(n))$, όπου $p$ κάποιο πολυώνυμο και $n$ το μήκος της εισόδου ενός παραδείγματος. Κάθε αλγόριθμος του οποίου η συνάρτηση χρόνου πολυπλοκότητας δεν μπορεί να οριοθετηθεί με αυτό τον τρόπο θα αποκαλείται *εκθετικού-χρόνου αλγόριθμος.*

Κάθε πρόβλημα προγραμματισμού μπορεί να μοντελοποιηθει σαν ένα *πρόβλημα απόφασης*, για παράδειγμα «Υπάρχει εφικτό πρόβλημα που ολοκληρώνεται εντός της προθεσμίας $\delta$?». Η απάντηση «ναι» μπορεί να πιστοποιηθεί μέσα σε μικρό χρονικό διάστημα και μπορεί τυπικά να επιβεβαιωθεί σε πολυωνυμικό χρόνο. Ένα πρόβλημα απόφασης δεν είναι υπολογιστικά πιο δύσκολο από το αντίστοιχο πρόβλημα βελτιστοποίησης «Να βρεθεί ένα εφικτό πρόγραμμα που να έχει το μικρότερο χρόνο ολοκλήρωσης». Αυτό σημαίνει οτι αν κανείς μπορεί να λύσει ένα πρόβλημα βελτιστοποίησης αποδοτικά, δηλαδή σε πολυωνυμικό χρόνο, τότε θα είναι δυνατό να λύσει και το αντίστοιχο πρόβλημα απόφασης αποδοτικά. Από την άλλη, αν ένα πρόβλημα απόφασης είναι υπολογιστικά δύσκολο, τότε και το αντίστοιχο πρόβλημα βελτιστοποίησης θα είναι επίσης δύσκολο (Herroelen et al., 1997).

Η κατηγορία **P** αποτελείται από όλα τα προβλήματα απόφασης που μπορούν να επιλυθούν από μια μηχανή Turing, σε χρόνο που οριοθετείται άνωθεν από έναν πολυωνυμικού χρόνου αλγόριθμο για το μήκος της εισόδου. Η κατηγορία **NP** αποτελείται από τα προβλήματα απόφασης για τα οποία δεν έχει βρεθεί πολυωνυμικού χρόνου αλγόριθμος, αλλά η απάντηση «ναι» μπορεί να δοθεί σε πολυωνυμικό χρόνο. Συνεπάγεται οτι $\mathbf{P} \subseteq \mathbf{NP}$. Ορίζεται ως **NP-ολοκληρωμένο** (NP-complete) ένα πρόβλημα που είναι **NP** και κάθε άλλο πρόβλημα **NP** μπορεί να μειωθεί σε αυτό. Για δύο προβλήματα $A$ και $B$, λέμε οτι το $A$ μειώνεται στο $B$ αν υπάρχει μια πολυωνυμικού χρόνου υπολογίσιμη συνάρτηση $g$ που μετατρέπει εισόδους για το $A$ σε εισόδους για το $B$ έτσι ώστε το $x$ να είναι είσοδος «ναι» για το $A$ αν και μόνο αν το $g(x)$ είναι είσοδος «ναι» για το $B$. Ορίζεται ως **NP-δύσκολο** (NP-hard) ένα πρόβλημα βελτιστοποίησης, αν το αντίστοιχο πρόβλημα απόφασης είναι **NP-ολοκληρωμένο**. Για να αποδείξει κανείς οτι ένα πρόβλημα βελτιστοποίησης είναι **NP-δύσκολο**, πρέπει να αποδείξει οτι το αντίστοιχο πρόβλημα απόφασης είναι **NP-ολοκληρωμένο**. Για να αποδειχθεί οτι ένα πρόβλημα βελτιστοποίησης είναι εύκολο, αρκεί να βρεθεί ένας πολυωνυμικού χρόνου αλγόριθμος βελτιστοποίησης. Έχει αποδειχθεί οτι το RCPSP ανήκει στην κατηγορία των **NP-δύσκολων** προβλημάτων (Blazewicz et al., 1983). Ειδικότερα, το πρόβλημα απόφασης που αντιστοιχεί στο RCPSP αποδείχθηκε οτι είναι **NP-ολοκληρωμένο** πραγματοποιώντας μείωση από

το πρόβλημα τριπλής διχοτόμησης (3-partition), που αφορά την απόφαση αν ένα δεδομένο σύνολο ακεραίων μπορεί να διχοτομηθεί σε τριάδες που να έχουν το ίδιο άθροισμα. Το πρόβλημα αυτό αποδείχθηκε οτι είναι **NP-ολοκληρωμένο** (Garey and Johnson, 1979), οπότε το αντίστοιχο πρόβλημα βελτιστοποίησης,δηλαδή το RCPSP, είναι **NP-δύσκολο.** Οι ορισμοί των συνόλων πολυπλοκότητας, με τον τρόπο που έχουν αποδοθεί, έχουν οδηγήσει στην ανάπτυξη δύο διαφορετικών αντιλήψεων ανάμεσα στους ερευνητές, χωρίς καμία από τις δύο να έχει μέχρι στιγμής αποδειχθεί σωστή ή λάθος, οι οποίες παρουσιάζονται στο Σχήμα 3.11. Η μια άποψη είναι οτι ο χώρος NP δεν είναι υποσύνολο του χώρου των NP-δύσκολων προβλημάτων και τα σύνολα P και NP-ολόκληρων προβλημάτων δεν τέμνονται, ενώ η άλλη υποστηρίζει οτι τα σύνολα P,NP και NP-ολόκληρων προβλημάτων είναι το ίδιο σύνολο, το οποίο είναι και υποσύνολο του συνόλου των NP-δύσκολων προβλημάτων. Ο




**Σχήμα 3.11 Διάγραμμα Euler για τα σύνολα των P, NP, NP-Ολόκληρων και NP-Δύσκολων προβλημάτων**

## 3.5 Μέθοδοι Επίλυσης

Για τα δύσκολα προβλήματα βελτιστοποίησης όπως το RCPSP, έχουν αναπτυχθεί δύο κατηγορίες μεθόδων επίλυσης, οι *ακριβείς αλγόριθμοι* (exact algorithms) ή *αναλυτικές προσεγγίσεις* (exact procedures), που βρίσκουν πάντοτε βέλτιστη λύση και οι *προσεγγιστικοί αλγόριθμοι* (approximation algorithms) ή *ευρετικές μέθοδοι* (heuristics), που παρέχουν προσεγγιστικές λύσεις.

Οι ακριβείς αλγόριθμοι για τα προβλήματα χρονικού προγραμματισμού έργων είναι συνήθως είτε οι αλγόριθμοι γραμμικού προγραμματισμού (linear programming), είτε οι τεχνικές περιορισμού και διακλάδωσης (branch and bound techniques). Οι μέθοδοι αυτοί οδηγούν πάντοτε στη βέλτιστη λύση, παρουσιάζουν όμως το μειονέκτημα των μεγάλων υπολογιστικών χρόνων, ειδικά σε προβλήματα μεγάλου αριθμου δραστηριοτήτων.

Ορίζεται ως ευρετική μέθοδος μια λογική ακολουθία βημάτων, που δίνει όχι απαραίτητα τη βέλτιστη λύση, αλλά μια αρκετά καλή ώστε να μπορεί να χρησιμοποιηθεί στην πράξη. Οι ευρετικές μέθοδοι για το RCPSP διακρίνονται σε δύο κατηγορίες, τις δομικές ευρετικές μεθόδους και τις μεθόδους βελτιστοποίησης ή αλλιώς τις *ευρετικές* και *μετα-ευρετικές* μεθόδους αντίστοιχα. Οι ευρετικές μέθοδοι ξεκινούν από ένα άδειο πρόγραμμα και προσθέτουν δραστηριότητες μέχρις ότου προκύψει ένα εφικτό πρόγραμμα. Για την επιλογή δραστηριοτήτων χρησιμοποιούν κανόνες προτεραιότητας, οι οποίοι ορίζουν τη σειρά με την οποία τοποθετούνται οι δραστηριότητες στο πρόγραμμα. Οι μετα-ευρετικές μέθοδοι ξεκινούν από ένα εφικτό πρόγραμμα που έχει παραχθεί από κάποια δομική ευρετική μέθοδο και πραγματοποιούν χειρισμούς που μετασχηματίζουν το αρχικό πρόγραμμα σε ένα νέο και βελτιωμένο πρόγραμμα. Οι ενέργειες αυτές επαναλαμβάνονται μέχρι να βρεθεί μια τοπικά βέλτιστη λύση. Στην κατηγορία αυτή ανήκουν αλγόριθμοι όπως η *μέθοδος προσομοιωμένης ανόπτησης (simulated annealing)*, η *αναζήτηση ταμπού (tabu search)*, οι *γενετικοί αλγόριθμοι (genetic algorithms)* και η *βελτιστοποίηση με αποικία μυρμηγκιών (ant colony optimization)*.

Ένα μειονέκτημα των ευρετικών μεθόδων είναι η εγκυρότητά τους, που βασίζεται σε σύγκριση των συμπεριφορών της κάθε φορά υπό εξέταση μεθόδου σε μεγάλα (συχνά τυχαία παραγόμενα) σύνολα προβλημάτων, με τα γνωστά βέλτιστα αποτελέσματα σε αυτά. Ένα δεύτερο μειονέκτημα είναι η αδυναμία να εγγυηθεί κανείς εκ των προτέρων ποιά από όλες τις ευρετικές μεθόδους, ή ποιός συνδυασμός τους, θα δώσει τα καλύτερα αποτελέσματα σε ένα δεδομένο πρόβλημα. Παρόλα αυτά όμως, οι ευρετικές μέθοδοι χρησιμοποιούντα ευρέως στην πράξη για την

αντιμετώπιση των σύνθετων, έντονα συνδυαστικών προβλημάτων του χρονοπρογραμματισμού (Herroelen et al., 1998).

### *3.5.1 Αναλυτικές Μέθοδοι*

#### *3.5.1.1 Αλγόριθμος Διακλάδωσης και Περιορισμού (Branch and Bound)*

Η μέθοδος αυτή προτάθηκε πρώτη φορά το 1960 (Land and Doig, 1960) για διακριτό προγραμματισμό. Ο αλγόριθμος Branch and Bound (B&B) είναι μια μέθοδος διακριτής και συνδυαστικής βελτιστοποίησης, όπου ένα μεγαλύτερο πρόβλημα διασπάται επαναληπτικά σε μικρότερα (λειτουργία της «διακλάδωσης») και για κάθε ένα υποπρόβλημα εκτιμώνται οι πιθανές λύσεις και αν δεν είναι ικανοποιητικές το «κλαδί» απομονώνεται και αγνοείται για την υπόλοιπη διαδικασία επίλυσης (λειτουργία «περιορισμού»).

Στο σημείο αυτό πρέπει να γίνει εισαγωγή της απαραίτητης ορολογίας και των κανόνων που χαρακτηρίζουν τους Branch and Bound αλγορίθμους.

Όροι:

- Κόμβος: κάθε πλήρης ή μερική λύση
- Κόμβος-φύλλο: κάθε ολοκληρωμένη λύση, με όλες τις μεταβλητές γνωστές και την τιμή της αντικειμενικής συνάρτησης υπολογισμένη, όχι απλώς εκτιμημένη.
- Κόμβος όρασης: μερική λύση, εφικτή ή ανέφικτη, με υπολογισμένη τιμή της αντικειμενικής συνάρτησης.
- Συνάρτηση περιορισμού: Μέθοδος εκτίμησης της καλύτερης τιμής της αντικειμενικής συνάρτησης με επέκταση των κόμβων όρασης, αποφεύγοντας παράλληλα την παράληψη καλών λύσεων.
- Διακλάδωση: Η διαδικασία δημιουργίας κόμβων-παιδιών για ένα κόμβο-όρασης. Δημιουργείται ένας κόμβος-παιδί για κάθε πιθανή τιμή της επόμενης μεταβλητής.
- Αξιωματική λύση: η καλύτερη ολοκληρωμένη εφικτή λύση που έχει βρεθεί ως εκείνο το σημείο.

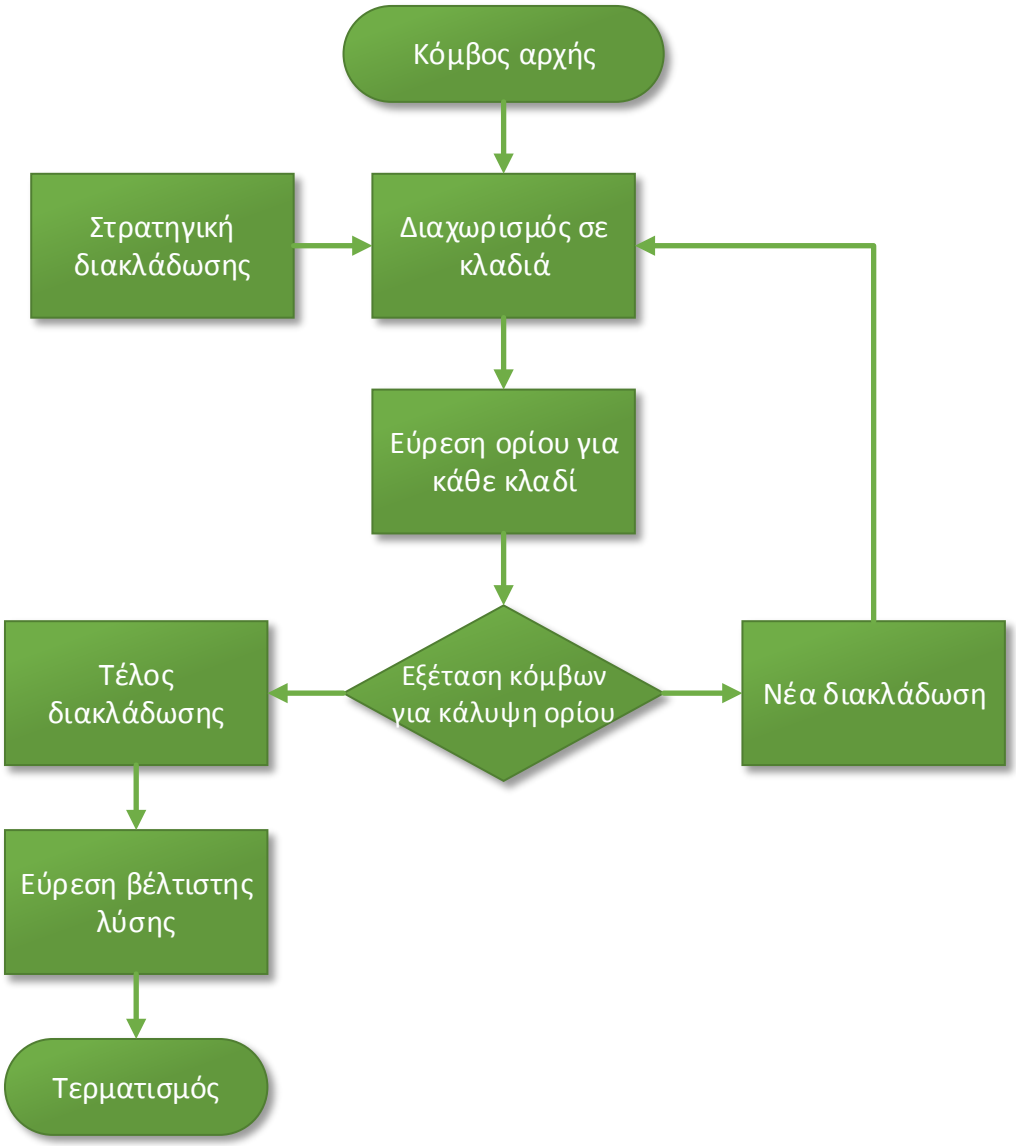
Κανόνες:

- Διακλάδωση: Με δεδομένο τον κόμβο-όρασης, ο τρόπος δημιουργίας των κόμβων-παιδιών.
- Υπολογισμός κατώτατου ορίου: Ο τρόπος με τον οποίο υπολογίζεται το κατώτατο όριο για ένα κόμβο, δηλαδή ο αριθμός που περιορίζει από κάτω

το σύνολο των λύσεων που μπορούν να υπολογιστούν από αυτό τον κόμβο.

- Επιλογή επόμενου κόμβου: Ο τρόπος επιλογής του κόμβου όρασης από τον οποίο θα ξεκινήσει η επόμενη διακλάδωση. Υπάρχουν τρεις διαφορετικοί τρόποι:
  - Καλύτερος-πρώτος κόμβος: Επιλογή του κόμβου όρασης με την καλύτερη τιμή της συναρτησης περιορισμού, οπουδήποτε στο «δέντρο» του B&B αλγορίθμου.
  - Σε βάθος-πρώτος κόμβος: Επιλογή από τους κόμβους-παιδιά του τρέχοντα κόμβου, έτσι ώστε κάθε επανάληψη να οδηγεί ένα βήμα βαθύτερα μέσα στο δέντρο και να επιτυγχάνονται σύντομα αξιωματικές λύσεις.
  - Σε πλάτος-πρώτος κόμβος: Επέκταση των κόμβων οράσεως με τη σειρά με την οποία δημιουργήθηκαν.
- Κλάδεμα-Εμβάθυνση: Ο τρόπος αναγνώρισης αν ένας κόμβος οδηγεί σε ανέφικτες ή μη βέλτιστες λύσεις, ή αν για κάθε λύση που μπορεί να δημιουργηθεί από αυτό τον κόμβο υπάρχει άλλη παρόμοια ή καλύτερη λύση που μπορεί να δημιουργηθεί με διακλάδωση από διαφορετικό κόμβο.
- Αξιωματοποίηση: Ο τρόπος αναγνώρισης αν η εφικτή λύση που δίνει ένας κόμβος-φύλλο είναι η βέλτιστη.

  Ένας κλασικός Branch & Bound αλγόριθμος παρουσιάζεται στο σχήμα 3.12 που ακολουθεί.




**Σχήμα 3.12 Σχηματική αναπαράσταση ενός Branch & Bound αλγορίθμου**

Έχει αναπτυχθεί ένας μεγάλος αριθμός B&B διαδικασιών για βέλτιστη επίλυση συγκεκριμένων περιπτώσεων του RCPSP (Pritsker et al., 1969, Davis, 1973, Stinson et al., 1978, Christofides et al., 1987, Bell and Park, 1990). Οι Demeulemeester και Herroelen (Demeulemeester and Herroelen, 1992) παρουσίασαν μια αποδοτική B&B διαδικασία με την ονομασία *«DH-procedure»* της οποίας η υπολογιστική ταχύτητα κατά τα πειράματα αποδείχθηκε σχεδόν δώδεκα φορές μεγαλύτερη από την ως τότε αποδοτικότερη διαδικασία που είχε αναπτυχθεί (Stinson et al., 1978).

Ακολουθεί ανάλυση της λειτουργίας του Branch and Bound αλγορίθμου DH-procedure για την επίλυση του RCPSP. Ο αλγόριθμος δημιουργεί ένα δέντρο αναζήτησης, που έχει ως κόμβους μερικώς ολοκληρωμένα προγράμματα $PS$ στα

οποία έχουν δοθεί χρόνοι τέλους προσωρινά, στο σύνολο των δραστηριοτήτων που τα απαρτίζουν. Τα προγράμματα αυτά εξετάζονται σε χρονική στιγμή $m$ ανάλογα με το χρόνο ολοκλήρωσης μιας η περισσότερων δραστηριοτήτων του έργου. Στα προγράμματα $PS$ οι αποφάσεις προγραμματισμού είναι προσωρινές με την έννοια οτι στους κόμβους-παιδιά οι δραστηριότητες που είχαν προηγουμένως προγραμματιστεί μπορεί να καθυστερήσουν ως αποτέλεσμα αποφάσεων που λαμβάνονται αργότερα στο πρόγραμμα. Τα μερικά προγράμματα $PS$ ξεκινούν να παράγονται με αρχή τη στιγμή μηδέν και προχωρώντας προσθέτονται σε κάθε σημείο απόφασης υποσύνολα δραστηριοτήτων μέχρις ότου ένα ολοκληρωμένο εφικτό πρόγραμμα δημιουργηθεί.

Τη στιγμή $m$ το αντίστοιχο πρόγραμμα $PS_m$ περιέχει κάποιες δραστηριότητες που έχουν ολοκληρωθεί και άλλες σε εξέλιξη. Οι ολοκληρωμένες δραστηριότητες τοποθετούνται στο σύνολο $F_m$ και οι μη ολοκληρωμένες στο σύνολο $S_m$. Το σύνολο των επιλέξιμων δραστηριοτήτων $E_m$ είναι το σύνολο των δραστηριοτήτων που δεν έχουν προγραμματιστεί ακόμα, αλλά έχουν προγραμματιστεί όλοι οι προκάτοχοί τους.

Κατά την προσπάθεια προγραμματισμού όλων των επιλέξιμων δραστηριοτήτων τη στιγμή $m$, προκαλούνται συγκρούσεις ανάμεσα στις δραστηριότητες εξαιτίας των απαιτήσεων τους σε πόρους και αυτό οδηγεί στην διακλάδωση στο δέντρο αναζήτησης λύσης. Κάθε νέο κλαδί περιγράφει και έναν διαφορετικό τρόπο επίλυσης των συγκρούσεων αυτών ανάμεσα στις δραστηριότητες και την επιλογή εκείνων των δραστηριοτήτων που θα οδηγηθούν σε καθυστέρηση. Το σύνολο των καθυστερημένων δραστηριοτήτων $D(p)$ αποτελείται από όλα τα υποσύνολα δραστηριοτήτων $D_q$ που είτε εκτελούνται είτε είναι επιλέξιμες, η καθυστέρηση των οποίων μπορεί να επιλύσει την αντιπαράθεση στο επίπεδο $p$ του δέντρου. Το $D_q$ είναι το ελάχιστο δυνατό υποσύνολο αν δεν περιέχει άλλες εναλλακτικές επιλογές καθυστέρησης σαν υποσύνολό του. Κάθε εναλλακτικό σύνολο καθυστερημένων δραστηριοτήτων αξιολογείται με υπολογισμό του κάτω ορίου (lower bound-LB) της κρίσιμης ακολουθίας (Stinson et al., 1978).

Χρησιμοποιούνται δύο κανόνες κυριαρχίας (dominance rules): ο κανόνας της μετακίνησης αριστερά και ο κανόνας των διαχωριστικών συνόλων. Στον πρώτο, στην περίπτωση ενός μη άδειου συνόλου καθυστερημένων δραστηριοτήτων, ελέγχεται η υπόθεση αν οι περιορισμοί προτεραιότητας που εφαρμόστηκαν σε προηγούμενα επίπεδα του δέντρου αναζήτησης ανάγκασαν μια δραστηριότητα $i$ να γίνει επιλέξιμη τη χρονική στιγμή $m$ και επιλέγεται να εκτελεστεί αυτή η δραστηριότητα τη στιγμή $m$ και αν το σύνολο των δραστηριοτήτων που καθυστερούν $D(q)$ θα επέτρεπε σε αυτή τη δραστηριότητα $i$ να μεταφερόταν αριστερά χωρίς να προκαλέσει κάποια

αντιπαράθεση σε ανάγκες σε πόρους, τότε το αντίστοιχο μερικό πρόγραμμα εγκαταλείπεται. Στο δεύτερο κανόνα, κάθε χρονική στιγμή $m$ ορίζεται ένα σύνολο διαχωρισμού $C_m$ αποτελούμενο από όλες τις μη προγραμματισμένες δραστηριότητες των οποίων όλοι οι προκάτοχοι ανήκουν στο μερικό πρόγραμμα. Αν ένα σύνολο διαχωρισμού $C_k$ που είχε αποθηκευτεί προηγουμένως και ανήκει σε διαφορετικό μονοπάτι του δέντρου είναι ίσο με το τρέχον $C_m$ και οι δραστηριότητές του δεν τελειώνουν αργότερα από αυτές στο $C_m$ και $k \leq m$, τότε το τρέχον μερικό πρόγραμμα μπορεί να εγκαταληφθεί.

Η διαδικασία της προς τα πίσω αναζήτησης (backtracking) πραγματοποιείται όταν ένα πρόγραμμα ολοκληρώνεται ή όταν ένα κλαδί είναι έτοιμο για κλάδεμα από τον υπολογισμό του κάτω ορίου (lower bound) ή/και από τους κανόνες κυριαρχίας (dominance rules). Αν δεν υπάρχει εναλλακτικό σύνολο καθυστερημένων δραστηριοτήτων σε αυτό το επίπεδο, γίνεται προς τα πίσω αναζήτηση για το προηγούμενο σύνολο. Όταν το backtracking φτάσει στο μηδενικό επίπεδο, δηλαδή στον κόμβο αρχής, η διαδικασία αναζήτησης έχει ολοκληρωθεί και η βέλτιστη λύση έχει βρεθεί και επικυρωθεί.

### *3.5.2 Ευρετικοί Αλγόριθμοι*

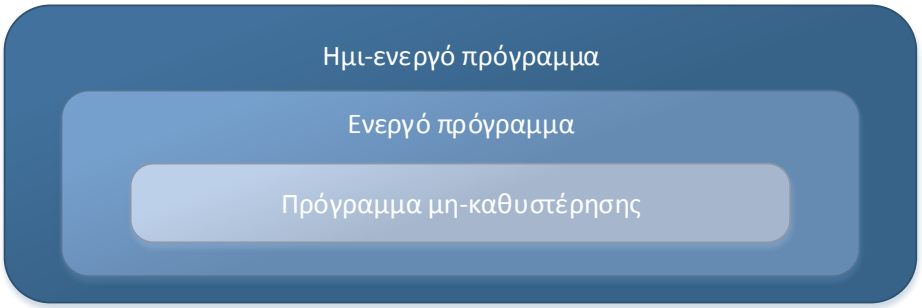
#### *3.5.2.1 Μέθοδος Παραγωγής Προγραμμάτων (Schedule Generation Scheme)*

Από την πρώτη στιγμή εισαγωγής της δομικής ευρετικής μεθόδου παραγωγής προγραμμάτων (Schedule Generation Scheme-SGS) (Kelley Jr and Walker, 1959) παρουσιάστηκε έντονο ενδιαφέρον προς αυτή και πολλοί ερευνητές ασχολήθηκαν και δημιούργησαν αλγορίθμους που βασίζονται σε αυτή. Οι δομικές ευρετικές μέθοδοι αποτελούνται από δύο βασικά τμήματα, τη μέθοδο προγραμματισμού και τον κανόνα προτεραιότητας που χρησιμοποιούν. Η μέθοδος προγραμματισμού καθορίζει τον τρόπο με τον οποίο κατασκευάζεται ένα εφικτό πρόγραμμα, δίνοντας αρχικούς χρόνους στις διαφορετικές δραστηριότητες. Οι δύο βασικές μέθοδοι προγραμματισμού είναι η σειριακή (serial) και η παράλληλη (parallel). Ο κανόνας προτεραιότητας (priority rule) από την άλλη, καθορίζει τη δραστηριότητα που επιλέγεται κάθε φορά κατά την ευρετική διαδικασία αναζήτησης. Η χρήση ενός κανόνα προτεραιότητας έχει σαν αποτέλεσμα τη δημιουργία μιας λίστας προτεραιότητας (priority list) στην οποία κατατάσσονται οι δραστηριότητες με σειρά που ικανοποιεί τους περιορισμούς προτεραιότητας μεταξύ τους (precedence relations). Τα προγράμματα που προκύπτουν εντάσσονται σε μια από τις παρακάτω κατηγορίες, όπως αυτές απεικονίζονται στο Σχήμα 3.13:

Ημι-ενεργά προγράμματα: Εφικτά προγράμματα, στα οποία οι διαδικασίες έχουν προγραμματιστεί όσο το δυνατόν νωρίτερα, δηλαδή καμιά δραστηριότητα δεν μπορεί να αρχίσει νωρίτερα χωρίς να αλλάξει τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων.

Ενεργά προγράμματα: Εφικτά προγράμματα, στα οποία καμιά δραστηριότητα δεν μπορεί να ξεκινήσει νωρίτερα χωρίς να καθυστερήσει κάποια άλλη δραστηριότητα ή να παραβιάσει κάποια σχέση προτεραιότητας.

Προγράμματα μη-καθυστέρησης: Εφικτά προγράμματα στα οποία κανένας πόρος δεν μένει άεργος, ενώ θα μπορούσε να αρχίσει η επεξεργασία μιας δραστηριότητας.




**Σχήμα 3.13 Κατηγορίες προγραμμάτων**

Αρχικά, αυτές οι ευρετικές μέθοδοι αποτελούνταν από μια προγραμματιστική μέθοδο που συνδυαζόταν με ένα κανόνα προτεραιότητας και παραγόταν ένα πρόγραμμα-λύση, το οποίο συνιστά μια «μέθοδο ενός περάσματος» (single-pass method). Αναπόφευκτα άρχισαν να γίνονται πιο περίτεχνες απαιτώντας την επανάληψη της διαδικασίας πάνω από μια φορά, χρησιμοποιώντας διαφορετικούς κανόνες προτεραιότητας ή/και μεθόδους προγραμματισμού, περνώντας έτσι στην προσέγγιση που καλείται «μέθοδος πολλαπλών περασμάτων» (multi-pass method).

Η σειριακή μέθοδος παραγωγής προγραμμάτων (serial SGS ή SSGS) προσθέτει διαδοχικά δραστηριότητες στο πρόγραμμα μέχρις ότου παραχθεί ένα εφικτό πρόγραμμα. Σε κάθε επανάληψη επιλέγεται η επόμενη προς προγραμματισμό δραστηριότητα με βάση τη λίστα προτεραιότητας και ορίζεται ο νωρίτερος δυνατός χρόνος έναρξής της έτσι ώστε να μην παραβιάζεται κανένας περιορισμός πόρων ή προτεραιοτήτων. Έστω $g = 1, \dots, n$ το σύνολο των βημάτων του αλγορίθμου, $S_g$ το σύνολο των ήδη προγραμματισμένων δραστηριοτήτων και $D_g$ το σύνολο των επιλέξιμων δραστηριοτήτων, αυτών δηλαδή που έχουν προγραμματιστεί και ολοκληρωθεί όλοι οι προκάτοχοί τους, άρα $D_g = \{j | j \notin S_g, Pred(j) \in S_g\}$. Έστω

$F_g = \{f_j | j \in S_g\}$ το σύνολο των χρόνων ολοκλήρωσης των δραστηριοτήτων κατά το βήμα $g$ και $\tilde{R}_k(t) = R_k - \sum_{j \in Act(t)} r_{jk}, k \in K$ η εναπομείνασα διαθεσιμότητα σε πόρο τύπου $k$ τη χρονική στιγμή $t$. Ακολουθεί ο Αλγόριθμος 3.1 που είναι αναπαράσταση σε ψευδοκώδικα του SSGS:

**Αλγόριθμος 3.1 Σειριακή μέθοδος παραγωγής προγραμμάτων (SSGS)**

---

$F_0 = 0,\ S_0 = \{0\}$ ;

for $g = 1\ to\ n$ do

Calculate $D_g,\ F_{g,}\ \tilde{R}_k(t)\ \ (k \in K,\ t \in F_g)$ ;

Select $j \in D_g$ ;

$EF_j = max_{h \in Pred(j)}(f_h + d_j)$ ;

$f_j = \min\{t \in [EF_j - d_j, LF_j - d_j] \cap F_g |\ r_{jk} \leq \tilde{R}_k(\tau),\ k \in K, \tau \in [t, t + d_j] \cap F_g\} + d_j$ ;

$S_g = S_{g-1} \cup \{j\}$ ;

end

$f_{n+1} = max_{h \in Pred(n+1)}\{f_h\}$

---

Έχει αποδειχθεί οτι κάθε πρόγραμμα που παράγεται από τη σειριακή μέθοδο παραγωγής προγραμμάτων ανήκει στο σύνολο των ενεργών προγραμμάτων, που έχουν την ιδιότητα οτι καμιά από τις δραστηριότητες δεν μπορεί να αρχίσει νωρίτερα χωρίς να καθηστερήσει η εκτέλεση κάποιας άλλης δραστηριότητας (Kolisch, 1996). Για προβλήματα προγραμματισμού με κανονικό μέτρο απόδοσης η βέλτιστη λύση βρίσκεται πάντοτε μέσα στο σύνολο των ενεργών προγραμμάτων.

Σε αντίθεση με τη σειριακή μέθοδο παραγωγής προγραμμάτων, η μέθοδος παράλληλης παραγωγής προγραμμάτων (parallel SGS ή PSGS) πραγματοποιεί επαναλήψεις επί των διαφορετικών σημείων απόφασης όπου μπορούν να προστεθούν δραστηριότητες στο πρόγραμμα, κάνοντας έτσι προσαύξηση χρόνου. Τα σημεία απόφασης είναι στην ουσία οι χρόνοι ολοκλήρωσης των ήδη προγραμματισμένων δραστηριοτήτων και έτσι μπορούν να υπάρχουν το πολύ $n$ σημεία απόφασης (αφού $n$ το πλήθος των δραστηριοτήτων) τα οποία θα ληφθούν υπόψη από τη μέθοδο παράλληλου προγραμματισμού. Σε κάθε σημείο απόφασης εξετάζονται με βάση τη σειρά προτεραιότητας οι μη προγραμματισμένες δραστηριότητες των οποίων οι προκάτοχοι έχουν ολοκληρωθεί και προγραμματίζονται υπό την προϋπόθεση οτι δεν προκύπτει καμιά σύγκρουση όσον αφορά τους πόρους τη δεδομένη στιγμή.

Ειδικότερα, για κάθε στάδιο $g$ υπαρχει ένας χρόνος προγράμματος $t_g$. Όσες δραστηριότητες έχουν προγραμματιστεί ως το στάδιο $g$ ανήκουν είτε στο ολοκληρωμένο σύνολο $C_g = \{j | f_j \leq t_g\}$, είτε στο ενεργό σύνολο $A_g = Act(t_g) = \{j | f_j - d_j \leq t < f_j\}$ του σταδίου $g$. Το επιλέξιμο σύνολο $D_g = \left\{j \notin (C_g \cup A_g) | (Pred(j) \subset C_g) \cap \left(r_{jk} \leq \tilde{R}_k(t_g)\right)\right\}$ αποτελείται από όλες τις δραστηριότητες που είναι εφικτές από άποψη περιορισμών σε πόρους και σχέσεων προτεραιότητας να αρχίσουν σε χρονο $t_g$ και $\tilde{R}_k(t_g) = R_k - \sum_{j \in A_g} r_{jk}$ είναι η εναπομείνασα διαθεσιμότητα του πόρου τύπου $k$ τη χρονική στιγμή $t_g$. Στν Αλγόριθμο 3.2 που ακολουθεί παρουσιάζεται ο αλγόριθμος του PSGS:

**Αλγόριθμος 3.2 Μέθοδος παράλληλης παραγωγής χρονοπρογραμμάτων (pSGS)**

---

$g = 0$, $t_g = 0, A_0 = \{0\}, C_0 = \{0\}$, $\tilde{R}_k(0) = R_k$;
while $|A_g \cup C_g| \leq n$ do
    $g = g + 1$ ;
    $t_g = min_{j \in A_g}\{f_j\}$;
    Calculate $C_g$, $A_g, \tilde{R}_k(t_g), D_g$ ;
    while $D_g \neq 0$ do
        Select $j \in D_g$ ;
        $f_j = t_g + d_j$ ;
        Calculate $A_g, \tilde{R}_k(t_g), D_g$ ;
    end
end
$f_{n+1} = max_{h \in Pred(n+1)}\{f_h\}$

---

Έχει αποδειχθεί οτι κάθε πρόγραμμα που παράγεται από τη μέθοδο παράλληλης παραγωγής προγραμμάτων ανήκει στο σύνολο των προγραμμάτων μη-καθυστέρησης, που είναι προγράμματα όπου, ακόμα κι αν επιτρέπεται η διακοπή κατά την εκτέλεση των δραστηριοτήτων, καμιά από τις δραστηριότητες δεν μπορεί να αρχίσει νωρίτερα χωρίς να καθυστερήσει κάποια άλλη δραστηριότητα (Kolisch, 1996). Το σύνολο των προγραμμάτων μη-καθυστέρησης είναι υποσύνολο του συνόλου των ενεργών προγραμμάτων, έχει όμως το μειονέκτημα οτι μπορεί να μην περιέχει τη βέλτιστη λύση για ένα πρόβλημα προγραμματισμού με ένα κανονικό μέτρο απόδοσης.

Για δεδομένη λίστα προτεραιότητας, και η σειριακή μέθοδος (SSGS) και η μέθοδος παράλληλης παραγωγής προγραμμάτων (PSGS) απαιτούν χρόνο $O(n^2k)$ (Pinson et al., 1994, Kolisch and Hartmann, 1999).

*Κανόνες προτεραιότητας*

«Ένας κανόνας προτεραιότητας είναι ένα πλάνο που εκχωρεί σε κάθε δραστηριότητα $j$ στο σύνολο απόφασης $D_g$ μια αξία $v(j)$ και μια δήλωση στόχου εάν η δραστηριότητα που επιλέγεται είναι εκείνη με τη μέγιστη ή την ελάχιστη αξία» (Kolisch and Hartmann, 1999). Στην περίπτωση ισότητας η διαφορά λύνεται με διάφορους κατάλληλα ορισμένους κανόνες.

Υπάρχουν δεκάδες τέτοιοι κανόνες στη βιβλιογραφία. Μια άποψη των πιο συχνά χρησιμοποιούμενων κανόνων προτεραιότητας και των μαθηματικών τους διατυπώσεων δίνεται στον Πίνακα 3.1: μικρότερη διάρκεια εκτέλεσης (shortest processing time-SPT), μέγιστος αριθμός άμεσων και έμμεσων διαδόχων $Suc_j$ της δραστηριότητας $j$ (most total successors-MTS), μέγιστη διάρκεια δραστηριότητας συμπεριλαμβανομένων των άμεσων διαδόχων της (greatest rank positional weight-GRPW), αργότερος χρόνος λήξης (latest finish time-LFT), αργότερος χρόνος έναρξης (latest start time-LST), μικρότερο περιθώριο (minimum slack-MSLK).

**Πίνακας 3.1 Κανόνες προτεραιότητας**

| Κανόνας | $\boldsymbol{V(j)}$ | Αναφορά | Στόχος |
|---|---|---|---|
| ***GPRW*** | $d_j + \sum_{i \in Succ_j} d_i$ | (Alvarez-Valdes and Tamarit, 1989) | Μεγιστοποίηση |
| ***LFT*** | $LF_J$ | (Davis and Patterson, 1975) | Ελαχιστοποίηση |
| ***LST*** | $LF_J - d_j$ | (Kolisch, 1995) | Ελαχιστοποίηση |
| ***MSLK*** | $LF_J - EF_J$ | (Davis and Patterson, 1975) | Ελαχιστοποίηση |
| ***MTS*** | $\lvert Succ_j \rvert$ | (Alvarez-Valdes and Tamarit, 1989) | Μεγιστοποίηση |
| ***SPT*** | $d_j$ | (Alvarez-Valdes and Tamarit, 1989) | Ελαχιστοποίηση |

### *3.5.3 Μετα-Ευρετικοί Αλγόριθμοι*

Οι μετα-ευρετικές προσεγγίσεις του RCPSP δεν επενεργούν σε προγράμματα, αλλά σε αναπαραστάσεις προγραμμάτων, οι οποίες έπειτα μετατρέπονται σε πρόγραμμα μέσω μιας διαδικασίας αποκωδικοποίησης. Επομένως, οι «χειριστές» (operators) που διαθέτουν οι αλγόριθμοι για την παραγωγή λύσεων, πρέπει να λαμβάνουν υπόψη την επιλεγμένη αναπαράσταση. Οι χειριστές αυτοί είναι είτε μοναδιαίοι, παράγουν δηλαδή μια νέα λύση από μια ήδη υπαρχουσα, είτε δυαδικοί,

παράγουν δηλαδή μια νέα λύση από δύο ήδη υπαρχουσες, όπως η διαδικασία της διασταύρωσης (crossover) των γενετικών αλγορίθμων (genetic algorithms-GA). Οι πιο συχνά χρησιμοποιούμενη αναπαράσταση για το RCPSP είναι η *λίστα δραστηριοτήτων (activity list)*, όπου δίνεται μια λίστα εφικτών από περιορισμούς προτεραιότητας δραστηριοτήτων, στην οποία κάθε δραστηριότητα πρέπει να έχει δείκτη μεγαλύτερο από αυτό των προκατόχων της. Άλλες αναπαραστάσεις που χρησιμοποιούνται είναι η *αναπαράσταση τυχαίου κλειδιού προτεραιότητας (random key of priority representation),* η *αναπαράσταση με κανόνες προτεραιότητας (priority rule representation)* (Hartmann, 1998) και η *αναπαράσταση με διάνυσμα μετατόπισης (shift vector representation).*

### *3.5.3.1 Γενετικοί Αλγόριθμοι (Genetic Algorithms)*

Οι Γενετικοί Αλγόριθμοι ανήκουν στην κατηγορία συστημάτων επίλυσης προβλημάτων που είναι ευρύτερα γνωστή με τον όρο Εξελικτικοί Αλγόριθμοι (Evolutionary Algorithms). Παρουσιάστηκαν πρώτη φορά από τον Holland (1975), και είναι εμπνευσμένοι από τη διαδικασία της βιολογικής εξέλιξης. Ένας Γενετικός Αλγόριθμος είναι μια τεχνική επίλυσης βασισμένη στις έννοιες της εξέλιξης και της κληρονομικότητας, που ταιριάζει στην περίπτωση σύνθετων προβλημάτων με μεγάλα πεδία λύσεων λόγω του εγγενούς παραλληλισμού του που επιτρέπει την αποδοτική εξερεύνηση των πεδίων αυτών. Η βασική ιδέα είναι οτι παράγεται μια ομάδα αρχικών λύσεων, οι οποίες στη συνέχεια δουλεύονται επαναληπτικά με στόχο τη βελτίωσή τους. Αυτή η ομάδα λύσεων ονομάζεται πληθυσμός (population) και η αρχική ομάδα ονομάζεται αρχικός πληθυσμός. Οι αρχικές λύσεις παράγονται με τυχαίο τρόπο από τον αλγόριθμο για τη δημιουργία του αρχικού πληθυσμού, κατάλληλα κωδικοποιημένες με χρήση μιας εκ των αναπαραστάσεων που περιγράφηκαν προηγουμένως, όπως τα χρωμοσώματα (chromosomes) δραστηριοτήτων. Έπειτα οι νέοι πληθυσμοί (γόνοι) παράγονται επαναληπτικά με συνδυασμό των χρωμοσωμάτων του τρέχοντα πληθυσμού με βάση κανόνες συνδυασμού χρωμοσωμάτων (διασταύρωση-crossover), με τυχαίες αλλαγές σε κομμάτια των χρωμοσωμάτων (μετάλλαξη-mutation) και με επιλογή των χρωμοσωμάτων που θα περάσουν στην επόμενη γενιά λύσεων με χρήση μιας πολιτικής επιλογής (μέθοδος επιλογής-selection method). Ο δείκτης καταλληλότητας (fitness value) μετρά την αξία μιας λύσης, συνήθως με βάση το αποτέλεσμα της αντικειμενικής συνάρτησης του προβλήματος βελτιστοποίησης που μελετάται. Στον Αλγόριθμο 3.3 που ακολουθεί παρουσιάζεται ένας κλασικός γενετικός αλγόριθμος.

**Αλγόριθμος 3.3: Γενετικός Αλγόριθμος**

set $Population\ Size = POP$;
set $Crossover\ Type$; set $Mutation\ Probability$;
set $Generation\ Counter\ g = 0$;
Generate $Initial\ Population\ P_g$;
**while** $stopping\ criteria\ not\ met$ **do**
    Evaluate $P_g$;
    Crossover $P_g$ and get $P_{children}$;
    Mutate $P_{children}$;
    Evaluate $P_{children}$;
    Select from $P_g$ and $P_{children}$ and form $P_{g+1}$;
    $g = g + 1$;
**end**

Μελετώντας τις εφαρμογές των Γενετικών Αλγορίθμων (GA) στο RCPSP (Hartmann, 1998, Hartmann, 2002, Kim et al., 2003, Cervantes et al., 2008, Mendes et al., 2009, Montoya-Torres et al., 2010, Wang et al., 2010, Proon and Jin, 2011) μπορεί να εξαχθεί το συμπέρασμα οτι αν και είναι ιδιαίτερα αποδοτικές διαδικασίες στο να βρίσκουν ολικά βέλτιστα ή κοντά στο βέλτιστο αποτελέσματα, οι κατάλληλοι ορισμοί της αναπαράστασης των δραστηριοτήτων, της συνάρτησης καταλληλότητας, της διασταύρωσης, της μετάλλαξης και των μεθόδων επιλογής αποτελούν καθοριστικούς παράγοντες για την αποδοτικότητα και την αποτελεσματικότητα του αλγορίθμου. Βαθύτερη ανάλυση των χαρακτηριστικών των χειριστών του αλγορίθμου και πειραματικές τους συγκρίσεις μπορούν να βρεθούν στη βιβλιογραφία (Hartmann and Kolisch, 2000).

#### *3.5.3.2 Προσομοιωμένη Ανόπτηση (Simulated Annealing)*

Η Προσομοιωμένη Ανόπτηση (SA) είναι μια μέθοδος εμπνευσμένη από την φυσική διαδικασία ανόπτησης των μετάλλων. Σε αυτήν τη διαδικασία, ένα μέταλλο θερμαίνεται πάνω από μια συγκεκριμένη θερμοκρασία, διατηρείται σε αυτή την κατάσταση για ένα ορισμένο χρονικό διάστημα και έπειτα ψύχεται με στόχο την ελαχιστοποίηση των εσωτερικών ατελειών στη δομή των κόκκων του πλέγματος του μετάλλου. Η αναλογία μεταξύ των συνδυαστικών προβλημάτων και της διαδικασίας αυτής παρατηρήθηκε και παρουσιάστηκε πρώτη φορά το 1983 (Kirkpatrick et al., 1983) σαν μέθοδο απεγκλωβισμού από τοπικά βέλτιστα κατά την επίλυση.

Η διαδικασία αρχίζει με μια αρχική λύση που λειτουργεί ως βάση για την παραγωγή της λεγόμενης «γειτονιάς» (neighborhood) λύσεων μέσα από ελαφρές μεταβολές της αρχικής λύσης. Μια νέα λύση γίνεται δεκτή και χρησιμοποιείται για να προχωρήσει η διαδικασία όταν είναι καλύτερη από την παρούσα λύση. Μπορεί όμως να γίνει αποδεκτή και αν είναι χειρότερη από την υπάρχουσα, με βάση κάποια πιθανότητα. Αυτή η πιθανότητα αποδοχής (acceptance probability) εξαρτάται από την θερμοκρασία ψύξης, η οποία είναι μια παράμετρος με τιμή που αρχικά είναι υψηλή ώστε να επιτρέπει την αποδοχή ενός μεγάλου ποσοστού των παραγόμενων λύσεων, στη συνέχεια όμως μειώνεται σταδιακά ώστε να περιορίζει βαθμιαία το ποσοστό αποδοχής των λιγότερα υποσχόμενων λύσεων. Αυτό δεν επιτρέπει στον αλγόριθμο να παγιδευτεί σε τοπικά βέλτιστα κατά τα πρώτα στάδια εκτέλεσής του. Ο αλγόριθμος σταματά μόλις το κριτήριο τερματισμού που έχει τεθεί φτάσει την προκαθορισμένη τιμή του. Με τον αλγόριθμο SA επιτυγχάνεται αφ' ενός μεν υψηλή εξερεύνηση της περιοχής των λύσεων στις αρχικές επαναλήψεις, που η θερμοκρασία είναι υψηλή και ο αλγόριθμος απεγκλωβίζεται από τα τοπικά ακρότατα, αφ' ετέρου δε υψηλή εκμετάλλευση των αξιόλογων λύσεων στα τελικά στάδια, όσο μειώνεται δηλαδή η θερμοκρασία, με στόχο την κατάληξη την περιοχή της βέλτιστης λύσης.

Στη συνέχεια παρουσιάζεται στον Αλγόριθμο 3.4 εν συντομία ο αλγόριθμος προσομειωμένης ανόπτησης των Bouleimen και Lecocq (2003). Η αναπαράσταση της λύσης που χρησιμοποιείται σε αυτή την προσέγγιση είναι η λίστα δραστηριοτήτων και για την κατασκευή της χρησιμοποιείται η σειριακή μέθοδος παραγωγής χρονοπρογραμμάτων (SSGS). Κάθε γενιά «γειτονιάς» λύσεων παράγεται από την τρέχουσα επίλυση και μια τυχαία επιλεγμένη δραστηριότητα. Υπολογίζονται οι θέσεις $lp$ της αργύτερης προκατόχου (latest predecessor) και $es$ της νωρίτερης δραστηριότητας που ακολουθεί (earliest successor) την επιλεγμένη. Έπειτα η νέα θέση της επιλεγμένης δραστηριότητας επιλέγεται τυχαία μεταξύ των δύω θέσεων $[lp, es]$. Η γειτονιά αποκτάται από μια κυκλική ανακατάταξη των δραστηριοτήτων που βρίσκονται μεταξύ της παλιάς και της νέας θέσης της επιλεγμένης δραστηριότητας.

**Αλγόριθμος 3.4 Αλγόριθμος προσομειωμένης ανόπτησης (Bouleimen and Lecocq, 2003)**

Calculate initial solution $x_0$ and fitness $f(x_0)$;
$x_{best} = x_0;\ f_{best} = f(x_0);$
$x_{current} = x_0;\ f_{current} = f(x_0);$
**for** $C\ chains$ **do**
    $T = T_{0_{max}};$
    $N_s = N_0;$
    **for** $S\ steps$ **do**
        $N_s = N_s(1 + h * s);$
        **for** $N_s\ neighbourhoods$ **do**
            Generate neighbour $x' of\ x_{current};$
            Calculate $f(x')$ and $\Delta = f(x') = f(x);$
            **if** $\Delta < 0$ **then**
                $x_{current} = x';\ f(x_{current} = f(x');$
                **if** $f(x') < f_{best}$ **then**
                    $x_{best} = x';\ f(x_{best)} = f(x');$
                **if** $f_{best} = CP\ value$ **then** EXIT;
            **else**
                **if** $\mathrm{P} = \mathrm{e}^{-\frac{\Delta}{T}} > y_{random}$ **then**
                    $x_{current} = x';\ f(\ x_{current}) = f(x');$
            **end**
        **end**
        $T = a^5 * T;$
    **end**
**end**
Explore neighboorhood for $f_{best}$;

Ο μηχανισμός ψύξης συνίσταται από μια ψυχόμενη αλυσίδα $C$ η οποία επανεκκινείται κάθε φορά που μια διαφορετική αρχική λύση ελέγχεται. Ο αριθμός των γειτονικών τέστ σε κάθε βήμα $s$ της αλυσίδας αυξάνεται προοδευτικά ως $N_s = N_s(1 + h * s)$ όπου ως $h$ ορίζεται το μήκος του βήματος που κάφε φορά πραγματοποιείται. Η θερμοκρασία $T$ μειώνεται σε $S$ βήματα, ξεκινώντας από μια αρχική τιμή $T_0$ η οποία πρέπει να είναι αρκετά ψηλή ώστε να επιτρέπει την αποδοχή κάθε νέας γειτονιάς στα πρώτα βήματα του αλγορίθμου και χρησιμοποιώντας ένα

συντελεστή απόσβεσης $\alpha, 0 < \alpha < 1$. Σε κάθε βήμα $s$, η διαδικασία παράγει ένα συγκεκριμένο αριθμό γειτονικών λύσεων, τις οποίες έπειτα αξιολογεί χρησιμοποιώντας την τρέχουσα τιμή της θερμοκρασίας $T_s = a^5 * T_0$.

#### *3.5.3.3 Αναζήτηση Ταμπού (Taby Search)*

Η αναζήτηση ταμπού (TS) αναπτύχθηκε από τον Glover (Glover, 1989) και μπορεί να οριστεί σαν μέθοδος απότομης καθόδου/ομαλής ανόδου. Πρόκειται για μια μέθοδο που προσπαθεί να ξεπεράσει τον εγκωβισμό σε τοπικά ακρότατα της αντικειμενικής συνάρτησης (Glover and Laguna, 1997).

Ο αλγόριθμος ξεκινά από μια λύση που χρησιμοποιείται για την δημιουργία της γειτονιάς και έπειτα αξιολογούνται όλες οι λύσεις που έχουν παραχθεί, επιλέγεται η καλύτερη και χρησιμοποιείται στην επόμενη επανάληψη. Επειδή αυτή η διαδικασία μπορεί να οδηγήσει σε κυκλικές κινήσεις γύρω από ένα τοπικό ακρότατο, ένας αριθμός προηγούμενων λύσεων αποθηκεύεται σε μια μνήμη της μορφής δομής δεδομένων (data-structure), τη λεγόμενη *λίστα Ταμπού*. Η λίστα Ταμπού χρησιμοποιείται για απόρριψη επαναλαμβανόμενων κινήσεων του αλγορίθμου που μπορεί να οδηγήσουν σε προηγουμένως υπολογισμένη λύση. Ο χαρακτηρισμός Ταμπού μπορεί να αγνοηθεί μόνο στην περίπτωση που η προτεινόμενη κίνηση από τον αλγόριθμο μπορεί να οδηγήσει σε μια νέα συνολικά καλύτερη λύση με βάση τον λεγόμενο κανόνα φιλοδοξίας (aspiration rule) (Nonobe and Ibaraki, 2002).

Στον Αλγόριθμο 3.5 που ακολουθεί περιγράφεται ένας βασικός αλγόριθμος αναζήτησης ταμπού για το RCPSP.

**Αλγόριθμος 3.5 Αλγόριθμος Αναζήτησης Ταμπού**

Generate Initial Solution $x_0$ and calculate $f(x_0)$;
Initialize Tabu List;
$x_{best} = x_0;\ f_{best} = f(x_0);$
$x_{current} = x_0;\ f_{current} = f(x_0);$
**while** $stopping\ criteria\ not\ met$ **do**
    Generate $Moves(x_0)$ list of candidate moves;
    **while** $move\ not\ effectuated$ **do**
        $Select\ best\ move\ \ M(x');$
        **If** $M(x') \notin Tabu\ List$ ***or*** $M(x')\ meets\ Aspiration\ Criteria$ **then**
            Execute move $M(x')$;
            $x_{current} = x';\ f_{current} = f(x');$
            Update $Tabu\ List$;
            Update $Aspiration\ Criteria$;
        **end**
    **end**
**end**

Η χρήση της Αναζήτησης Ταμπού στο RCPSP δεν είναι τόσο εκτεταμένη όσο των Γενετικών Αλγορίθμων ή της Προσωμοιομένης Ανόπτησης, έχει όμως δώσει συχνά πολύ καλά αποτελέσματα (Thomas and Salhi, 1998, Klein, 2000, Nonobe and Ibaraki, 2002).

## 3.6 Βελτιστοποίηση Αποικίας Μυρμηγκιών *(Ant Colony Optimization)*

Ο πρώτος αλγόριθμος βελτιστοποίησης με αποικία μυρμηγκιών (ACO) προτάθηκε από τον Marco Dorigo και τους συνεργάτες του στις αρχές της δεκαετίας του ’90 (Dorigo, 1992, Colorni et al., 1991, Dorigo et al., 1991, Colorni et al., 1994, Dorigo et al., 1996), είναι δηλαδή μια από τις πιο πρόσφατα ανεπτυγμένες μεθόδους βελτιστοποίησης. Ανάλογα με την οπτική του καθενός, οι ACO αλγόριθμοι μπορεί να ανήκουν σε διαφορετικές κατηγορίες προσεγγιστικών αλγορίθμων.

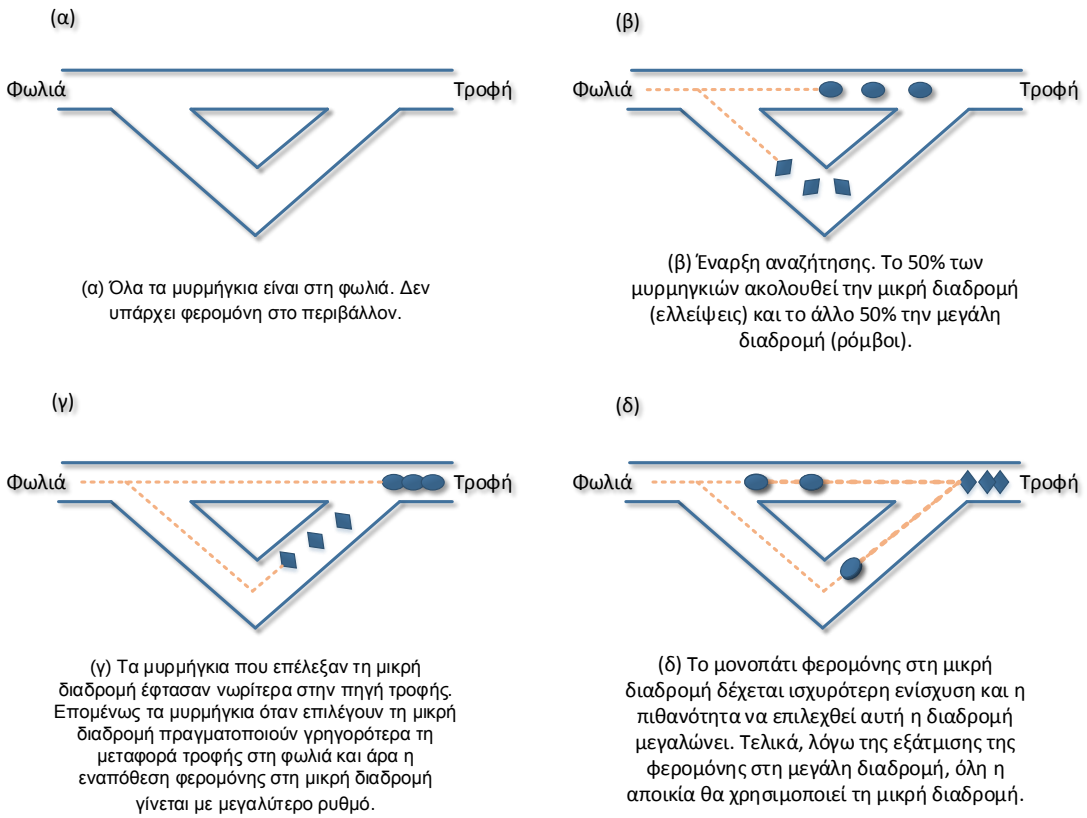
Από την οπτική της τεχνητής νοημοσύνης (artificial intelligence-AI), οι αλγόριθμοι ACO είναι ένας από τους πιο επιτυχημένους κλάδους της νοημοσύνης των σμηνών (swarm intelligence-SI) (Dorigo, 2006). Στην SI, η βάση στην οποία στηρίζεται η διαδικασία επίλυσης είναι η ενδοεπικοινωνία μεταξύ των πρακτόρων (agents) που αποτελούν το σμήνος και η παράλληλη επικοινωνία τους με το

περιβάλλον. Τα σμήνη (swarms) χαρακτηρίζονται από δύο δράσεις (Jaiswal and Aggarwal, 2011):

i. την αυτο-οργάνωση (self-organization), δηλαδή την ανάδραση (θετική και/ή αρνητική) μεταξύ των πρακτόρων, την ενίσχυση των τυχαίων φαινομένων (τυχαίες κινήσεις, λάθη) και την πολλαπλή επικοινωνία μεταξύ των ατόμων, και
ii. τη στιγμέργεια (stigmergy), δηλαδή την έμμεση επικοινωνία μεταξύ των πρακτόρων μέσω της αλληλεπίδρασης με το περιβάλλον.

Από την οπτική της επιχειρησιακής έρευνας (OR) όμως, οι αλγόριθμοι ACO ανήκουν στην κατηγορία των μετα-ευρετικών αλγορίθμων (υποσύνολο των εξελικτικών αλγορίθμων), μαζί με τους γενετικούς αλγορίθμους, τους αλγορίθμους προσομοιωμένης ανόπτησης και τους αλγορίθμους ταμπού αναζήτησης.

Αναπτύχθηκαν μετά από παρατήρηση της λειτουργίας των αποικιών των μυρμηγκιών. Τα μυρμήγκια είναι κοινωνικά έντομα, ζουν σε αποικίες και η συμπεριφορά τους επικεντρώνεται στην επιβίωση και εξέλιξη της αποικίας ως σύνολο και όχι κάθε ατόμου ξεχωριστά. Ειδικότερα, η έμπνευση των ACO αλγορίθμων προήλθε από την συμπεριφορά των μυρμηγκιών κατά την αναζήτηση τροφής και τον τρόπο με τον οποίο τα μυρμήγκια βρίσκουν το συντομότερο δρόμο ανάμεσα στην τροφή και την αποικία τους. Αρχικά πραγματοποιούν τυχαία αναζήτηση στο χώρο γύρω από την αποικία τους. Κατά την κίνησή τους αποελευθερώνουν ένα ίχνος χημικής φερομόνης στο έδαφος. Τα μυρμήγκια μπορούν να ανιχνεύσουν τη φερομόνη και έχουν την τάση να κινούνται επιλέγοντας τις οδούς με τη μεγαλύτερη συγκέτρωση φερομόνης. Μόλις ένα μυρμήγκι βρει τροφή την αξιολογεί ως προς την ποσότητα, την ποιότητα και την απόσταση από την φωλιά και μεταφέρει μια ποσότητα αυτής πίσω στην φωλιά. Κατά την επιστροφή του, το ίχνος φερομόνης που εναποθέτει στη διαδρομή του είναι ανάλογο της αξιολόγησης που έκανε στην τροφή. Τα ίχνη αυτά οδηγούν τα υπόλοιπα μυρμήγκια στο να ακολουθήσουν ή όχι τη διαδρομή αυτή και έτσι τα μυρμήγκια βρίσκουν τις ελάχιστες αποστάσεις μεταξύ των πηγών τροφής και της αποικίας τους. Στο Σχήμα 3.14 που ακολουθεί αναπαριστάται η διαδικασία εύρεσης τροφής από τα μυρμήγκια. Τίθεται ένα πειραματικό σενάριο που παρουσιάζει την ικανότητα των αποικιών μυρμηγκιών να βρίσκουν την μικρότερη διαδρομή. Μεταξύ της φωλιάς των μυρμηγκιών και της μοναδικής πηγής τροφής υπάρχουν δύο διαδρομές με διαφορετικά μήκη. Στα τέσσερα γραφήματα, τα ίχνη φερομόνης απεικονίζονται ως διακεκομμένες γραμμές των οποίων το πάχος είναι ανάλογο της ισχύος τους, δηλαδή της συγκέντρωσης φερομόνης σε αυτά.




**Σχήμα 3.14 Αναπαράσταση της ικανότητας εύρεσης της μικρότερης διαδρομής από αποικίες μυρμηγκιών**

Για την κατανόηση του ACO αλγορίθμου που μοντελοποιεί αυτό το φαινόμενο, γίνεται σε πρώτο στάδιο η ανάλυση ενός απλοποιημένου μοντέλου του φαινομένου. Το μοντέλο αποτελείται από ένα γράφο $G = (V, E)$, όπου το σύνολο κόμβων $V$ αποτελείται από δύο κόμβους $u_s$ και $u_d$, που αναπαριστούν την φωλιά των μυρμηγκιών και την πηγή τροφής αντίστοιχα, ενώ το σύνολο ακμών $E$ αποτελείται από δύο ακμές $e_1$ και $e_2$ μεταξύ των κόμβων $u_s$ και $u_d$. Στην ακμή $e_1$ εκχωρείται μήκος $l_1$ και στην ακμή $e_2$ εκχωρείται μήκος $l_2$, έτσι ώστε $l_2 > l_1$. Επομένως, η $e_1$ αναπαριστά τη μικρότερη και η $e_2$ τη μεγαλύτερη διαδρομή μεταξύ $u_s$ και $u_d$. Τα αληθινά μυρμήγκια εναποθέτουν φερομόνη στις διαδρομές που ακολουθούν. Τα ίχνη χημικής φερομόνης μοντελοποιούνται με την εισαγωγή τεχνητής φερομόνης $\tau$ με διαφορετική τιμή $\tau_i$ για κάθε ένα από τους δύο συνδέσμους $e_i$, $i = 1,2$. Η τιμή αυτή υποδεικνύει την δύναμη του ίχνους φερομόνης στην αντίστοιχη διαδρομή. Τέλος, εισάγονται $n_\alpha$ τεχνητά μυρμήγκια. Κάθε μυρμήγκι, ξεκινώντας από τον κόμβο $u_s$ (δηλαδή τη φωλιά) επιλέγει με πιθανότητα $p_i = \tau_i/\tau_1 + \tau_2$, $i = 1,2$ μια εκ των διαδρομών $e_1$ και $e_2$ για να φτάσει στον κόμβο $u_d$ (δηλαδή την πηγή τροφής).

Προφανώς, αν $\tau_1 > \tau_2$ η πιθανότητα επιλογής της $e_1$ είναι υψηλότερη και το αντίστροφο. Κατά την επιστροφή του από το $u_d$ στο $u_s$ ένα μυρμήγκι ακολουθεί το ίδιο μονοπάτι που ακολούθησε για να φτάσει στο $u_d$ και αλλάζει την τιμή της τεχνητής φερομόνης που αντιστοιχεί στη συγκεκριμένη ακμή. Αναλυτικότερα, έχοντας επιλέξει την ακμή $e_i$, ένα μυρμήγκι αλλάζει την τιμή της τεχνητής φερομόνης $\tau_i$ ακολούθως: $\tau_i \leftarrow \tau_i + Q/l_i$ όπου η θετική σταθερά $Q$ είναι μια παράμετρος του μοντέλου. Αυτό σημαίνει οτι η ποσότητα τεχνητής φερομόνης που προστίθεται εξαρτάται από το μήκος της επιλεγμένης διαδρομής: όσο πιο σύντομη η διαδρομή τόσο υψηλότερη η ποσότητα της προστιθέμενης φερομόνης.

Σε αυτό το μοντέλο η συμπεριφορά αναζήτησης τροφής μιας αποικίας μυρμηγκιών μοντελοποιείται επαναληπτικά με τον εξής τρόπο: Σε κάθε βήμα (επανάληψη) όλα τα μυρμήγκια είναι αρχικά τοποθετημένα στον κόμβο $u_s$. Έπειτα κάθε μυρμήγκι κινείται απο το $u_s$ στο $u_d$ όπως περιγράφηκε παραπάνω. Όπως αναφέρθηκε πιο πριν, στη φύση η αποθκευμένη φερομόνη εξατμίζεται με την πάροδο του χρόνου. Η εξάτμιση της φερομόνης μοντελοποιείται στο τεχνητό μοντέλο ως εξής: $\tau_i \leftarrow (1-\rho)\cdot\tau_i,\ i = 1{,}2$. Η παράμετρος $\rho \in (0{,}1]$ είναι εκείνη που ρυθμίζει την εξάτμιση της φερομόνης. Τέλος, όλα τα μυρμήγκια διεξάγουν το ταξίδι της επιστροφής τους και ενισχύουν το μονοπάτι που επέλεξαν με τον τρόπο που περιγράφηκε νωρίτερα. Η όλη διαδικασία παρουσιάζεται στο σχήμα που ακολουθεί.

Στους ACO αλγορίθμους τα τεχνητά μυρμήγκια μοντελοποιούνται με τις εξής διαφορές από τα αληθινά (Blum, 2005):

- Τα αληθινά μυρμήγκια κινούνται ασύγχρονα στο περιβάλλον τους ενώ τα τεχνητά είναι συγχρονισμένα, δηλαδή σε κάθε επανάληψη του αλγορίθμου κάθε ένα από αυτά κινείται ταυτόχρονα από τη φωλιά προς την πηγή τροφής και πίσω από τον ίδιο δρόμο.
- Τα αληθινά μυρμήγκια αφήνουν φερομόνη σε κάθε τους μετακίνηση ενώ τα τεχνητά εναποθέτουν φερομόνη μόνο κατά την επιστροφή τους στην φωλιά.
- Η συμπεριφορά εύρεσης τροφής των αληθινών μυρμηγκιών βασίζεται σε μια έμμεση αξιολόγηση της λύσης (της απόστασης από την τροφή ως τη φωλιά), με την έννοια οτι τα μικρότερα μονοπάτια ολοκηρώνονται συντομότερα από τα μακρύτερα επομένως λαμβάνουν μεγαλύτερη και ταχύτερη ενίσχυση σε φερομόνη. Αντίθετα, τα τεχνητά μυρμήγκια αξιολογούν μια λύση σύμφωνα με κάποιο μέτρο απόδοσης και ποιότητας το οποίο καθορίζει την ποσότητα της φερομόνης με την οποία θα ενισχύσουν τα μυρμήγκια το μονοπάτι κατά την επιστροφή τους στη φωλιά.

### *3.6.1 Ο πρώτος ACO αλγόριθμος: Ant System*

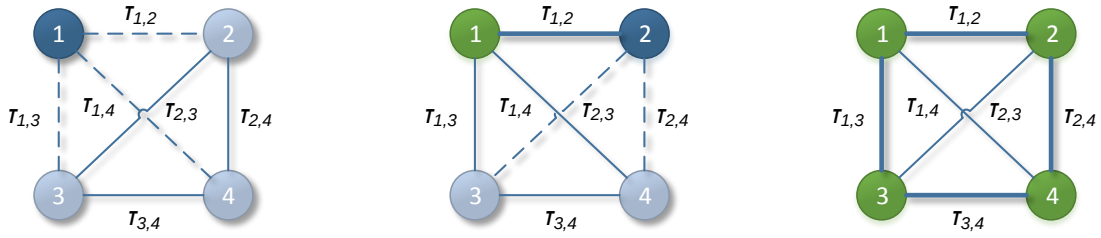
Το μοντέλο που περιγράφηκε στο προηγούμενο κομμάτι για την αναπαράσταση της συμπεριφοράς εύρεσης τροφής των αληθινών μυρμηγκιών δεν μπορεί να εφαρμοστεί άμεσα σε προβλήματα συνδυαστικής βελτιστοποίησης. Αιτία αποτελεί το γεγονός ότι συσχετίστηκαν οι τιμές της φερομόνης απευθείας με τις λύσεις του προβλήματος (δηλαδή μια παράμετρος με τη σύντομη διαδρομή και μια με την μακρύτερη διαδρομη). Με τον τρόπο αυτό η μοντελοποίηση ουσιαστικά θεωρεί οτι οι λύσεις για το πρόβλημα που αντιμετωπίζεται είναι γνωστές. Όμως στα προβλήματα συνδυαστικής βελτιστοποίησης, επιδιώκεται η *εύρεση* μιας άγνωστης βέλτιστης λύσης. Συνεπώς οι τιμές της φερομόνης αντιστοιχίζονται με τμήματα λύσεων και όχι με ολόκληρες λύσεις. Τα τμήματα λύσεων είναι τα κομμάτια με συνδυασμό των οποίων συναρμολογούνται οι λύσεις του προβλήματος που αντιμετωπίζεται. Γενικά, το σύνολο των τμημάτων μιας λύσης αναμένεται να είναι πεπερασμένο και μετρίου μεγέθους. Σαν παράδειγμα παρουσιάζεται ο πρώτος ACO αλγόριθμος που καλείται Ant System (AS) (Dorigo, 1992, Dorigo et al., 1996) και εφαρμόστηκε στο πρόβλημα του περιοδεύοντος πωλητή (travelling salesman problem - TSP), του οποίου ο ορισμός είναι ο εξής: «Στο πρόβλημα TSP δίνεται ένας πλήρως συνδεδεμένος, μη προσανατολισμένος γράφος $G = (V, E)$ με ακμές-βάρη. Το σύνολο των κόμβων $V$ του γράφου αναπαριστά τις πόλεις και οι ακμές-βάρη αναπαριστούν τις απιστάσεις μεταξύ των πόλεων. Στόχος είναι να βρεθεί μια κλειστή διαδρομή στον G που να περιέχει κάθε κόμβο ακριβώς μια φορά (που καλείται ταξίδι) και της οποίας το μήκος είναι το ελάχιστο δυνατό. Συνεπώς ο χώρος αναζήτησης $S$ αποτελείται από όλα τα ταξίδια στον $G$. Η τιμή της αντικειμενικής συνάρτησης $f(s)$ ενός ταξιδιού $s \in S$ ορίζεται ως το σύνολο όλων των ακμών-βαρών των ακμών του $s$. Το TSP μπορεί να μοντελοποιηθεί με πολλούς διαφορετικούς τρόπους ως πρόβλημα διακριτής βελτιστοποίησης. Το πιο σύνηθες μοντέλο αποτελείται από μια δυαδική μεταβλητή απόφασης $X_e$ για κάθε ακμή $e$ στον $G$. Αν σε μια λύση είναι $X_e = 1$ τότε η ακμή $e$ είναι κομμάτι του ταξιδιού που ορίζεται από τη συγκεκριμένη λύση».

Όσον αφορά την προσέγγιση του AS, οι ακμές του δεδομένου TSP γράφου μπορούν να θεωρηθούν ως τμήματα λύσης, δηλαδή για κάθε $e_{i,j}$ εισάγεται μια τιμή φερομόνης $\tau_{i,j}$. Αποστολή κάθε μυρμηγκιού είναι η κατασκευή μιας εφικτής λύσης για το TSP, δηλαδή ένα εφικτό ταξίδι. Με άλλα λόγια, η έννοια της *αποστολής κάθε μυρμηγκιού* αλλάζει από «την επιλογή μιας διαδρομής από την φωλιά στην πηγή τροφής» σε «κατασκευή μιας εφικτής λύσης στο πρόβλημα βελτιστοποίησης που αντιμετωπίζεται». Με την αλλαγή αυτή, οι έννοιες της φωλιάς και της πηγής τροφής χάνουν το νόημά τους.

Κάθε μυρμήγκι κατασκευάζει μια λύση με τον ακόλουθο τρόπο. Αρχικά ένας από τους κόμβους του TSP γράφου επιλέγεται τυχαία ως κόμβος αρχής. Έπειτα το μυρμήγκι κατασκευάζει ένα ταξίδι στον TSP γράφο, κινούμενο σε κάθε βήμα από τον τρέχοντα κόμβο (την πόλη στην οποία βρίσκεται) σε άλλο κόμβο, τον οποίο δεν έχει προηγουμένως επισκεφθεί. Σε κάθε βήμα η ακμή που διανύεται προστίθεται στην λύση υπό κατασκευή. Αυτή η διαδικασία κατασκευής λύσης συνεπάγεται οτι ένα μυρμήγκι έχει μια μνήμη $T$ όπου αποθηκεύει τους κόμβους που έχει ήδη επισκεφθεί. Πιο αναλυτικά, υποθέτοντας οτι το μυρμήγκι είναι στον κόμβο $v_i$ το ακόλουθο βήμα πραγματοποιείται με πιθανότητα

$$\boldsymbol{p}(e_{i,j}) = \frac{\tau_{i,j}}{\sum_{\{k\in\{1,\dots,|V|\}|\ v_k\notin T\}}\tau_{i,k}}\ ,\forall j \in \{1,\dots,|V|\}, v_j \notin T \tag{3.1}$$

Ένα παράδειγμα αυτού του μηχανισμού κατασκευής λύσης δίνεται στο Σχήμα 3.15, για το πρόβλημα του περιοδεύοντος πωλητή αποτελούμενο από τέσσερις πόλεις μοντελοποιημένο ως γράφος με τέσσερις κόμβους. Η κατασκευή της λύσης ξεκινά με την τυχαία επιλογή ενός κόμβου αρχής για το μυρμήγκι, στην περίπτωση αυτή επιλέχθηκε ο κόμβος 1. Τα γραφήματα (α) και (β) δείχνουν τις επιλογές για το πρώτο και το δεύτερο κατασκευαστικό βήμα αντίστοιχα. Ο τρέχων κόμβος σε κάθε περίπτωση απεικονίζεται με μπλέ χρώμα, οι μη επιλεγμένοι κόμβοι με γαλάζιο χρώμα και οι ήδη επιλεγμένοι κόμβοι με πράσινο χρώμα. Οι πιθανές επιλογές του μυρμηγκιού απεικονίζονται με διακεκομμένες γραμμές. Μετά το δεύτερο βήμαμ όπου χάριν παραδείγματος έγινε η υπόθεση ότι το μυρμήγκι επέλεξε τον κόμβο 4, το μυρμήγκι μπορεί να κινηθεί μόνο προς τον κόμβο 3 και έπειτα πίσω στον κομβο 1 για να ολοκληρώσει τη διαδρομή.




P($e_{1,j}$)=$\tau_{1,j}$/($\tau_{1,2}$+$\tau_{1,3}$+$\tau_{2,4}$)

(α) Πρώτο βήμα της κατασκευής λύσης.

P($e_{2,j}$)=$\tau_{2,j}$/($\tau_{2,3}$+$\tau_{2,4}$)

(β) Δεύτερο βήμα της κατασκευής λύσης.

(γ) Η πλήρης λύση, μετά το τελευταίο κατασκευαστικό βήμα.

**Σχήμα 3.15 Αναπαράσταση της διαδικασίας κατασκευής λύσης**

Όταν όλα τα μυρμήγκια της αποικίας έχουν ολοκληρώσει την κατασκευή της λύσης τους πραγματοποιείται η εξάτμιση της φερομόνης ως εξής:

(3.2) $$\tau_{i,j} \leftarrow (1-\rho)\cdot\tau_{i,j}\,, \forall\, \tau_{i,j} \in \mathcal{T},$$

όπου $\mathcal{T}$ είναι το σύνολο όλων των τιμών της φερομόνης. Έπειτα τα μυρμήγκια πραγματοποιούν το ταξίδι επιστροφής. Σε αυτό, ένα μυρμήγκι, έχοντας κατασκευάσει μια λύση $s$, πραγματοποιεί για κάθε $e_{i,j} \in s$ την ακόλουθη εναπόθεση φερομόνης:

(3.3) $$\tau_{i,j} \leftarrow \tau_{i,j} + \frac{Q}{f(s)}\,,$$

όπου $Q$ μια θετική σταθερά και $f(s)$ είναι η τιμή της αντικειμενικής συνάρτησης της λύσης $s$. Η διαδικασία επαναλαμβάνεται, εφαρμόζοντας $n_\alpha$ μυρμήγκια ανά επανάληψη/γενιά, εώς ότου ένα κριτήριο τερματισμού (όπως παραδείγματος χάριν ένα χρονικό όριο) να ικανοποιηθεί.

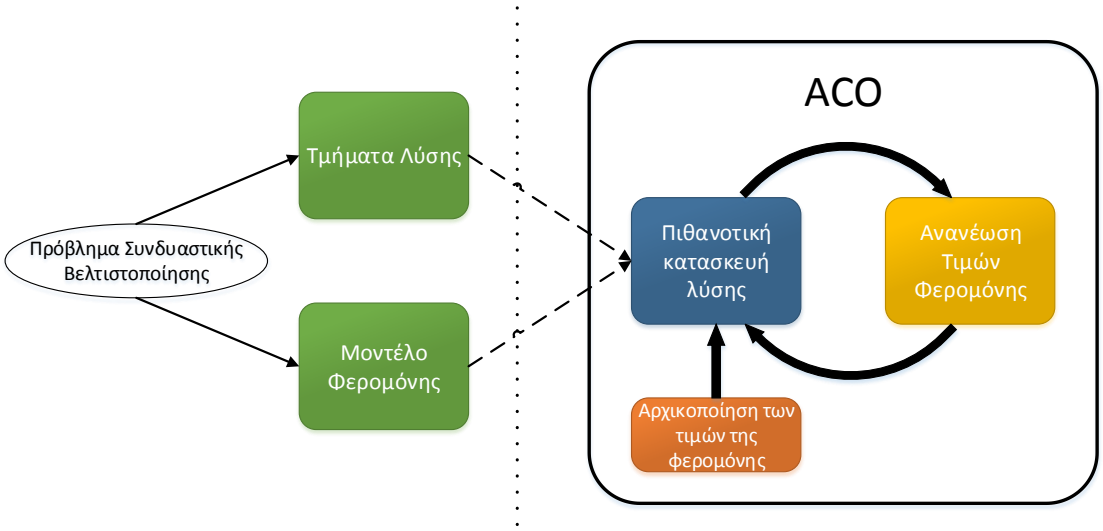
Αν και ο αλγόριθμος AS αποδεικνύει οτι η συμπεριφορά αναζήτησης τροφής των μυρμηγκιών μπορεί να μεταφερθεί σε έναν αλγόριθμο συνδυαστικής βελτιστοποίησης, βρέθηκε οτι ήταν γενικά κατώτερος των σύγχρονών του αλγορίθμων. Ως εκ τούτου, με τα χρόνια αναπτύχθηκαν αρκετές επεκτάσεις και βελτιώσεις του αρχικού αυτού αλγορίθμου βελτιστοποίησης με αποικία μυρμηγκιών.

### *3.6.2 Ο Σύγχρονος Μετα-ευρετικός ACO Αλγόριθμος*

Ο μετα-ευρετικός ACO, όπως τον γνωρίζουμε σήμερα, επισημοποιήθηκε πρώτη φορά το 1999 από τον M.Dorigo και τους συνεργάτες του (Dorigo and Di Caro, 1999). Ο βασικός τρόπος λειτουργίας ενός ACO αλγορίθμου φαίνεται στο Σχήμα 3.16. Με δεδομένο ένα πρόβλημα συνδυαστικής βελτιστοποίησης (CO), πρέπει κανείς πρώτα να παράγει ένα πεπερασμένο σύνολο $\mathcal{C}$ τμημάτων λύσεων, τα οποία χρησιμοποιούνται για την κατασκευή λύσεων για το CO πρόβλημα. Δεύτερον, πρέπει να ορίσει ένα σύνολο από τιμές φερομόνης $\mathcal{T}$. Το σύνολο αυτό τιμών καλείται ως το *μοντέλο φερομόνης* και είναι ουσιαστικά ένα παραμετροποιημένο πιθανοτικό μοντέλο. Το μοντέλο φερομόνης είναι ένα από τα κεντρικά μέρη του μετα-ευρετικού ACO. Οι τιμές φερομόνης $\tau_i \in \mathcal{T}$ αντιστοιχούν στα τμήματα λύσης. Το μοντέλο φερομόνης χρησιμοποιείται για την πιθανοτική κατασκευή λύσεων για το πρόβλημα υπό μελέτη, συναρμολογώντας τες από το σύνολο των τμημάτων λύσης. Γενικά, η προσέγγιση αυτή προσπαθεί να λύσει ένα πρόβλημα βελτιστοποίησης με επανάληψη των ακόλουθων δύο βημάτων:

- Υποψήφιες λύσεις κατασκευάζονται χρησιμοποιώντας ένα μοντέλο φερομόνης, το οποίο είναι μια παραμετροποιημένη πιθανοτική κατανομή στο χώρο επίλυσης.

- Οι υποψήφιες λύσεις χρησιμοποιούνται για να τροποποιηθούν οι τιμές της φερομόνης με τρόπο τέτοιο ώστε τα μελλοντικά δείγματα να συγκλίνουν σε υψηλής ποιότητας λύσεις.




**Σχήμα 3.16 Η λειτουργία του μετα-ευρετικού ACO αλγορίθμου**

Η ανανέωση της φερομόνης έχει στόχο να συγκεντρώσει την αναζήτηση στις περιοχές του χώρου αναζήτησης λύσεων που περιέχουν υψηλής ποιότητας λύσεις. Συγκεκριμένα, το γεγονός οτι η ενίσχυση των τμημάτων επίλυσης εξαρτάται από την ποιότητα της λύσης είναι ένα σημαντικό συστατικό των ACO αλγορίθμων. Γίνεται η έμμεση υπόθεση οτι οι καλές λύσεις αποτελούνται από καλά τμήματα λύσεων. Ο αλγόριθμος μαθαίνει έτσι πια τμηματα συμβάλλουν σε καλές λύσεις και αυτό τον βοηθάει να συναρμολογήσει με αυτά καλύτερες λύσεις. Ακολουθεί η παρουσίαση της κλασικής μορφής του ACO αλγορίθμου:

**Αλγόριθμος 3.6 Ant Colony Optimization (ACO)**

---

$Set\ PopulationSize = POP;$

$Set\ PheromoneUpdateRule;\quad SetDaemonActions;$

$Generate\ initial\ population;$

$\boldsymbol{While}\ termination\ conditions\ not\ met\ \boldsymbol{do}$

$\quad \boldsymbol{ScheduleActivites}$

$\quad\quad AntBasedSolutionConstruction;$

$\quad\quad PheromoneUpdate;$

$\quad\quad DaemonActions;$

$\quad \boldsymbol{End\ ScheduleActivities}$

$\boldsymbol{End\ While}$

---

Ο ACO είναι ένας επαναληπτικός αλγόριθμος του οποίου ο χρόνος εκτέλεσης εξαρτάται από την κεντρική δομή επανάληψης (*While - End While)*. Σε κάθε επανάληψη πρέπει να προγραμματιστούν τα τρία τμήματα του αλγορίθμου Κατασκευή Λύσης Βασισμένη Στα Μυρμήγκια (Ant Based Solution Construction), Ανανέωση Φερομόνης (Pheromone Update) και Ειδικές Ενέργειες (Daemon Actions) (συγκεντωμένα στην εσωτερική δομή επανάληψης Schedule Activities). Η δομή Schedule Activities δεν καθορίζει το πώς αυτές οι τρεις διαδικασίες προγραμματίζονται ή συγχρονίζονται. Αυτό είναι ευθύνη του σχεδιαστή του αλγορίθμου. Ακολουθεί αναλυτική παρουσίαση των τριών τμημάτων του αλγορίθμου.

**Αλγόριθμος 3.7 Διαδικασία AntBasedSolutionConstruction**

---

$s = \{\emptyset\}$;

$Determine\ \mathcal{E}(s)$;

***While*** $\mathcal{N}(s) \neq 0$ ***do***

  $c \leftarrow ChooseFrom(\mathcal{E}(s))$

  $s \leftarrow extend\ s\ by\ appending\ solution\ component\ c$;

  $Determine\ \mathcal{E}(s)$;

***End While***

---

Κατασκευή Λύσης Βασισμένη Στα Μυρμήγκια (Ant Based Solution Construction): Τα τεχνητά μυρμήγκια μπορούν να θεωρηθούν πιθανοτικές κατασκευαστικές ευρετικές διαδικασίες, οι οποίες συναρμολογούν λύσεις σαν ακολουθίες τμημάτων λύσεων. Το πεπερασμένο σύνολο των τμημάτων λύσης $C = \{c_1, \dots, c_n\}$ αντλείται από το πρόβλημα διακριτής βελτιστοποίησης που μελετάται. Για παράδειγμα, στην περίπτωση εφαρμογής του AS στο TSP κάθε ακμή του TSP γράφου θεωρείται ως τμήμα λύσης. Κάθε κατασκευή λύσης ξεκινά με την άδεια ακολουθία $s = \{\}$. Έπειτα η τρέχουσα ακολουθία $s$ σε κάθε κατασκευαστικό βήμα επεκτείνεται με την πρόσθεση ενός εφικτού τμήματος λύσης από το σύνολο $\mathcal{E}(s) \subseteq C\backslash s$. Ο ορισμός του $\mathcal{E}(s)$ εξαρτάται από τον μηχανισμό κατασκευής λύσης. Στο παράδειγμα της εφαρμογής του AS στο TSP ο μηχανισμός κατασκευής λύσης περιόριζε το σύνολο των προς διάνυση ακμών σε εκείνες που συνέδεαν τον τρέχοντα κόμβο των μυρμηγκιών με κόμβους που δεν είχαν επισκευθεί. Η επιλογή του τμήματος λύσης από το $\mathcal{E}(s)$ (η συνάρτηση ChooseFrom($\mathcal{E}(s)$) στον Αλγόριθμο 6.2) σε κάθε βήμα γίνεται πιθανοτικά, με εκτίμηση του μοντέλου φερομόνης. Οι

πιθανότητες (που ονομάζονται και *πιθανότητες μετάβασης*) ορίζονται ως εξής:

(3.4) $$\boldsymbol{p}(c_i|s) = \frac{[\tau_i]^{\alpha}\cdot[c(\eta_i)]^{\beta}}{\sum_{c_j\in\mathcal{E}(s)}[\tau_j]^{\alpha}\cdot[c(\eta_j)]^{\beta}}\ ,\ \forall c_i \in \mathcal{E}(s),$$

Όπου $c$ είναι μια προαιρετική συνάρτηση βάρους, η οποία, εξαρτώμενη κάποιες φορές από την τρέχουσα ακολουθία, εκχωρεί σε κάθε κατασκευαστικό βήμα μια ευρετική τιμή $c(\eta_j)$ σε κάθε εφικτό τμήμα λύσης $\eta_j \in \mathcal{E}(s)$. Οι τιμές που δίνονται από τη συνάρτηση βάρους καλούνται συνήθως *ευρετική πληροφορία*. Επιπλέον, οι εκθέτες $\alpha$ και $\beta$ είναι θετικές παράμετροι των οποίων οι τιμές καθορίζουν τη σχέση μεταξύ της πληροφορίας της φερομόνης και της ευρετικής πληροφορίας. Στο παράδειγμα του TSP που προηγήθηκε, επιλέχθηκε να μη χρησιμοποιηθεί κάποια συνάρτηση βάρους και η παράμετρος $\alpha$ τέθηκε 1.

Ανανέωση Φερομόνης (Pheromone Update): Οι διαφορετικές παραλλαγές ACO αλγορίθμων διαφέρουν κυρίως στην ανανέωση που εφαρμόζουν στις τιμές της φερομόνης. Η ανανέωση φερομόνης αποτελείται από δύο μέρη. Αρχικά, εφαρμόζεται η *εξάτμιση φερομόνης (pheromone evaporation)* που ομοιόμορφα μειώνει όλες τις τιμές φερομόνης. Η εξάτμιση φερομόνης είναι αναγκαία για να αποφευχθεί πολύ γρήγορη σύγκλιση του αλγορίθμου προς κάποια υπο-βέλτιστη περιοχή λύσεων. Έπειτα, μια ή περισσότερες λύσεις από την τρέχουσα και/ή από προηγούμενες επαναλήψεις χρησιμοποιούνται για την αύξηση των τιμών των παραμέτρων του ίχνους φερομόνης των τμημάτων λύσης που αποτελούν μέρη αυτών των λύσεων.

(3.5) $$\tau_i \leftarrow (1-\rho)\cdot\tau_i + \rho\cdot\sum_{\{s\in S_{upd}|c_i\in s\}}\omega_s\cdot F(s)\ ,\ \text{για}\ i=1,\dots,n$$

Ορίζεται ως $S_{upd}$ το σύνολο των λύσεων που χρησιμοποιούνται για την ανανέωση του μοντέλου φερομόνης. Επιπλέον, $\rho \in (0,1]$ είναι μια παράμετρος που ονομάζεται *ρυθμός εξάτμισης(evaporation rate)* και $F: S \to \mathbb{R}^+$ είναι η λεγόμενη *συνάρτηση ποιότητας/καταλληλότητας (quality/fitness function)*, τέτοια ώστε $f(s) < f(s') \Longrightarrow F(s) \geq F(s'),\ \forall s \neq s' \in S$. Αυτό πρακτικά σημαίνει ότι αν η τιμή της αντικειμενικής συνάρτησης μιας λύσης $s$ είναι καλύτερη από την τιμή της αντικειμενικής συνάρτησης μις λύσης $s'$ τότε η ποιότητα της λύσης $s$ θα είναι το λιγότερο τόσο υψηλή όσο η ποιότητα της λύσης $s'$. Επίσης, η εξίσωση 6.5 επιτρέπει μια επιπλέον ενίσχυση της συνάρτησης καταλληλότητας με τον ορισμό βάρους $\omega_s \in \mathbb{R}^+$ για κάθε λύση $s$.

Παραλλαγές αυτού του κανόνα ανανέωσης λαμβάνονται από διαφορετικούς προσδιορισμούς του $S_{upd}$ και από διαφορετικές ρυθμίσεις των βαρών. Σε πολλές περιπτώσεις το $S_{upd}$ συντίθεται από κάποιες από τις λύσεις που παράχθηκαν στην τρέχουσα επανάληψη, που καλούνται $S_{iter}$ και από την καλύτερη λύση που έχει βρεθεί από την έναρξη του αλγορίθμου, που καλείται $s_{bs}$. Η λύση $s_{bs}$ καλείται συχνά ως η «βέλτιστη-προς-το-παρόν» λύση. Ένα γνωστό παράδειγμα είναι ο κανόνας ανανέωσης *AS (AS-update rule),* ο οποίος είναι ο κανόνας ανανέωσης που εφαρμόσθηκε στον αλγόριθμο AS (Ant System). Ο AS-update rule, ο οποίος είναι ευρέως γνωστός λόγω του ότι ο αλγόριθμος AS ήταν ο πρώτος ACO αλγόριθμος που προτάθηκε στη βιβλιογραφία, λαμβάνεται από τη σχέση 3.5 θέτοντας $S_{upd} \leftarrow S_{iter}$ και $\omega_s = 1 \ \forall \ s \in S_{upd}$, δηλαδή χρησιμοποιεί όλες τις λύσεις που παράχθηκαν κατά την τρέχουσα επανάληψη για την ανανέωση της φερομόνης, θέτοντας παράλληλα το βάρος κάθε λύσης ως μονάδα. Ένα άλλο παράδειγμα ενός κανόνα ανανέωσης φερομόνης που χρηιμοποιείται πιο συχνά στην πράξη είναι ο κανόνας ανανέωσης *IB (IB update rule)*, όπου τα αρχικά IB σημαίνουν Iteration Best (Βέλτιστο της Επανάληψης). Ο IB-update rule ορίζεται ως: $S_{upd} \leftarrow \{s_{ib} = argmax\{F(s)|s \in S_{iter}\}\}$ με $\omega_{s_{ib}} = 1$, δηλαδή επιλέγεται μόνο η καλύτερη λύση που παράχθηκε από την τρέχουσα επανάληψη για την ανανέωση των τιμών της φερομόνης. Αυτή η λύση, που καλείται $s_{ib}$, έχει βάρος μονάδα. Ο IB-update rule εισάγει μια πολύ μεγαλύτερη κλίση προς τις καλές λύσεις που βρέθηκαν σε σχέση με τον AS-update rule, αυξάνοντας όμως τον κίνδυνο πρόωρης σύγκλισης του αλγορίθμου σε τοπικά βέλτιστα. Μια ακόμα ισχυρότερη κλίση εισάγεται με τον κανόνα ανανέωσης *BS (BS-update rule)*, όπου τα αρχικά BS σημαίνουν Best Solution ( Βέλτιστο του Αλγορίθμου) και αναφέρονται στη χρήση της βέλτιστης προς το παρόν λύσης $s_{bs}$. Στην περίπτωση αυτή το $S_{upd}$ τίθεται ως $\{s_{bs}\}$ και η λύση $s_{bs}$ έχει βάρος μονάδα. Στην πράξη οι ACO αλγόριθμοι που χρησιμοποιούν παραλλαγές των κανόνων IB-update rule και BS-update rule (Merkle et al., 2002) και επιπρόσθετα περιλαμβάνουν μηχανισμούς αποφυγής πρόωρης σύγκλισης πετυχαίνουν καλύτερα αποτελέσματα από αλγορίθμους που χρησιμοποιούν τον AS-update rule.

Ένας ακόμα τρόπος ανανέωσης φερομόνης δίνεται από τον τύπο 3.5 κρατώντας ίδιο το τμήμα εξάτμισης της φερομόνης και μετατρέποντας το τμήμα ανανέωσης ως εξής :

(3.6) $\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j} + \ \rho/2T^*, \ \tau_{i,j} \in \ s_{bs}$ ή $\ s_{ıb}, \ T^* = makespan \ of \ BS$ ή $IB$

Σύμφωνα με την σχέση 6.6 (Merkle et al., 2002) η ανανέωση της φερομόνης γίνεται σε όσα τμήματα λύσης *j* ανήκουν στην λύση $s_{bs}$ ή την $s_{ib}$, ανάλογα κάθε φορά με το ποιός από τους δύο κανόνες ανανέωσης χρησιμοποιείται, ενώ σε όλες τις

λύσεις όποιο τμήμα λύσης τους $j$ έχει θέση $i$ ίδια με εκείνη του αντίστοιχου τμήματος $j$ στην $s_{bs}$ ή την $s_{ib}$ και εκείνο θα δεχτεί ανανέωση φερομόνης, ενώ τα υπόλοιπα τμήματά του όχι.

Ειδικές Ενέργειες (Daemon Actions): Ειδικές Ενέργειες μπορούν να χρησιμοποιηθούν για την εφαρμογή κεντρικών ενεργειών, οι οποίες δεν μπορούν να εκτελεστούν ατομικά από τα μυρμήγκια. Παραδείγματα τέτοιων ενεργειών είναι η εφαρμογή μεθόδων τοπικής αναζήτησης στις κατασκευασμένες λύσεις ή η συλλογή σφαιρικών πληροφοριών που μπορούν να χρησιμοποιηθούν για να αποφασισθεί αν είναι χρήσιμο ή όχι να εναποτεθεί επιπρόσθετη φερομόνη για να κατευθύνει την διαδικασία αναζήτησης από μια μη-τοπική προοπτική (non-local perspective).

Ένα τέτοιο παράδειγμα είναι η αξιολόγηση της φερομόνης όχι μόνο άμεσα, μέσω της σχέσης 3.5, αλλά και έμμεσα, δηλαδη και με άμεση αξιολόγηση (direct evaluation) και με αθροιστική αξιολόγηση (summation evaluation). Άμεση αξιολόγηση γίνεται όταν τα μυρμήγκια κάθε επόμενης γενιάς χρησιμοποιούν απευθείας στην σχέση 3.4β την τιμή της φερομόνης $\tau_{i,j}$ για να εκτιμήσουν το πόσο επιθυμητή είναι η τοποθέτηση ενός τμήματος λύσης $j$ στην $i$ θέση κατά τον υπολογισμό μιας νέας λύσης. Η έμμεση αξιολόγηση είναι ένας διαφορετικός τρόπος εκτίμησης του πίνακα φερομονών. Αντί να χρησιμοποιηθεί μόνο η τοπική τιμή $\tau_{i,j}$ για την εξαγωγή της πιθανότητας τοποθέτησης ενός τμήματος λύσης $j$ στην $i$ θέση, χρησιμοποιείται μια πιο σφαιρική οπτική των τιμών των φερομονών από τα μυρμήγκια. Σε αυτή τη μορφή σφαιρικής αξιολόγησης της φερομόνης τα μυρμήγκια εκτιμούν τον πίνακα φερομονών χρησιμοποιώντας το άθροισμα $\sum_{k=1}^{i} \tau_{k,j}$ για τον υπολογισμό της πιθανότητας τοποθέτησης του τμήματος λύσης $j$ στη θέση $i$ (Merkle and Middendorf, 2000).

Με την χρήση μόνο της άμεσης αξιολόγησης της φερομόνης παρουσιάζεται το εξής πρόβλημα: Αν ένα τμήμα λύσης $j$ έχει για τη θέση $i$ υψηλή τιμή $\tau_{i,j}$ και δεν επιλεγεί από τα μυρμήγκια, τότε αν για τις επόμενες θέσεις έχει χαμηλές τιμές $\tau_{i+1,j}, \tau_{i+2,j}$, ... υπάρχει ο κίνδυνος να τοποθετηθεί πολύ πίσω από την θέση $i$. Αυτό είναι ένα σημαντικό πρόβλημα αν η αργοπορία κάθε τμήματος πρέπει να ελαχοστιποιηθεί ή αν οι σχέσεις προτεραιότητας πιθανά εμποδίζουν άλλα τμήματα λύσης να προγραμματιστούν. Αυτό με την αθροιστική αξιολόγηση συνήθως δεν θα συμβεί, καθώς είναι δύσκολο ένα τμήμα λύσης να μπει στο πρόγραμμα πολύ πίσω από θέσεις που έχουν υψηλές αντίστοιχες τιμές φερομόνης.

Το επόμενο τμήμα λύσης με βάση την άμεση αξιολόγηση επιλέγεται με μια σχέση αντίστοιχη της 3.4, όπου ορίζονται τμήμα λύσης και θέση ως $j$ και $i$ αντίστοιχα και το σύνολο των επιλέξιμων τμημάτων είναι το $\mathcal{E}$:

(3.7) $$p(i,j) = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{h \in \mathcal{E}} [\tau_{ih}]^{\alpha} \cdot [\eta_{ih}]^{\beta}} \ , \ \forall h \in \mathcal{E}$$

Και με βάση την αθροιστική αξιολόγηση:

(3.8) $$p'(i,j) = \frac{\left(\sum_{k=1}^{i} [\gamma^{i-k} \cdot \tau_{kj}]\right)^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{h \in \mathcal{E}} \left(\sum_{k=1}^{i} [\gamma^{i-k} \cdot \tau_{kh}]\right)^{\alpha} \cdot [\eta_{ih}]^{\beta}} \ , \ \forall h \in \mathcal{E}$$

Όπου ορίζεται η παράμετρος $\gamma > 0$ η οποία καθορίζει τη σχετική επιρροή των τιμών της φερομόνης που αντιστοιχούν στις νωρίτερες αποφάσεις των μυρμηγκιών, δηλαδή στις προηγούμενες θέσεις της υπό κατασκευή επίλυσης. Η τιμή $\gamma = 1$ οδηγεί σε αθροιστική αξιολόγηση χωρίς βάρος, δηλαδή σε κάθε $\tau_{kj}, k \leq i$ δίνεται η ίδια δύναμη επιρροής. Η τιμή $\gamma < 1$ ή $\gamma > 1$ δίνει στις τιμές φερομόνης που αντιστοιχούν σε νωρίτερες αποφάσεις μικρότερη ή μεγαλύτερη σημασία αντίστοιχα (Merkle et al., 2002).

Από την άλλη άποψη και ειδικά στο RCPSP είναι σημαντικό κάποιες δραστηριότητες (τμήματα λύσης) να προγραμματιστούν όχι πολύ αργά, αλλά είναι επίσης σημαντικό να προγραμματιστούν στον ίδιο χρόνο ομάδες δραστηριοτήτων που έχουν απαιτήσεις σε πόρους που μπορούν να ταιριάξουν στους περιορισμούς σε πόρους του προβλήματος. Επομένως, για κάποιες δραστηριότητες υπάρχουν πιθανά αρκετές θέσεις στη λίστα των δραστηριοτήτων που είναι καλές, ενώ άλλες ανάμεσά τους πιθανά είναι χειρότερες. Αυτή η συμπεριφορά μοντελλοποιείται καλύτερα με την άμεση αξιολόγηση, παρά με την αθροιστική. Για να αξιοποιηθούν τα πλεονεκτήματα και των δύο τρόπων αξιολόγησης δεν πρέπει να χρησιμοποιηθεί μόνο η άμεση ή μόνο η αθροιστική αλλά ένας συνδυασμός τους όπως αυτός που ακολουθεί. Μια παράμετρος $c, 0 \leq c \leq 1$ καθορίζει τη σχετική επιρροή της άμεσης και της αθροιστικής αξιολόγησης. Αυτό επιτυγχάνεται χρησιμοποιώντας τη σχέση 3.7, αλλά οι τιμές $\tau_{ij}$ αντικαθιστώνται από τις παρακάτω «νέες» τιμές $\tau'_{ij}$:

(3.9) $$\tau'_{ij} = c \cdot x_i \cdot \tau_{ij} + (1-c) \cdot y_i \cdot \sum_{k=1}^{i} \gamma^{i-k} \cdot \tau_{kj}$$

Όπου $x_i = \sum_{h \in \mathcal{E}} \sum_{k=1}^{i} \gamma^{i-k} \cdot \tau_{kh}$ και $y_i = \sum_{h \in \mathcal{E}} \tau_{ih}$ είναι συντελεστές που χρησιμοποιούνται για να ρυθμίσουν τη σχετική επιρροή της άμεσης και της αθροιστικής αξιολόγησης. Για $c = 1$ γίνεται καθαρά άμεση αξιολόγηση και για $c = 0$ καθαρά αθροιστική αξιολόγηση.

### *3.6.3 Επιτυχείς παραλλαγές ACO αλγορίθμων*

Παρόλο που ο πρωτότυπος AS αλγόριθμος πέτυχε ενθαρρυντικά αποτελέσματα για το πρόβλημα του περιοδεύοντος πωλητή, βρέθηκε οτι ήταν κατώτερος από τους σύγχρονους του αλγόριθμους για το TSP καθώς και για άλλα προβλήματα συνδυαστικής βελτιστοποίησης (CO). Έτσι, αρκετές επεκτάσεις και

βελτιώσεις του πρωτότυπου AS αλγορίθμου προτάθηκαν με το πέρασμα των χρόνων. Οι βασικές παραλαγές του περιγράφονται στον Πίνακα 3.3

**Πίνακας 3.3 Παραλλαγές ACO Αλγορίθμων**

| Παραλλαγές ACO | Αναφορές |
|---|---|
| Elitist AS (EAS) | (Dorigo, 1992, Dorigo et al., 1996) |
| Rank-based AS (RAS) | (Bullnheimer et al., 1997) |
| MAX-MIN Ant System (MMAS) | (Stützle and Hoos, 2000) |
| Ant Colony System (ACS) | (Dorigo and Gambardella, 1997) |
| Hyper-Cube Framework (HCF) | (Blum and Dorigo, 2004) |

Μια πρώτη βελτίωση επί του AS αλγορίθμου ήταν ο Elitist AS αλγόριθμος (EAS), ο οποίος λαμβάνεται τοποθετώντας στη σχέση 3.5 του κανόνα ανανέωσης της φερομόνης $S_{upd} \leftarrow S_{iter} \cup \{s_{bs}\}$, δηλαδή χρημοποιώντας όλες τις λύσης που παράχθηκαν κατά την τρέχουσα επανάληψη και επιπρόσθετα την βέλτιστη προς το παρόν λύση για την ανανέωση των τιμών της φερομόνης. Τα βάρη θα είναι $\omega_s = 1 \; \forall \; s \in S_{iter}$ εκτός από το βάρος της βέλτιστης προς το παρόν λύσης που μπορεί να είναι υψηλότερο: $\omega_{s_{bs}} \geq 1$. Ο βασικός στόχος ήταν να αυξηθεί η εκμετάλλευση της βέλτιστης προς το παρόν λύσης εισάγοντας ισχυρή κλίση προς τα τμήματα λύσης που αυτή περιέχει.

Μια άλλη βελτίωση επί του AS είναι ο Rank-based AS αλγόριθμος (RAS). Η ανανέωση φερομόνης στον RAS λαμβάνεται από τη σχέση 3.5 γεμίζοντας το σύνολο $S_{upd}$ με τις καλύτερες $m-1$ (όπου $m-1 \leq n_\alpha$) λύσεις από το σύνολο $S_{iter}$ και επιπρόσθετα εισάγοντας και την βέλτιστη προς το παρόν λύση $s_{bs}$ στο $S_{upd}$. Τα βάρη των λύσεων τίθενται ως $\omega_s = m - r_s \; \forall \; s \in S_{upd} \backslash \{s_{bs}\}$, όπου $r_s$ είναι η τάξη της λύσης $s$. Τέλος, το βάρος $\omega_{s_{bs}}$ της λυσης $s_{bs}$ τίθεται ως $m$. Αυτό σημαίνει ότι σε κάθε επανάληψη η βέλτιστη προς το παρόν λύση έχει τη μεγαλύτερη επιρροή στην ανανέωση της φερομόνης ενώ ένα σύνολο από επιλεγμένες βέλτιστες λύσεις που κατασκευάζεται στην παρούσα επανάληψη επηρρεάζει την ανανέωση ανάλογα με την τάξη κάθε μιας από αυτές.

Μια από τις πιο επιτυχημένες παραλλαγές ACO σήμερα είναι ο MAX-MIN Ant System αλγόριθμος (MMAS). Με βάση κάποιο μέτρο σύγκλισης, σε κάθε επανάληψη είτε ο IB-update rule είναι ο BS-update rule (όπως περιγράφηκαν προηγουμένως) χρησιμοποιούνται για την ανανέωση των τιμών της φερομόνης. Στην αρχή του αλγορίθμου ο IB-update rule χρησιμοποιείται πιο συχνά, ενώ κατά την διάρκεια εκτέλεσης του αλγορίθμου η συχνότητα με την οποία χρησιμοποιείται ο BS-update

rule αυξάνεται. Οι MMAS αλγόριθμοι χρησιμοποιούν ένα αυστηρό κατώτατο όριο $\tau_{min} > 0$ για τις τιμές φερομόνης. Επιπρόσθετα σε αυτό το κάτω όριο, οι MMAS αλγόριθμοι χρησιμοποιούν το λόγο $F(s_{bs})/\rho$ σαν ένα άνω όριο για τις τιμές φερομόνης. Η τιμή αυτού του ορίου ανανεώνεται κάθε φορά που μια νέα βέλτιστη προς το παρόν λύση βρίσκεται από τον αλγόριθμο.

Ο αλγόριθμος Ant Colony System (ACS) διαφέρει από τον πρωτότυπο AS αλγόριθμο σε περισσότερες πτυχές και όχι μόνο στην ανανέωση της φερομόνης. Πρώτον, αντί να επιλέγει σε κάθε βήμα κατά τη διάρκεια κατασκευής μιας λύσης το επόμενο τμήμα λύσης σύμφωνα με την εξίσωση 3.4, το τεχνητό μυρμήγκι επιλέγει με πιθανότητα $q_0$ το τμήμα λύσης που μεγιστοποιεί τον όρο $[\tau_i]^a \cdot [\eta(c_i)]^\beta$ ή πραγματοποιεί με πιθανότητα $1 - q_0$ ένα πιθανοτικό κατασκευαστικό βήμα σύμφωνα με την εξίσωση 3.4. Αυτός ο τύπος κατασκευής λύσης καλείται *ψευδο-τυχαία αναλογικός (pseudo-random proportional)*. Δεύτερον, ο ACS χρησιμοποιεί τον BS-update rule με την επιπλέον ιδιαιτερότητα οτι η ανανέωση φερομόνης εφαρμόζεται μόνο στις τιμές των παραμέτρων ιχνών φερομόνης που ανήκουν στα τμήματα λύσης που περιλαμβάνονται στην $s_{bs}$. Τρίτον, μετά από κάθε κατασκευαστικό βήμα λύσης η ακόλουθη επιπλέον ανανέωση φερομόνης εφαρμόζεται στην τιμή φερομόνης $\tau_i$ της οποίας το αντίστοιχο τμήμα λύσεως $c_i$ προστέθηκε στην υπό κατασκευή λύση: $\tau_i \leftarrow (1 - \xi) \cdot \tau_i + \xi \cdot \tau_0$, όπου $\tau_0$ είναι μια μικρή θετική σταθερά, τέτοια ώστε $F_{min} \geq \tau_0 \geq c$ , $F_{min} \leftarrow min\{F(s) | \, s \in S\}$, και $c$ είναι η αρχική τιμή των τιμών φερομόνης. Στην πράξη, το αποτέλεσμα αυτής της τοπικής ανανέωσης φερομόνης είναι να μειώσει τις τιμές της φερομόνης στα τμήματα λύσεων που έχουν ήδη επισκεφθεί τα μυρμήγκια, κάνοντας με αυτό τον τρόπο τα τμήματα αυτά λιγότερο επιθυμητά από τα μυρμήγκια που θα ακολουθήσουν. Ο μηχανισμός αυτός αυξάνει την εξερεύνηση του χώρου επίλυσης εντός κάθε επανάληψης.

Μια από τις πιο σύγχρονες εξελίξεις είναι το Hyper-Cube Framework (HCF) για τους ACO αλγόριθμους. Το HCF, δεν είναι τόσο μια παραλλαγή των ACO αλγορίθμων, αλλά περισσότερο ένα πλαίσιο εκτέλεσης ACO αλγορίθμων, το οποίο χαρακτηρίζεται από μια ανανέωση φερομόνης που λαμβάνεται από τη σχέση 3.5, ορίζοντας το βάρος κάθε επίλυσης στο $S_{upd}$ ως $\left(\sum_{\{s \in S_{upd}\}} F(s)\right)^{-1}$. Υπενθυμίζεται εδώ οτι στην σχέση 3.5 οι λύσεις συνοδεύονται από βάρη. Το σύνολο $S_{upd}$ μπορεί να συντεθεί με κάθε πιθανό τρόπο, κάτι το οποίο συνεπάγεται οτι παραλλαγές ACO όπως οι AS, ACS και MMAS μπορούν να εκτελεστούν εντός του HCF. Το HCF έχει αρκετά προτερήματα. Από πρακτικής όψεως, το νέο πλαίσιο αυτόματα χειρίζεται την κλιμάκωση των τιμών της αντικειμενικής συνάρτησης και περιορίζει τις τιμές της φερομόνης στο διάστημα $[0,1]$. Από θεωριτικής όψεως, η μέση ποιότητα των λύσεων

που παράγονται αυξάνεται συνεχόμενα σε προσδοκία με την πάροδο του χρόνου. Το όνομα Hyper-Cube Framework προέρχεται από το γεγονός ότι με τον συγκεκριμένο ορισμό των βαρών στη σχέση 6.5 η ανανέωση της φερομόνης μπορεί να παρασταθεί σαν μια μετατόπιση σε έναν υπερκύβο.

### *3.6.4 Εφαρμογές των ACO αλγορίθμων στα προβλήματα διακριτής βελτιστοποίησης*

Όπως προαναφέρθηκε, ο αλγόριθμος ACO παρουσιάστηκε στο πλαίσιο απόδειξης της έννοιάς του για εφαρμογή στο TSP. Έκτοτε οι ACO αλγόριθμοι έχουν εφαρμοσθεί σε πολλά προβλήματα διακριτής βελτιστοποίησης (CO). Πρώτα επιχειρήθηκε να αντιμετωπιστούν κλασσικά προβλήματα πέραν του TSP, όπως προβλήματα καταμερισμού (assignment problems), προβλήματα πργραμματισμού (scheduling problems), χρωματισμού γράφων (graph coloring problems) ή πρβλήματα δρομολόγησης οχημάτων (vehicle routing problems). Πιο πρόσφατες εφαρμογές περιλαμβάνουν, για παράδειγμα, προβλήματα τοποθέτησης κελιών (cell placement problems), το σχεδιασμό επικοινωνιακών δικτύων (communication networks), ή προβλήματα βιοπληροφορικής (bioinformatics problems). Τα τελευταία χρόνια κάποιοι ερευνητές έχουν επικεντρωθεί στην εφαρμογή ACO αλγορίθμων σε προβλήματα πολλαπλών στόχων (multi objective problems) και σε δυναμικά (dynamic) ή στοχαστικά προβλήματα (stochastic problems).

Οι ACO αλγόριθμοι βρίσκονται σήμερα μεταξύ των κορυφαίων (state-of-the-art) μεθόδων για την επίλυση προβλημάτων όπως το πρόβλημα διαδοχικών παραγγελιών (sequential ordering problems), το πρόβλημα προγραμματισμού με περιορισμένους πόρους (RCPSP), το πρόβλημα προγραμματισμού ανοικτού καταστήματος (open-shop scheduling problem) και το διδιάστατο και τριδιάστατο πρόβλημα δίπλωσης της υδροφοβικής πολικής πρωτεΐνης (2D & 3D hydrophobic polar protein folding problem). Συνδυάζονται με ευρετικές μεθόδους όπως ο SSGS ή ο PSGS και με χρήση κατάλληλων κανόνων προτεραιότητας, συναρτήσεων φερομόνης και κανόνων επιλογής εργασιών και αξιολόγησης προγραμμάτων έχουν δώσει συστήματα με πολύ καλά αποτελέσματα στην αντιμετώπιση προβλημάτων όπως όσα προαναφέρθηκαν και για το λόγο αυτό εφαρμόζονται όλο και συχνότερα στο RCPSP και τις επεκτάσεις του (Dorigo and Gambardella, 1997, Dorigo and Di Caro, 1999, Merkle et al., 2002, Luo et al., 2003, Dorigo and Blum, 2005, Chiang et al., 2008).

# 4. Ορισμός του υπό μελέτη προβλήματος

## 4.1 Περιγραφή του ύπο μελέτη προβλήματος

Ο προγραμματισμός έργων είναι ένα πολύπλοκο πρόβλημα, το οποίο αντιμετωπίζουν όλοι οι διαχειριστές έργων (project manager) στην αρχή κάθε έργου. Από το σωστό σχεδιασμό του προγράμματος κρίνεται σε μεγάλο βαθμό η επιτυχία της εκτέλεσης και ολοκλήρωσης ενός έργου. Ένα κακώς σχεδιασμένο πρόγραμμα μπορεί να θέσει σε κίνδυνο ολόκληρο το έργο.

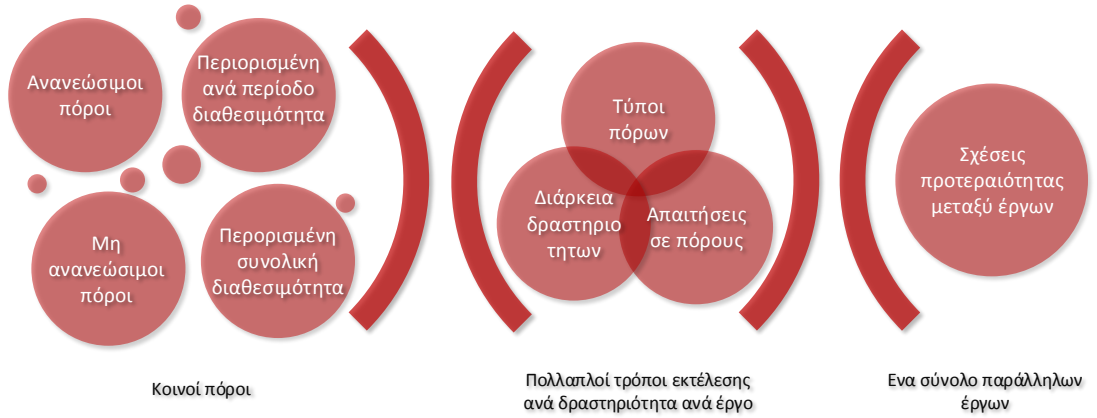
Το πρόβλημα αυτό, όταν εισάγονται οι περιορισμοί στους πόρους, στις αλληλοεξαρτήσεις μεταξύ των δραστηριοτήτων του, οι στόχοι που πρέπει να επιτευχθούν και αυξάνεται ο αριθμός των έργων που πρέπει να προγραμματιστούν, μετατρέπεται σε ένα ιδιαίτερα απαιτητικό, συνδυαστικού προγραμματισμού πρόβλημα. Για την επίλυση τέτοιων προβλημάτων απαιτούνται υπολογιστικές διαδικασίες, ικανές να παρέχουν λύσεις εφικτές, όσο το δυνατόν καλύτερες, με το μικρότερο δυνατό κόστος χρόνου, όπως οι μέθοδοι που περιγράφηκαν στην προηγούμενη παράγραφο. Οι μέθοδοι αυτές όμως, για να δώσουν αποτελέσματα που να μπορούν να ανταποκριθούν στα πρακτικά προβλήματα απαιτούν την κατάλληλη μοντελοποίηση του προβλήματος με τρόπο που να μπορεί από τη μία να το χειριστεί ένας αλγόριθμος, αλλά από την άλλη να πλησιάζει ικανοποιητικά την πολυπλοκότητα των πραγματικών, πρακτικών προβλημάτων που αντιμετωπίζονται σε ένα έργο.

Ένα πρόγραμμα πρέπει:

i. Να οδηγεί στο προσδωκόμενο αποτέλεσμα, εκφρασμένο σε όρους ποιότητας, κόστους και χρονικού πλαισίου.
ii. Να χρησιμοποιεί βέλτιστα τους διαθέσιμους πόρους και να ελαχιστοποιεί το κόστος του έργου χωρίς να παρακάμπτεται το (i).
iii. Να είναι ανθεκτικό σε αλλαγές και να είναι εύρωστο, δηλαδή μικρές αλλαγές στο πλάνο του έργου, διαταραχές στη διάρκεια εκτέλεσης ή στη διαθεσιμότητα των πόρων να έχουν περιορισμένη επίδραση στο συνολικό πρόγραμμα.

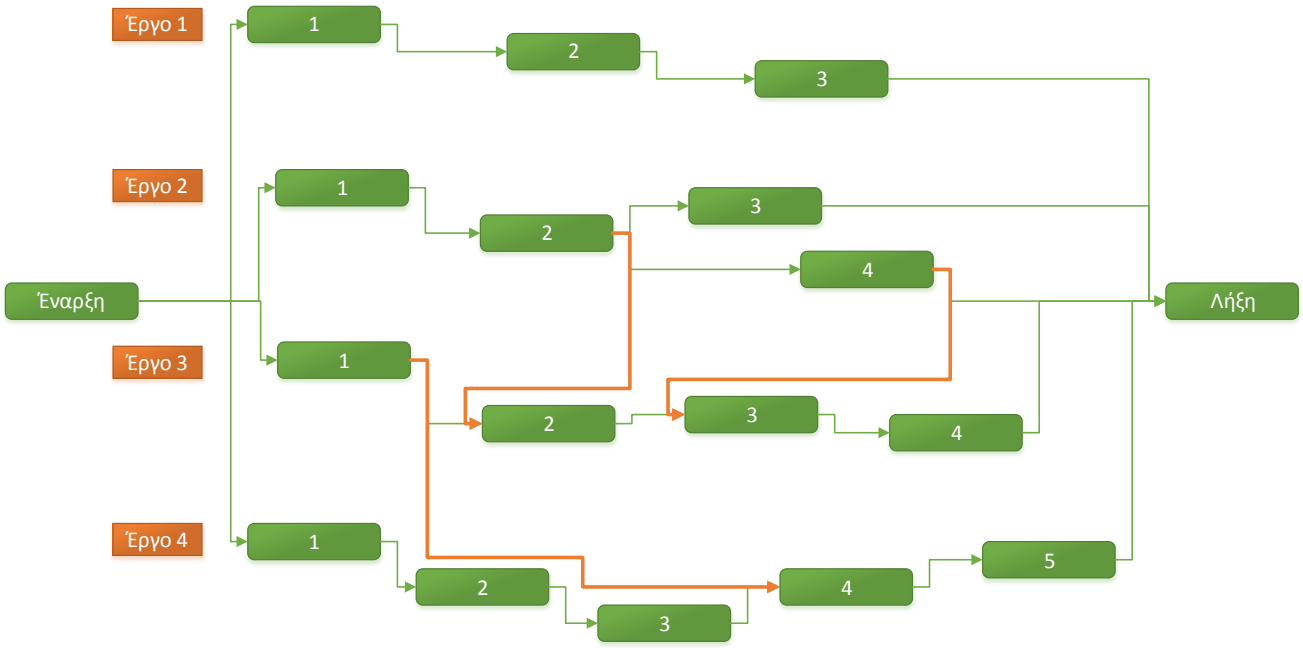
Ένα πρόβλημα προγραμματισμού που προσεγγίζει κάποιες καταστάσεις της πραγματικότητας είναι το πρόβλημα του προγραμματισμού πολλαπλών έργων, των οποίων οι δραστηριότητες έχουν πολλαπλούς τρόπους εκτέλεσης και οι πόροι είναι κοινοί για όλα τα έργα, ενώ υπάρχουν και συσχετίσεις μεταξύ δραστηριοτήτων διαφορετικών έργων (Multi-Project Multi-Mode RCPSP). Το πρόβλημα Multi-Project MRCPSP αποτελεί γενίκευση του RCPSP. Αποτελείται από αρκετά παράλληλα έργα

και ένα σύνολο κοινών αποθεμάτων σε πόρους, ανανεώσιμους και μη-ανανεώσιμους. Όλοι οι πόροι παρέχονται με περιορισμούς. Τα έργα μπορεί να είναι ανεξάρτητα μεταξύ τους ή και όχι. Κάθε έργο έχει ένα σύνολο δραστηριοτήτων που κάθε μία μπορεί να εκτελεστεί με έναν ή περισσότερους τρόπους (multi-mode). Κάθε τρόπος αντιστοιχεί σε διαφορετικό τύπο και απαιτούμενη ποσότητα πόρων και οδηγεί σε διαφορετική διάρκεια δραστηριότητας. Η δομή και η σύνθεση του προβλήματος απεικονίζεται στο Σχήμα 4.1.




**Σχήμα 4.1 Σύνθεση προβλήματος**

Το πρόβλημα αυτό είναι ορισμένο έτσι ώστε να είναι αφ' εαυτού κοντά στην πραγματικότητα, προσπαθώντας να ικανοποιήσει όσα περιγράφηκαν παραπάνω καταρχήν σαν έννοια αλλά και στη συνέχεια με τη μοντελοποίηση και την επίλυσή του. Στην πράξη ένας διαχειριστής έργου έρχεται συχνά αντιμέτωπος με πολλαπλά έργα τα οποία πρέπει να προγραμματίσει. Επίσης, συνήθως οι πόροι που είναι διαθέσιμοι προς αξιοποίηση είναι πολλών τύπων και κοινοί για όλα τα έργα, ενώ οι δραστηριότητες ενός έργου μπορεί να εξαρτώνται από δραστηριότητες ενός άλλου έργου, δηλαδή τα έργα μπορεί να έχουν όχι μόνο εσωτερικές αλλά και εξωτερικές εξαρτήσεις, όπως φαίνεται στο Σχήμα 4.2 που ακολουθεί.




**Σχήμα 4.2 Πολλαπλά έργα και συσχετίσεις μεταξύ τους**

Όλα τα παραπάνω οδηγούν πρακτικά στον πρόβλημα του ταυτόχρονου προγραμματισμού όλων αυτών των παράλληλων έργων. Ακολουθεί διακριτή ανάλυση των στόχων, των πόρων και του τρόπου προγραμματισμού στο πρόβλημα που αντιμετωπίζεται.

### *4.1.1 Στόχοι*

Το πρόγραμμα έργου πρέπει να οδηγεί σε ένα προϊόν με την ορισμένη εξ' αρχής ποιότητα, εντός χρόνου και προϋπολογισμού και με ελαχιστοποίηση του κόστους, ισορροπημένη χρήση πόρων και ανθεκτικότητα σε μεταβολές αστάθμητων παραγόντων. Οι στόχοι που επιδιώκονται κατά τον προγραμματισμό του έργου ορίζονται μεταφράζοντας τα απαιτούμενα χαρακτηριστικά ενός προγράμματος στους πραγματικούς στόχους προς βελτιστοποίηση κατά τον προγραμματισμό του έργου.

Σε όλα τα έργα υπάρχει μια ημερομηνία ολοκήρωσης που τίθεται είτε από τον πελάτη είτε εσωτερικά στο έργο. Επομένως η διάρκεια ενός έργου είναι ένα βασικό μέτρο απόδοσης και είναι ο στόχος που πιο συχνά χρησιμοποιείται στη βιβλιογραφία και στην πράξη (Boctor, 1990, Icmeli and Erenguc, 1996b, Hartmann and Drexl, 1998). Συχνά οι προθεσμίες για ένα έργο ή για ενδιάμεσα διαστήματά του συνδέονται με ποινές με την μορφή πρόσθετου κόστους ή με επιβραβεύσεις των νωρίτερων

εκτελέσεων, συνδέοντας έτσι τους χρονικούς στόχους με την οικονομική διάσταση του έργου.

Η αβεβαιότητα είναι παρούσα σε κάθε έργο οδηγώντας σε πολύ χαμηλή πιθανότητα το προϋπολογισμένο πρόγραμμα να εκτελεστεί ακριβώς όπως σχεδιάστηκε. Δραστηριότητες μπορεί να πάρουν λιγότερο ή περισσότερο χρόνο να εκτελεστούν από ότι εκτιμήθηκε, πόροι μπορεί ξαφνικά να σταματήσουν να είναι διαθέσιμοι, νέες δραστηριότητες να προστεθούν λόγω αλλαγών στους στόχους του έργου κτλ. Πιθανά λοιπον ένα φαινομενικά βέλτιστο πρόγραμμα μπορεί να βασίζεται σε παράλογες απαιτήσεις σε σχέση με την πραγματικότητα και να αποδειχθεί σημαντικά λιγότερο βέλτιστο όταν εκτελεστεί. Παρόμοια, ένα αρχικό πρόγραμμα φαινομενικά όχι βέλτιστο, που όμως είναι ευλύγιστο ώστε λαμβάνει υπόψη απροσδόκητα φαινόμενα, μπορεί να αποδειχθεί πολύ αποδοτικό κατά την εκτέλεσή του (Davenport et al., 2004).

Οι στόχοι του υπό μελέτη προβλήματος προγραμματισμού πολλαπλών έργων είναι η ελαχιστοποίηση της διάρκειας του συνολικού προγράμματος έργων που θα παραχθεί, αλλά και της διάρκειας του κάθε έργου ξεχωριστά, και η ταυτόχρονη διατήρηση του προγράμματος εύρωστου όπως και στην περίπτωση ενός μόνο έργου, ώστε να μπορεί να διαχειριστεί πιθανά απροσδόκητα φαινόμενα μικρής κλίμακας με αποδοτικό τρόπο.

### *4.1.2 Διαθέσιμοι Πόροι & Περιορισμοί*

Ένα πρόγραμμα έργου είναι ένα πλάνο που ορίζει ποιά δραστηριότητα πρέπει να εκτελεστεί, πότε πρέπει να αρχίσει η εκτέλεσή της και ποιά ποσότητα για κάθε πόρο απαιτείται για την εκτέλεσή της. Τα βασικά κομμάτια λοιπόν ενός προγράμματος έργου είναι οι δραστηριότητες και οι πόροι.

Μια δραστηριότητα μπορεί να έχει έναν ή περισσότερους τρόπους εκτέλεσης, μπορεί δηλαδή να εκτελεστεί χρησιμοποιώντας διάφορους συνδυασμούς τύπων και ποσοτήτων πόρων, με αποτέλεσμα διαφορετικές δυνατές διάρκειες για τη δραστηριότητα. Κάθε τρόπος αντιστοιχεί σε έναν εφικτό συνδυασμό πόρων και διάρκειας που επιτρέπουν την εκτέλεση της προκείμενης δραστηριότητας. Με τους πολλαπλούς τρόπους εκτέλεσης θεωρείται οτι είτε περισσότεροι πόροι του ίδιου τύπου, είτε πιο αποδοτικοί τύποι πόρων μπορούν να χρησιμοποιηθούν για την μείωση του χρόνου εκτέλεσης.

Για την έναρξη της εκτέλεσης μιας δραστηριότητας πρέπει όλοι οι άμεσοι προκάτοχοί της να έχουν προγραμματιστεί και ολοκληρωθεί. Αυτή η έννοια της προτεραιότητας εκφράζεται με σχέσεις τέλους-αρχής μεταξύ των δραστηριοτήτων.

Τα διαφορετικά είδη πόρων που χρησιμοποιούνται είναι δύο: ανανεώσιμοι και μη ανανεώσιμοι πόροι. Οι ανανεώσιμοι πόροι είναι περιορισμένοι ανά περίοδο και οι μη ανανεώσιμοι έχουν περιορισμένη διαθεσιμότητα για το σύνολο του έργου.

Τα δεδομένα εισόδου του προβλήματος όπως η διάρκειες των δραστηριοτήτων, οι απαιτήσεις τους σε πόρους, οι διαθεσιμότητες και τα είδη των πόρων και οι συσχετίσεις των έργων μεταξύ τους, ενώ στην πραγματικότητα είναι πολύ πιθανό να μην είναι ντετερμινιστικά δεδομένα εδώ θεωρούνται ντετερμινιστικά. Τα παραγόμενο πρόγραμμα-επίλυση του προβλήματος παρουσιάζεται τελικά στο διαχειριστή έργου και μπορεί να χρησιμοποιηθεί ως εργαλείο παρακολούθησης και ελέγχου της εκτέλεσης του έργου.

# 5. Μαθηματική Μοντελοποίηση Προβλήματος

## 5.1 Ορισμοί

Η προτεινόμενη παραλλαγή του προβλήματος προγραμματισμού με περιορισμένους πόρους μπορεί να διατυπωθεί εννοιολογικά με τον ακόλουθο τρόπο:

- Όλα τα δεδομένα θεωρείται ότι είναι ντετερμινιστικά και γνωστά εκ των προτέρων.
- Υπάρχουν $P$ έργα, όπου $P = 0,1, \dots, p$, καθένα από τα οποία αποτελείται από $n$ δραστηριότητες, συν μια βοηθητική δραστηριότητα αρχής $0_p$ που αναπαριστά την αρχή του έργου και μια βοηθητική δραστηριότητα λήξης $(n+1)_p$ που αναπαριστά το τέλος του έργου, και οι δύο με μηδενική διάρκεια και απαιτήσεις σε πόρους. Συμβολίζεται με $V_p = \{0_p, 1_p, \dots, n_p, (n+1)_p\}$ το σύνολο όλων των δραστηριοτήτων κάθε έργου $p$. Επιπλέον, ορίζεται μία βοηθητική υπέρ-δραστηριότητα αρχής $0$ που αντιπροσωπεύει την αρχή του συνόλου των έργων και μια βοηθητική υπέρ-δραστηριότητα λήξης $(n+1)$ που αντιπροσωπεύει το τέλος του συνόλου των έργων, και οι δύο με μηδενική διάρκεια και απαιτήσεις σε πόρους.
- Ορίζεται ο χρονικός ορίζοντας $T_p$ κάθε έργου $p$ ως το σύνολο της διάρκειας ολοκλήρωσης όλων των δραστηριοτήτων του.
- Ορίζεται ο χρονικός ορίζοντας $T_{tot}$ του συνόλου των έργων $P$ ως το σύνολο της διάρκειας ολοκλήρωσης του συνόλου των έργων $P$.
- Το σύνολο των ανανεώσιμων πόρων συμβολίζεται με $R^\rho$. Για κάθε τύπο ανανεώσιμου πόρου $k \in R^\rho$ η συνολική διαθεσιμότητα ανα χρονική περίοδο συμβολίζεται με $\alpha_k^\rho$.
- Το σύνολο των μη ανανεώσιμων πόρων συμβολίζεται με $R^\nu$. Για κάθε τύπο μη ανανεώσιμου πόρου $l \in R^\nu$ η συνολική διαθεσιμότητα για όλη τη διάρκεια του έργου συμβολίζεται με $\alpha_l^\nu$.
- Σε κάθε δραστηριότητα $i_p$ αντιστοιχίζεται ένα σύνολο $M_{i_p}$, το οποίο είναι το σύνολο των διαφορετικών τρόπων εκτέλεσης της δραστηριότητας $i_p$.
    - Κάθε δραστηριότητα $i_p$ κάθε έργου $p$ πρέπει να εκτελεστεί με ακριβώς ένα τρόπο $m \in M_{i_p}$ σε κάθε ξεχωριστό πρόγραμμα.
    - Κάθε τρόπος $m$ κάθε δραστηριότητας $i_p$ έχει μια καθορισμένη διάρκεια από $d_{i_p m}$ χρονικές μονάδες.
    - Κάθε δραστηριότητα $i_p$ σε τρόπο $m$ απαιτεί $r_{i_p mk}^\rho$ ανανεώσιμους πόρους τύπου $k \in R^\rho$ για κάθε χρονική περίοδο κατά την εκτέλεσή της. Οι απαιτούμενοι πόροι δεν καταναλώνονται αλλά χρησιμοποιούνται κατά τη

διάρκεια εκτέλεσης της δραστηριότητας και έπειτα επιστρέφονται στο απόθεμα των πόρων.

- Κάθε δραστηριότητα $i_p$ σε τρόπο $m$ απαιτεί την κατανάλωση $r^{\nu}_{i_p ml}$ μη ανανεώσιμων πόρων τύπου $l \in R^{\nu}$.

- Ορίζεται ως $s_{i_p m}$ ο χρόνος έναρξης της δραστηριότητας $i_p \in V_p$ σε τρόπο $m$.
- Ορίζεται ως $f_{i_p m}$ ο χρόνος λήξης της δραστηριότητας $i_p \in V_p$ σε τρόπο $m$.
- Η βοηθητική δραστηριότητα έναρξης κάθε έργου έχει ένα τρόπο $m = 0$ και διάρκεια $0$ χρονικές μονάδες. Συνεπώς, θέτοντας ένα έργο $p$ να αρχίσει να εκτελείται τη χρονική στιγμή μηδέν, δίνει $s_{o_p} = 0$.
- Η βοηθητική δραστηριότητα λήξης κάθε έργου έχει ένα τρόπο $m = 0$ και διάρκεια $0$ χρονικές μονάδες. Επομένως με $f_{(n+1)_p}$ αναπαριστάται η συνολική διάρκεια ή χρονικός ορίζοντας του έργου.
- Η βοηθητική υπερ-δραστηριότητα αρχής έχει ένα τρόπο $m = 0$ και διάρκεια 0 χρονικές μονάδες. Συνεπώς, θέτοντας το σύνολο όλων των έργων $P$ να ξεκινά τη χρονική στιγμή μηδέν, προκύπτει $s_0 = 0$.
- Η βοηθητική υπερ-δραστηριότητα λήξης έχει ένα τρόπο $m = 0$ και διάρκεια 0 χρονικές μονάδες Επομένως με $f_{n+1}$ αναπαριστάται η συνολική διάρκεια ή χρονικός ορίζοντας του συνόλου των έργων $P$.
- Ορίζονται δύο είδη σχέσεων προτεραιότητας, η εσωτερική σχέση τέλους-αρχής $FS_{i_p m j_p n}$ μεταξύ των δραστηριοτήτων $i$ και $j$ του έργου $p$, εκτελούμενων σε τρόπους $m$ και $n$ αντίστοιχα και η εξωτερική σχέση τέλους αρχής $FS_{i_p m j_q n}$ μεταξύ των δραστηριοτήτων $i$ του έργου $p$ και $j$ του έργου $q$, εκτελούμενων σε τρόπους $m$ και $n$ αντίστοιχα.
- Με βάση τους τύπους των σχέσεων προτεραιότητας μεταξύ των δραστηριοτήτων τα έργα διαιρούνται σε δύο συμπληρωματικά σύνολα, το $P_{in}$ που περιέχει όλα τα έργα που δεν περιλαμβάνουν δραστηριότητες που σχετίζονται εξωτερικά με δραστηριότητες άλλων έργων και το $P_{out}$ που περιέχει όλα τα έργα που έχουν μια ή περισσότερες δραστηριότητες που σχετίζονται με μια ή περισσότερες δραστηριότητες άλλων έργων. Είναι $P = P_{in} \cup P_{out}$ .
- Το σύνολο $P_{out}$ διαιρείται αυθαίρετα με βάση τον αριθμό των εξωτερικών εξαρτήσεων κάθε έργου σε τρία υποσύνολα $P_{out_l}$ , $P_{out_m}$ και $P_{out_h}$ που αναπαριστούν τα σύνολα έργων με χαμηλό, μεσαίο και υψηλό αριθμό εξαρτήσεων αντίστοιχα. Είναι $P_{out} = \left( P_{out_l} \cup P_{out_m} \cup P_{out_h} \right)$

- Όταν η δομή του κάθε έργου παρουσιάζεται με ένα δίκτυο δραστηριότητας-στον-κόμβο $G_p = (V_p, A_p)$ τότε το διάνυσμα $V_p = \{0,1,\dots,n,n+1\}$ περιέχει όλες της δραστηριότητες και το σύνολο των τόξων $A_p = \{(i,j)|i,j \in V_p, i \to j\}$ αναπαριστά τις σχέσεις προτεραιότητας μεταξύ τους. Πια αναλυτικά, θα υπάρχει ένα τόξο από τον κόμβο $i$ στον κόμβο $j$ αν και μόνο αν υπάρχει μια σχέση προτεραιότητας μεταξύ των δύο κόμβων.
- Όταν η δομή του συνόλου των έργων παρουσιάζεται με ένα υπερ-δίκτυο δραστηριότητας-στον-κόμβο $G_{tot} = (V_{tot}, A_{tot})$, τότε το διάνυσμα $V_{tot} = \sum_{\forall p \in P}\{0_p, 1_p, \dots, n_p, (n+1)_p\} + \{0, n+1\}$ περιέχει όλες της δραστηριότητες όλων των έργων και το σύνολο των τόξων $A_{tot} = \{(i,j)|i,j \in V_{tot}, i \to j\}$ αναπαριστά τις σχέσεις προτεραιότητας μεταξύ τους. Πια αναλυτικά, θα υπάρχει ένα τόξο από τον κόμβο $i$ στον κόμβο $j$ αν και μόνο αν υπάρχει μια σχέση προτεραιότητας μεταξύ των δύο κόμβων.
- Ο ελάχιστος και ο μέγιστος αριθμός σε πόρους τύπου $\tau$, $\tau \in \{R^\rho, R^\nu\}$, που χρησιμοποιείται ή καταναλώνεται από ένα συνδυασμό δραστηριότητας-τρόπου $[i_p, m]$ $i_p = 1, \dots, n$ και $m = 1, \dots, M_{i_p}$, για κάθε έργο $p \in P$, συμβολίζεται με $A_{\tau_p}^{min}$ και $A_{\tau_p}^{max}$ αντίστοιχα. Είναι η μικρότερη και η μέγιστη τιμή του συνόλου $A_{i_p m}^{\tau} = \left\{[i_p, m, \tau]\,;\, r_{i_p m w} > 0,\ w \in \tau,\ j = 1, \dots, n,\ m = 1, \dots, M_{i_p}\right\}$.
- Ορίζεται ο παράγοντας πόρων $RF_{\tau_p}$ για πόρους τύπου $\tau$, $\tau \in \{R^\rho, R^\nu\}$, ο οποίος δηλώνει το μέσο αριθμό πόρων τύπου $\tau$ που χρησιμοποιείται ή καταναλώνεται αντίστοιχα από κάθε έργο $p$. Ειδικότερα, ο πραγματικός παράγοντας πόρων $ARF_{\tau_p}$ για κάθε έργο υπολογίζεται ώς $ARF_{\tau_p} = \frac{1}{n}\sum_{i_p=1}^{n}\frac{1}{M_{i_p}}\frac{1}{|\tau|}\sum_{m=1}^{M_{i_p}}\left|A_{i_p m}^{\tau}\right|$ και ελέγχεται από το $\varepsilon_{RF_p}$, που είναι η ανοχή απόκλισης του παράγοντα πόρων έτσι ώστε $ARF_{\tau_p} \in \left[\left(1-\varepsilon_{RF_p}\right)\cdot RF_{\tau_p},\ \left(1+\varepsilon_{RF_p}\right)\cdot RF_{\tau_p}\right]$.
- Ορίζεται ο βαθμός δυναμικότητας ανανεώσιμων πόρων $RS_{pk}$ για κάθε έργο $p = P_{in} \cup P_{out}$ και κάθε πόρο $k \in R^\rho$. Είναι ένας βαθμός απόδοσης που εκφράζει την δυνατότητα κάλυψης των αναγκών του έργου σε πόρο $k$ ως τη σχέση της δεδομένης διαθεσιμότητας $\alpha_k^\rho$ του πόρου αυτού με ένα ελάχιστο και ένα μέγιστο επιπέδο απαιτούμενης από το έργο διαθεσιμότητας $\alpha_k^{min}$ και $\alpha_k^{max}$ αντίστοιχα. Η τιμή του δίνεται από τον τύπο $RS_{pk} = \left(\alpha_k^\rho - \alpha_k^{min}\right) / \left(\alpha_k^{max} - \alpha_k^{min}\right)$. Το ελάχιστο επίπεδο $\alpha_k^{min}$ είναι το ελάχιστο δυνατό επίπεδο διαθεσιμότητας ώστε να είναι εφικτό το έργο όσον αφορά το συγκεκριμένο ανανεώσιμο πόρο και είναι $\alpha_k^{min} =$

$max_{i_p=1}^{n} min_{m=1}^{M_{i_p}} \left\{ r_{i_p mk}^{\rho} \right\}$. Το μέγιστο επίπεδο $\alpha_k^{max}$ καθορίζεται μέσω του εξαρτώμενου από τον πόρο $k$ προγράμματος νωρίτερης έναρξης που λαμβάνεται όταν όλες οι δραστηριότητες του προγράμματος εκτελούνται στους ελάχιστους καταταγμένους πόρους $m_{i_p k}^{*}$ έχοντας μέγιστη ανά περίοδο χρήση του συγκεκριμένου πόρου. Είναι $m_{i_p k}^{*} = min\left\{ m \in \left\{1, \dots, M_{i_p}\right\}; \times r_{i_p mk}^{\rho} = max_{m=1}^{M_{i_p}} \left\{ r_{i_p mk}^{\rho} \right\} \right\}$. Το $\alpha_k^{max}$ δηλαδή ορίζεται από την μέγιστη ανά περίοδο χρήση του πόρου $k$ στο πρόγραμμα νωρίτερης έναρξης με εξάρτηση από τον πόρο αυτό.

- Ορίζεται ο βαθμός δυναμικότητας μη ανανεώσιμων πόρων $RS_{pl}$ για κάθε έργο $p \in P_{out}$ και κάθε πόρο $l \in R^{\nu}$. Είναι ένας βαθμός απόδοσης που εκφράζει την δυνατότητα κάλυψης των αναγκών του έργου σε πόρο $l$, ως τη σχέση της δεδομένης διαθεσιμότητας $\alpha_l^{\nu}$ του πόρου αυτού με ένα ελάχιστο και ένα μέγιστο επιπέδο απαιτούμενης από το έργο διαθεσιμότητας $\alpha_l^{min}$ και $\alpha_l^{max}$ αντίστοιχα. Η τιμή του δίνεται από τον τύπο $RS_{pl} = \left(\alpha_l^{\nu} - \alpha_l^{min}\right) / \left(\alpha_l^{max} - \alpha_l^{min}\right)$. Το ελάχιστο επίπεδο $\alpha_l^{min}$ και το μέγιστο επίπεδο $\alpha_l^{max}$ λαμβάνονται με συσσώρευση των καταναλώσεων που πραγματοποιούνται όταν εκτελείται κάθε διαδικασία στον τρόπο που απαιτεί τις ελάχιστες και μέγιστες καταναλώσεις αντίστοιχα. Είναι $\alpha_l^{min} = \sum_{i_p=1}^{n} min_{m=1}^{M_{i_p}} \left\{ r_{i_p ml}^{\nu} \right\}$ και $\alpha_l^{max} = \sum_{i_p=1}^{n} max_{m=1}^{M_{i_p}} \left\{ r_{i_p ml}^{\nu} \right\}$.
- Υπολογίζεται ο συνολικός βαθμός δυναμικότητας πόρων $RS_{tot_p}$ για κάθε έργο $p \in P_{out}$ και για το σύνολο των ανανεώσιμων πόρων $k \in R^{\rho}$ και των μη ανανεώσιμων πόρων $l \in R^{\nu}$ ως $RS_{tot_p} = \sum_{\forall l \in R^{\nu}} w_l \cdot RS_{pl} + \sum_{\forall k \in R^{\rho}} w_{\rho} \cdot RS_{pk}$ όπου $w$ τα βάρη για κάθε τύπο πόρου, που δηλώνουν την σχετική αξία του ως προς τους υπόλοιπους πόρους του έργου, και είναι $\sum_{\forall l \in R^{\nu}} w_l + \sum_{\forall k \in R^{\rho}} w_{\rho} = 1$.
- Τα έργα που περιλαμβάνονται στο σύνολο $P$ κατηγοριοποιούνται με βάση το συνολικό πραγματικό παράγοντα πόρων τους $ARF_{tot}$. Όταν οι διαφορές των συνολικών πραγματικών παραγόντων πόρων δύο ή περισσότερων έργων δεν είναι αρκετά μεγάλες ώστε να γίνει σαφής κατάταξή τους, το φαινόμενο καλείται «ισοπαλία» μεταξύ έργων και λύνεται με σύγκριση του συνολικού βαθμού δυναμικότητάς $RS_{tot_p}$ κάθε έργου.
- Το διάνυσμα $S = \left(s_0, s_{i_p m}, s_{n+1}\right)_{i_p=0,1,\dots,n,n+1,\ p=0,\dots,P,\ m=0,\dots,M_{i_p}}$ ορίζει ένα πρόγραμμα του συνόλου των έργων. Ένα πρόγραμμα $S$ καλείται εφικτό όταν όλοι οι περιορισμοί πόρων και προτεραιοτήτων έχουν καλυφθεί.

- Συμβολίζουμε με $Act(t)$ όλες τις δραστηριότητες των οποίων μία χρονική μονάδα της διάρκειας εκτέλεσής τους είναι σε εξέλιξη κατά τη χρονική στιγμή $t$, $t = 0,1, \dots, T_{tot}$ .

Ο στόχος είναι να καθοριστούν οι τρόποι εκτέλεσης $m$ και οι χρόνοι έναρξης $s_{i_p m}$ για όλες τις δραστηριότητες $i = 1, \dots, n$ και όλα τα έργα $p = 0, \dots, P$ με τέτοιο τρόπο ώστε οι στόχοι του προβλήματος να βελτιστοποιηθούν, ενώ όλοι οι δεδομένοι περιορισμοί υπακούονται.

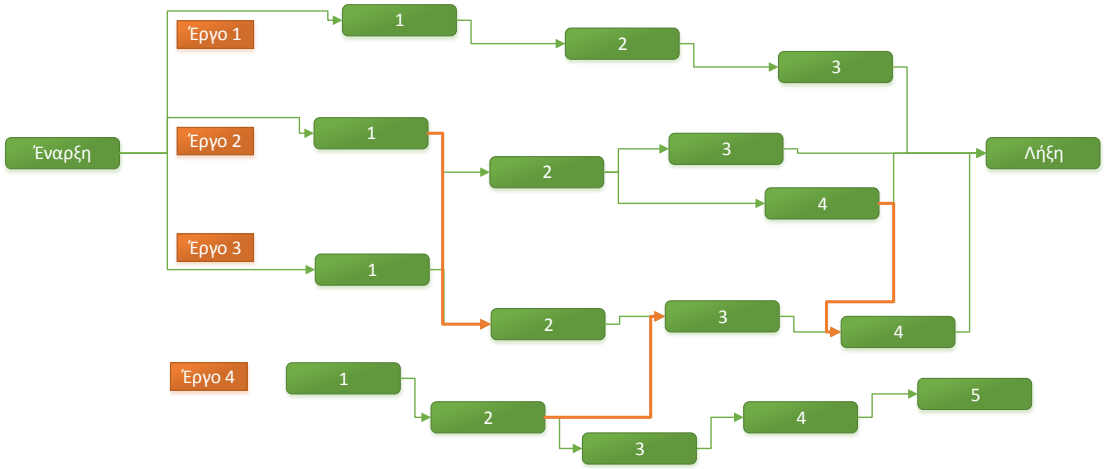
Ο Πίνακας 5.1 που ακολουθεί συνοψίζει τη σημειογραφία που παρουσιάστηκε στην παράγραφο αυτή.

| **Σύμβολο** | **Ορισμός** |
|---|---|
| $\boldsymbol{P}$ | Ο αριθμός των έργων $p$ |
| $\boldsymbol{n_p}$ | Ο αριθμός των πραγματικών δραστηριοτήτων $i$ κάθε έργου $p$ |
| $\boldsymbol{V_p} = \{\boldsymbol{0_p}, \boldsymbol{1_p}, \dots, \boldsymbol{n_p}, (\boldsymbol{n+1})_p\}$ | Το σύνολο των δραστηριοτήτων $i_p$ κάθε έργου $p$ |
| $\boldsymbol{A_p} = \{(\boldsymbol{i},\boldsymbol{j}) \mid \boldsymbol{i},\boldsymbol{j} \in \boldsymbol{V_p}, \boldsymbol{i} \to \boldsymbol{j}\}$ | Το σύνολο των τόξων που αναπαριστά τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων $i_p$ κάθε έργου $p$ |
| $\boldsymbol{G_p} = (\boldsymbol{V_p}, \boldsymbol{A_p})$ | Προσανατολισμένο δίκτυο των σχέσεων προτεραιότητας μεταξύ των δραστηριοτήτων $i_p$ κάθε έργου $p$ |
| $\boldsymbol{V_{tot}} = \sum_{\forall \boldsymbol{p} \in \boldsymbol{P}} \{\boldsymbol{0_p}, \boldsymbol{1_p}, \dots, \boldsymbol{n_p}, (\boldsymbol{n+1})_p\} + \{\boldsymbol{0}, \boldsymbol{n+1}\}$ | Το σύνολο των δραστηριοτήτων όλων των έργων. |
| $\boldsymbol{A_{tot}} = \{(\boldsymbol{i},\boldsymbol{j}) \mid \boldsymbol{i},\boldsymbol{j} \in \boldsymbol{V_{tot}}, \boldsymbol{i} \to \boldsymbol{j}\}$ | Το σύνολο των τόξων που αναπαριστά τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων όλων των έργων |
| $\boldsymbol{G_{tot}} = (\boldsymbol{V_{tot}}, \boldsymbol{A_{tot}})$ | Προσανατολισμένο δίκτυο των σχέσεων προτεραιότητας μεταξύ των δραστηριοτήτων όλων των έργων |
| $\boldsymbol{T_p}$ | Ο χρονικός ορίζοντας κάθε έργου $p$ |
| $\boldsymbol{T_{tot}}$ | Ο χρονικός ορίζοντας του συνόλου των έργων $P$ |
| $\boldsymbol{t}$ | Χρονική στιγμή, όπου $t = 0,1, \dots, T_{tot}$ |
| $\boldsymbol{Act(t)}$ | Το σύνολο των δραστηριοτήτων ρων οποίων μια χρονική μονάδα της διάρκειας εκτέλεσής τους είναι σε εξέλιξη τη χρονική στιγμή t, $t = 0,1, \dots, T_{tot}$ |
| $\boldsymbol{R^{\rho}}$ | Το σύνολο των ανανεώσιμων πόρων |
| $\boldsymbol{\alpha_k^{\rho}}$ | Η διαθεσιμότητα σε μονάδες ανά χρονική περίοδο για κάθε ανανεώσιμο πόρο $k \in R^{\rho}$ |
| $\boldsymbol{R^{\nu}}$ | Το σύνολο των μη ανανεώσιμων πόρων |

| | |
|---|---|
| $\boldsymbol{\alpha_l^v}$ | Η συνολική διαθεσιμότητα για όλη τη διάρκεια του έργου για κάθε μη ανανεώσιμο πόρο $l \in R^v$ |
| $\boldsymbol{M_{i_p}}$ | Το σύνολο των διαφορετικών τρόπων εκτέλεσης της δραστηριότητας $i_p$ |
| $\boldsymbol{d_{i_p m}}$ | Η διάρκεια της δραστηριότητας $i_p$ σε τρόπο $m$ |
| $\boldsymbol{r^{\rho}_{i_p mk}}$ | Η ανά περίοδο χρήση από τη δραστηριότητα $i_p$ του ανανεώσιμου πόρου $k$ στον τρόπο $m$ |
| $\boldsymbol{r^{v}_{i_p ml}}$ | Η κατανάλωση από τη δραστηριότητα $i_p$ του μη ανανεώσιμου πόρου $l$ στον τρόπο $m$ |
| $\boldsymbol{s_{i_p m}}$ | Ο χρόνος έναρξης της δραστηριότητας $i_p$ σε τρόπο $m$. |
| $\boldsymbol{f_{i_p m}}$ | Ο χρόνος λήξης της δραστηριότητας $i_p$ σε τρόπο $m$. |
| $\boldsymbol{s_{o_p}}$ | Ο χρόνος έναρξης του έργου $p$ |
| $\boldsymbol{f_{(n+1)_p}}$ | Ο χρόνος λήξης του έργου $p$ |
| $\boldsymbol{s_0}$ | Ο χρόνος έναρξης του συνόλου των έργων $P$ |
| $\boldsymbol{f_{n+1}}$ | Ο χρόνος λήξης του συνόλου των έργων $P$ |
| $\boldsymbol{FS_{i_p m j_p n}}$ | Η εσωτερική σχέση τέλους-αρχής μεταξύ των δραστηριοτήτων $i$ και $j$ του έργου $p$, εκτελούμενων σε τρόπους $m$ και $n$ αντίστοιχα |
| $\boldsymbol{FS_{i_p m j_q n}}$ | Η εξωτερική σχέση τέλους αρχής μεταξύ των δραστηριοτήτων $i$ του έργου $p$ και $j$ του έργου $q$, εκτελούμενων σε τρόπους $m$ και $n$ αντίστοιχα |
| $\boldsymbol{P_{in}}$ | Το σύνολο που περιέχει όλα τα έργα που δεν περιλαμβάνουν δραστηριότητες που σχετίζονται εξωτερικά με δραστηριότητες άλλων έργων |
| $\boldsymbol{P_{out}}$ | Το σύνολο που περιέχει όλα τα έργα που έχουν μια ή περισσότερες δραστηριότητες που σχετίζονται με μια ή περισσότερες δραστηριότητες άλλων έργων |
| $\boldsymbol{P_{out_l}}$ | Υποσύνολο του $P_{out}$ που αναπαριστά το σύνολο των έργων με χαμηλό αριθμό εξαρτήσεων |
| $\boldsymbol{P_{out_m}}$ | Υποσύνολο του $P_{out}$ που αναπαριστά το σύνολο των έργων με μεσαίο αριθμό εξαρτήσεων |
| $\boldsymbol{P_{out_h}}$ | Υποσύνολο του $P_{out}$ που αναπαριστά το σύνολο των έργων με υψηλό αριθμό εξαρτήσεων |
| $\boldsymbol{S = \left(s_0, s_{i_p m}, s_{n+1}\right)_{i_p=0,1,\ldots,n,n+1,\ p=0,\ldots,P,\ m=0,\ldots,M_{i_p}}}$ | Ένα πρόγραμμα - διάνυσμα των χρόνων έναρξης όλων των δραστηριοτήτων όλων των έργων |
| $\boldsymbol{A^{min}_{\tau_p} / A^{max}_{\tau_p}}$ | Η ελάχιστη/μέγιστη απαιτούμενη ποσότητα σε πόρο τύπου $\tau$, $\tau \in \{R^{\rho}, R^{v}\}$ , που χρησιμοποιείται ή καταναλώνεται από ένα συνδυασμό δραστηριότητας-τρόπου $[i_p, m]$, για κάθε έργο $p \in P$ |
| $\boldsymbol{RF_{\tau_p}}$ | Ο παράγοντας πόρων για πόρους τύπου $\tau$, $\tau \in \{R^{\rho}, R^{v}\}$, για κάθε έργο $p$ |
| $\boldsymbol{ARF_{\tau_p}}$ | Ο πραγματικός παράγοντας πόρων για πόρους |

| | |
|---|---|
| | τύπου $\tau$, $\tau \in \{R^{\rho}, R^{\nu}\}$, για κάθε έργο $p$ |
| $\boldsymbol{\varepsilon_{RF_p}}$ | Η ανοχή απόκλισης του παράγοντα πόρων |
| $\boldsymbol{ARF_{tot_p}}$ | Ο συνολικός πραγματικός παράγοντας πόρων για κάθε έργο $p$ |
| $\boldsymbol{RS_{pk}}$ | Ο βαθμός δυναμικότητας ανανεώσιμων πόρων για κάθε έργο $p \in P_{out}$ και κάθε πόρο $k \in R^{\rho}$ |
| $\boldsymbol{\alpha_k^{min}/\alpha_k^{max}}$ | Ελάχιστο/μέγιστο επιπέδο απαιτούμενης από το έργο διαθεσιμότητας για πόρο $k \in R^{\rho}$ |
| $\boldsymbol{RS_{pl}}$ | Ο βαθμός δυναμικότητας μη ανανεώσιμων πόρων για κάθε έργο $p \in P_{out}$ και κάθε πόρο $l \in R^{\nu}$ |
| $\boldsymbol{\alpha_l^{min}/\alpha_l^{max}}$ | Ελάχιστο/μέγιστο επιπέδο απαιτούμενης από το έργο διαθεσιμότητας για πόρο $l \in R^{\nu}$ |
| $\boldsymbol{RS_{tot_p}}$ | Ο συνολικός βαθμός δυναμικότητας πόρων για κάθε έργο $p \in P_{out}$, για το σύνολο των ανανεώσιμων πόρων $k \in R^{\rho}$ και των μη ανανεώσιμων πόρων $l \in R^{\nu}$ |

Ακολουθεί ένα παράδειγμα όπου επεξηγούνται περεταίρω οι παραπάνω ορισμοί, όπου οι σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων κάθε έργου και μεταξύ των έργων απεικονίζονται στο Σχήμα 5.1 και οι ανάγκες κάθε έργου σε πόρους καθώς και οι διάφοροι τρόποι εκτέλεσης των δραστηριοτήτων του δίνονται στον Πίνακα 5.1.




**Σχήμα 5.1 Σχέσεις προτεραιότητας μεταξύ των έργων**

**Πίνακας 5.1 Έργα, δραστηριότητες, τρόποι εκτέλεσης, ανάγκες σε πόρους για κάθε έργο και διαθεσιμότητες πόρων**

| Έργο | Δραστηριότητες | Τρόποι εκτέλεσης | Διάρκεια | Ανανεώσιμος πόρος | Μη ανανεώσιμος πόρος |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 |
| | 2 | 1 | 2 | 2 | 1 |
| | | 2 | 4 | 1 | 1 |
| | 3 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 | 0 |
| | 2 | 1 | 2 | 3 | 0 |
| | | 2 | 3 | 2 | 0 |
| | 3 | 1 | 2 | 1 | 0 |
| | | 2 | 1 | 2 | 0 |
| | 4 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 0 |
| | 2 | 1 | 1 | 2 | 1 |
| | | 2 | 3 | 1 | 1 |
| | 3 | 1 | 1 | 1 | 3 |
| | | 2 | 2 | 2 | 0 |
| | 4 | 1 | 3 | 1 | 2 |
| | 1 | 2 | 2 | 1 | 3 |
| | 5 | 1 | 0 | 0 | 0 |
| 4 | 1 | 1 | 0 | 0 | 0 |
| | 2 | 1 | 3 | 1 | 1 |
| | | 2 | 2 | 2 | 3 |
| | 3 | 1 | 1 | 1 | 5 |
| | | 2 | 2 | 2 | 2 |
| | 4 | 1 | 1 | 2 | 1 |
| | | 2 | 3 | 1 | 1 |
| | 5 | 1 | 0 | 0 | 0 |
| Συνολικές Διαθεσιμότητες | | | | 6 | 15 |

Κάθε έργο έχει μια βοηθητική δραστηριότητα αρχής και μια βοηθητική δραστηριότητα τέλους, ενώ το υπερ-έργο έχει μια βοηθητική υπερ-δραστηριότητα αρχής και μια βοηθητική υπερ-δραστηριότητα τέλους με μηδενική διάρκεια και απαιτήσεις σε πόρους. Κάθε δραστηριότητα έχει το πολύ δύο διαφορετικούς τρόπους εκτέλεσης και για κάθε έναν ορίζονται η διάρκεια και οι απαιτήσεις σε πόρους της δραστηριότητας. Ακολουθεί ο Πίνακας 5.2 που παρουσιάζει την κατηγοριοποίηση των έργων σε υποσύνολα με βάση τις εξωτερικές συσχετίσεις μεταξύ τους και κατάταξης των έργων κάθε υποσυνόλου σύμφωνα με τις απαιτήσεις του σε πόρους (μέσω του συντελεστή $ARF$ κάθε έργου).

**Πίνακας 5.2 Κατηγοριοποίηση των έργων σε υποσύνολα και κατάταξή τους εντός των υποσυνόλων**

| Έργο | Υποσύνολο | Κατάταξη | $ARF$ |
|---|---|---|---|
| 1 | $P_{in}$ | 1 | 1 |
| 2 | $P_{out_1}$ | 2 | 0.5 |
| 3 | $P_{out_2}$ | 1 | 1 |
| 4 | $P_{out_1}$ | 1 | 1 |

## 5.2 Αντικειμενικοί Στόχοι

Κατά την δημιουργία του προγράμματος πολλαπλών έργων αναζητούμε ένα πρόγραμμα βάσης για το σύνολο των έργων, του οποίου ο στόχος είναι η ελαχιστοποίηση της χρονικής διάρκειας κάθε έργου και της χρονικής διάρκειας του συνόλου των έργων, ταυτόχρονα.

Ο στόχος της ελαχιστοποίησης της χρονικής διάρκειας κάθε έργου ξεχωριστά και της χρονικής διάρκειας του συνόλου τους μπορεί να εκφαστεί ως η ελαχιστοποίηση των χρόνων λήξης των βοηθητικών δραστηριοτήτων τέλους κάθε έργου και του χρόνου λήξης της βοηθητικής υπερ-δραστηριότητας τέλους:

(5.1) $$\min \begin{cases} f_{(n+1)_p} \ \forall p \in P \\ f_{n+1} \end{cases}$$

## 5.3 Περιορισμοί

Οι δραστηριότητες πρέπει να εκτελεστούν με συγκεκριμένη σειρά που δίνεται από τις σχέσεις προτεραιότητας μεταξύ τους:

(5.2) $$s_{i_p m} \geq f_{j_p n} \ \forall (i,j) \in A_p, \forall p \in P_{in}, \ \forall m \in M_{i_p}, \forall n \in N_{j_p}$$

(5.3) $$s_{i_p m} \geq f_{j_q n} \ \forall (i,j) \in A_{tot}, \forall p,q \in P_{out}, \ \forall m \in M_{i_p}, \forall n \in N_{j_q}$$

Η βοηθητική υπερ-δραστηριότητα αρχής πρέπει πάντα να είναι η πρώτη δραστηριότητα που προγραμματίζεται, σε χρόνο $t = 0$:

(5.4) $$s_0 = 0$$

Αντίστοιχα, η βοηθητική υπερ-δραστηριότητα τέλους πρέπει πάντα να είναι η τελευταία που προγραμματίζεται:

(5.5) $$f_{n+1} \geq f_{(n+1)_p} \ \forall p \in P$$

Η χρήση ανανεώσιμων και μη-ανανεώσιμων πόρων κάθε χρονική στιγμή $t$ πρέπει να είναι μικρότερη ή ίση με την διαθέσιμη ποσότητα:

(5.6) $$\sum_{p\in P}\sum_{i\in Act(t)} r^{\rho}_{i_p mk} \le a^{\rho}_{k} \;\; \forall\, k \in R^{\rho}, \forall m \in M_{i_p}, \forall t \in \{0,1,\dots,T_{tot}-1\}$$

(5.7) $$\sum_{p\in P}\sum_{i_p\in V_p} r^{\nu}_{i_p ml} \le a^{\nu}_{l} \;\; \forall\, l \in R^{\nu}, \forall m \in M_{i_p}$$

Στην προτεινόμενη επίλυση ο περιορισμός που ορίζεται από την σχέση 5.5 είναι «μαλακός» και όχι «σκληρός» περιορισμός, δηλαδή μπορεί να παραβιαστεί και το γεγονός αυτό κωδικοποιείται με την εισαγωγή συνάρτησης ποινής στην αντικειμενική συνάρτηση, η οποία μετασχηματίζεται ως εξής:

(5.8) $$\min \begin{cases} f_{(n+1)_p} \;\forall p \in P \\ f_{n+1} + Penalty \end{cases}$$

Όπου, αν η συνθήκη 5.7 παραβιάζεται τότε είναι:

(5.9) $$Penalty = \left(\sum_{l\in R^{\nu}} \left|\alpha^{\nu}_{l} - \sum_{p\in P}\sum_{i_p\in V_p} r^{\nu}_{i_p ml}\right|\right) \times c \;\;,\;\; c = const.$$

Αλλιώς:

(5.10) $$Penalty = 0$$

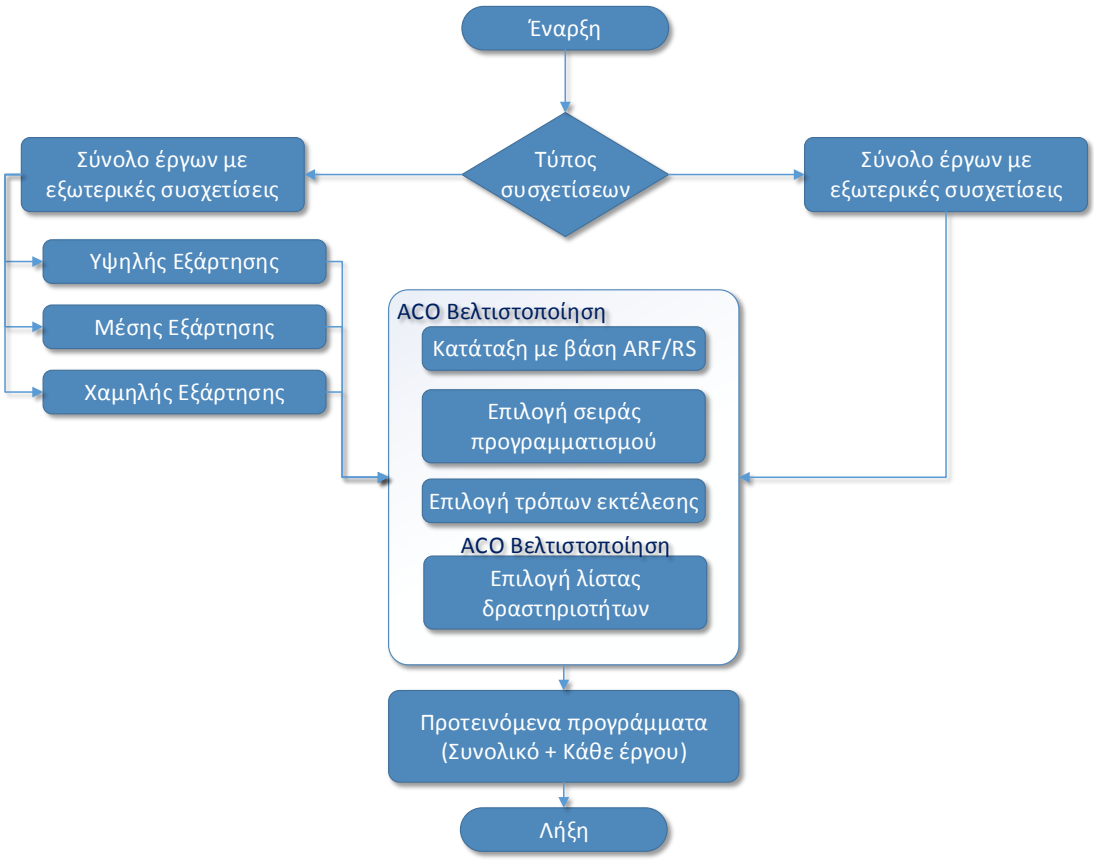
Στην προτεινόμενη επίλυση η εισαγωγή των ποινών στην αντικειμενική συνάρτηση χρησιμοποιείται αφενός για να δώσει ευελιξία στη μέθοδο κατά την αναζήτηση λύσεων και αφετέρου γιατί είναι δύσκολο να βρεθεί εφικτό πρόγραμμα για το πρόβλημα που μελετάται εδώ, αν κρατηθεί «σκληρός» ο περιορισμός 5.7.

# 6. Διαδικασία Επίλυσης

## 6.1 Πολλαπλά Έργα & Συσχετίσεις Μεταξύ Τους

Όταν το πρόβλημα προγραμματισμού έργων επεκτείνεται στον προγραμματισμό πολλαπλών έργων, ανακύπτει η ανάγκη επιλογής του κατάλληλου τρόπου αντιμετώπισης του νέου προβλήματος. Η συνήθης μέθοδος που ακολουθείται στην περίπτωση του προβλήματος πολλαπλών έργων είναι η μετατροπή του από πρόβλημα πολλαπλών έργων σε πρόβλημα ενός έργου, μέσα από μια συγχώνευση των δικτύων των έργων. Αυτή η μέθοδος μπορεί να οδηγήσει σε ογκώδη, πολύπλοκα δίκτυα, των οποίων η επίλυση συχνά απαιτεί μεγάλη δαπάνη χρόνου, κάτι που δεν βοηθά τον διαχειριστή έργου.

Εδώ προτείνεται μια διαφορετική μέθοδος βασισμένη σε διπλή βελτιστοποίηση με αποικία μυρμηγκιών (ACO), η οποία αναπαριστάται στο σχήμα 6.1.




**Σχήμα 6.1 Διαγραμματική αναπαράσταση διαδικασίας επίλυσης**

Η διαφοροποιήση έγκειται στην θεώρηση πιθανής ύπαρξης σχέσεων μεταξύ μιας ή περισσότερων δραστηριοτήτων ενός έργου με δραστηριότητες ενός ή περισσότερων άλλων έργων. Αρχικά δημιουργούνται δύο σύνολα έργων, ένα σύνολο

έργων ανεξάρτητων από τα υπόλοιπα και ένα σύνολο έργων με εξωτερικές εξαρτήσεις. Το δεύτερο αυτό σύνολο, των έργων με εξωτερικές εξαρτήσεις, διαιρείται με βάση τον αριθμό των εξωτερικών εξαρτήσεων κάθε έργου σε τρία υποσύνολα: μικρής, μεσαίας και υψηλής εξάρτησης.

Ακολουθεί ανάλυση των απαιτήσεων σε πόρους για το κάθε έργο κάθε υποσυνόλου και γίνεται κατηγοριοποίησή τους εντός του υποσυνόλου σύμφωνα με τις απαιτήσεις κάθε έργου σε πόρους. Έτσι κατά τον προγραμματισμό τους, πρώτα προγραμματίζονται τα έργα που ανήκουν στο σύνολο με υψηλές εξαρτήσεις και έχουν υψηλότερες απαιτήσεις σε πόρους. Ακολουθούν τα έργα με υψηλές εξαρτήσεις και χαμηλότερες απαιτήσεις σε πόρους και ούτω καθεξής. Ο προτεινόμενος αλγόριθμος χρησιμοποιείται για την επίλυση τόσο του MRCPSP όσο και για την βέλτιστη κατηγοριοποίηση των έργων στα σύνολα εξαρτήσεων και την ιεράρχηση των έργων κάθε συνόλου.

## 6.2 Παρουσίαση Αλγορίθμου Επίλυσης

### 6.2.1 *Βασικό Σχέδιο Αλγορίθμου*

Ο προτεινόμενος αλγόριθμος είναι ένας τρόπος επίλυσης της επέκτασης του RCPSP που ονομάζεται Multi-Project MRCPSP. Χρησιμοποιείται στην εκδοχή των προβλημάτων προγραμματισμού με στόχο την ελαχιστοποίηση του χρόνου ολοκλήρωσης του συνόλου των υποέργων αλλά και κάθε υποέργου ξεχωριστά. Επίσης είναι προσαρμόσιμος και σε άλλες υπάρχουσες παραλλαγές του RCPSP, μπορεί δηλαδή να επιλύσει αποτελεσματικά και αποδοτικά το κλασικό RCPSP και το multi-mode RCPSP (MRCPSP), έχοντας απαιτήσεις και περιορισμούς σε ανανεώσιμους ή/και μη ανανεώσιμους πόρους.

Το βασικό κομμάτι της όλης διαδικασίας είναι ένας ACO αλγόριθμος που λειτουργεί ως διαχειριστής της διαδικασίας επίλυσης. Ουσιασικά, η διαδικασία επίλυσης επιτρέπει στο Διαχειριστή (Moderator) ACO αλγόριθμο να ορίσει τη σειρά προγραμματισμού κάθε έργου και τους τρόπους εκτέλεσης των δραστηριοτήτων κάθε έργου, αναθέτοντας σε έναν δεύτερο ACO αλγόριθμο, τον Activity List ACO, την εύρεση βέλτιστου προγράμματος για κάθε ένα από τα επιμέρους έργα. Αυτό επιτυγχάνεται με χρήση επιπρόσθετων θέσεων στα διανύσματα-προγράμματα που χειρίζεται ο Διαχειριστής ACO για την ενεργοποίηση της εξελικτικής διαδικασίας απόφασης του ποιά είναι η βέλτιστη σειρά προτεραιότητας με βάση την οποία θα προγραμματιστούν τα έργα, σύμφωνα με τις εξωτερικές εξαρτήσεις τους και τις ανάγκες τους σε πόρους. Με άλλα λόγια, η βελτιστοποίηση δεν περιορίζεται στην

εύρεση του βέλτιστου προγράμματος για κάθε έργο, αλλά επεκτείνεται και στο βέλτιστο προγραμματισμό του συνόλου τους. Έτσι, αντικείμενο της βελτιστοποίησης είναι οι λίστες με της δραστηριότητες κάθε έργου, οι λίστες με τους τρόπους εκτέλεσης των δραστηριοτήτων κάθε έργου και η σειρά προγραμματισμού των έργων.

Αρχικά, τα δεδομένα εισόδου αναλύονται και μετασχηματίζονται σε μια ενιαία, προκαθορισμένη μορφή ώστε να διευκολυνθεί η χρήση τους και πραγματοποιείται μια προεπεξεργασία τους, προς εξάλειψη των περιττών δεδομένων, όπως είναι για παράδειγμα οι πλεονάζοντες τύποι πόρων που δεν επηρεάζουν την διαδικασία προγραμματισμού και οι μη αποδοτικοί ή μη εκτελέσιμοι τρόποι εκτέλεσης δραστηριοτήτων. Σε δεύτερο στάδιο της προεπεξεργασίας γίνεται ο υπολογισμός των συντελεστών $ARF$ και $RS_{tot}$ για κάθε έργο. Έπειτα μια αρχική λύση δημιουργείται και οι μηχανισμοί της ανανέωσης και εξάτμισης της φερομόνης και του πιθανοτικού κατασκευαστικού μοντέλου εφαρμόζονται για την παραγωγή του πληθυσμού της επόμενης γενιάς επιλύσεων-προγραμμάτων.

Το κάθε διάνυσμα (αναπαράσταση ενός έργου) που χειρίζονται τα μυρμήγκια του Διαχειριστή ACO και του Activity List ACO συντίθεται από τρία μέρη: την *Λίστα Δραστηριοτήτων (Activity List)* που είναι μια κατάταξη σε τυχαία σειρά των προς προγραμματισμό δραστηριοτήτων του έργου, τη *Λίστα Τρόπων (Mode List)* που αναπαριστά τους επιλεγμένους τρόπους εκτέλεσης για κάθε δραστηριότητα του έργου και το *Στοιχείο Θέσης Υποσυνόλου (Subset Cell)* που αναπαριστά το υποσύνολο στο οποίο ανήκει το έργο. Η δομή του διανύσματος δίνεται στο σχήμα που ακολουθεί.

Ο Διαχειριστής ACO αλγόριθμος προκειμένου να δημιουργήσει κάθε επόμενη γενιά λύσεων χρησιμοποιεί ένα Μοντέλο Φερομόνης για τη Λίστα Τρόπων και ένα δεύτερο Μοντέλο Φερομόνης για τα Στοιχεία Θέσης Υποσυνόλου, ενώ ο βοηθητικός Activity List ACO χρησιμοποιεί ένα τρίτο Μοντέλο Φερομόνης για την Λίστα Δραστηριοτήτων κάθε έργου.

Επομένως, ο Διαχειριστής ACO είναι υπεύθυνος για την εύρεση των υποσυνόλων και της κατηγοριοποίησης των έργων σε υποσύνολα καθώς και την επιλογή του τρόπου εκτέλεσης κάθε δραστηριότητας κάθε έργου, ενώ ο Activity List ACO είναι υπεύθυνος για την αναζήτηση εντός του χώρου λύσεων για το καλύτερο πρόγραμμα, με βάση τα διανύσματα που δέχεται ως είσοδο. Σαν αποτέλεσμα, ο Activity List ACO ψάχνει τον χώρο λύσεων για το βέλτιστο πρόγραμμα δραστηριοτήτων σε μια εσωτερική επανάληψη, ενώ εξωτερικά ο Moderator ACO αναζητά τον βέλτιστο τρόπο εκτέλεσης για κάθε δραστηριότητα κάθε έργου και ταυτόχρονα την καλύτερη δυνατή κατανομή των έργων σε υποσύνολα με βάση τις

εξαρτήσεις τους και τις ανάγκες τους σε πόρους. Η καταλληλότητα κάθε προγράμματος-λύσης ισούται με την τιμή της αντικειμενικής συνάρτησης κάθε κωδικοποιημένου διανύσματος.

Στους Αλγορίθμους 6.1 και 6.2 παρουσιάζονται σε μορφή ψευδοκώδικα ο Moderator ACO και ο Activity List ACO αλγόριθμος αντίστοιχα.

**Αλγόριθμος 6.1 Moderator ACO αλγόριθμος**

set Generations $= Gen$;

set Population $= Pop$;
set Projects′ Number $= ProjNum$;
set $UpdateRule$;
Calculate $Initial\ Population$;
Initialize $ModePheromoneModel$;
Initialize $SubsetPheromoneModel$;
$\boldsymbol{for}\ i = 0 \dots Gen\ \boldsymbol{do}$
    $\boldsymbol{for}\ i = 0 \dots Pop\ \boldsymbol{do}$
        $\boldsymbol{for}\ i = 0 \dots ProjNum\ \boldsymbol{do}$
            $AntBasedModeListConstruction$;
            $AntBasedSubsetSelection$;
            Calculate $Activity\ List$ with $ActivityList\ ACO$;
        $\boldsymbol{end}$
    $\boldsymbol{end}$
    Select $ProjectsSchedules$ with $IB/BS\ Update\ Rule$;
    Transform $Projects\ to\ SuperProject$;
    Calculate $SuperProjectSchedule$ with $ActivityListACO$;
    $ModePheromoneUpdate$;
    $SubsetPheromoneUpdate$;
$\boldsymbol{end}$
return $SuperProjectSchedule$; $ProjectsSchedules$;

**Αλγόριθμος 6.2 Activity List ACO αλγόριθμος**

set Generations $= Gen$;

set Population $= Pop$;
set $UpdateRule$;
Calculate $Initial\ Population$;
Initialize $ActivityListPheromoneModel$;
***for*** $i = 0 \dots Gen$ ***do***
    ***for*** $i = 0 \dots Pop$ ***do***
        $AntBasedModeListConstruction$;
    ***end***
    Select $ProjectSchedule$ with $IB/BS\ UpdateRule$;
    $ActivityListPheromoneModelUpdate$;
***end***
return $ProjectSchedule$;

### *6.2.2 Προεπεξεργασία*

Στο συγκεκριμένο μοντελοποιημένο πρόβλημα η ταυτόχρονη ύπαρξη πολλών παραμέτρων και μεταβλητών όπως η διαθεσιμότητα ανανεώσιμων και μη ανανεώσιμων πόρων, οι διαφορετικοί τρόποι εκτέλεσης των δραστηριοτήτων των έργων και οι εξαρτήσεις μεταξύ των δραστηριοτήτων του ίδιου ή διαφορετικών έργων οδηγούν σε μια αρκετά πολύπλοκη κατάσταση κατά την προσπάθεια παραγωγής ενός εφικτού προγράμματος. Προς χειρισμό της κατάστασης αυτής εφαρμόζεται μια διαδικασία προεπεξεργασίας επί των δεδομένων του έργου για τη μείωση της πολυπλοκότητας της διαδικασίας, μετατρέποντας τις εισόδους σε πιο διαχειρίσιμη μορφή, προετοιμάζοντας το διαδικασία επίλυσης και περιορίζοντας τον χώρο αναζήτησης λύσεων.

Το σύνολο των μη ανανεώσιμων πόρων μπορεί να περιέχει πόρους των οποίων η διαθεσιμότητα να είναι μεγαλύτερη από τη μέγιστη ζήτηση, οι οποίοι καλούνται πλεονάζοντες ή περιττοί. Αυτοί οι τύποι πόρων δεν περιορίζουν τον προγραμματισμό του συνολικού υπερ-έργου καθώς βρίσκονται σε αφθονία, επομένως δεν επηρεάζουν τη διαδικασία του προγραμματισμού και μπορούν να αφαιρεθούν από το υπολογιστικό κομμάτι της διαδικασίας. Ένας μη ανανεώσιμος τύπος πόρου ορίζεται ως πλεονάζων όταν το άθροισμα της μέγιστης χρήσης αυτού του τύπου πόρου για όλες τις δραστηριότητες όλων των έργων, που δίνεται

παίρνοντας τον τρόπο εκτέλεσης που έχει τη μέγιστη χρήση αυτού του πόρου, είναι μικρότερο ή ίσο με τη διαθέσιμη ποσότητα.

Κάποιοι τρόποι εκτέλεσης μπορούν να είναι μη εκτελέσιμοι, χωρίς αυτό να υπονοεί ότι ο διαχειριστής έργου εισήγαγε έναν τρόπο που δεν μπορεί να εκτελεστεί, το οποίο είναι ένα εύκολα διαχειρίσιμο λάθος, αλλά ότι οι απαιτήσεις που τίθενται από έναν τρόπο όταν συνδυάζονται με τις υπόλοιπες δραστηριότητες και τις δεδομένες συγκεκριμένες διαθεσιμότητες πόρων δεν μπορούν να ικονοποιηθούν από κανένα συνδυασμό τρόπων όλων των υπολοίπων δραστηριοτήτων. Ένας τρόπος ορίζεται ως μη εκτελέσιμος αν οι απαιτήσεις του σε έναν ή περισσότερους τύπους πόρων, όταν προστεθούν στις ελάχιστες απαιτήσεις ανα τύπο πόρων όλων των υπολοίπων δραστηριοτήτων, υπερβαίνουν τη διαθεσιμότητα για τουλάχιστον μία χρονική περίοδο. Ένας τρόπος εκτός από το να μην είναι δυνατό να εκτελεστεί μπορεί επίσης να είναι μη αποδοτικός (inefficient). Ένας τρόπος καλείται μη αποδοτικός όταν η διάρκεια και οι απαίτηση σε ανανεώσιμους και μη ανανεώσιμους πόρους είναι υψηλότερες από εκείνες όλων των άλλων τρόπων εκτέλεσης της ίδιας δραστηριότητας. Όλοι οι μη αποδοτικοί τρόποι εκτέλεσης αφαιρούνται από την διαδικασία προγραμματισμού καθώς δεν μπορούν να οδηγήσουν σε κάποια καλή λύση. Τέλος υπολογίζονται οι συντελεστές $ARF$ και $RS_{tot}$ για κάθε έργο όπως αυτοί ορίζονται στο κεφάλαιο 5. Η διαδικασία προεπεξεργασίας που ακολουθείται απεικονίζεται στο Σχήμα 6.3


**Σχήμα 6.2 Διαδικασία προεπεξεργασίας**

### *6.2.3 Ομαδοποίηση & Κατάταξη Έργων & Δραστηριοτήτων*

Την διαδικασία της προεπεξεργασίας ακολουθεί η διαδικασία κατηγοριοποίησης των έργων σε σύνολα σύμφωνα με τις απαιτήσεις τους σε πόρους και τις εξαρτήσεις μεταξύ τους. Οι εξαρτήσεις μεταξύ των έργων διακρίνονται σε δύο τύπους. Σε εσωτερικές εξαρτήσεις μεταξύ δραστηριοτήτων του ίδιου υποέργου και σε εξωτερικές εξαρτήσεις μεταξυ δραστηριοτήτων διαφορετικών έργων. Οι εξαρτήσεις αυτές μπορούν να έρχονται από ένα ή περισσότερα υποέργα που αποτελούν την αρχή της εξάρτησης, προς ένα η περισσότερα έργα που αποτελούν το στόχο της εξάρτησης. Ένα έργο μπορεί να είναι ταυτόχρονα αρχή και στόχος διαφορετικών εξαρτήσεων. Σαν πρώτο βήμα αυτού του τμήματος της επίλυσης δημιουργούνται δύο σύνολα έργων. Ένα σύνολο στο οποίο τοποθετούνται όλα τα ανεξάρτητα έργα και ένα δεύτερο σύνολο όπου τοποθετούνται τα έργα με εξωτερικές εξαρτήσεις. Στη συνέχεια το σύνολο των έργων με εξωτερικές εξαρτήσεις διαιρείται με βάση το πλήθος των εξωτερικών εξαρτήσεων με σκοπό τη δημιουργία δύο ή τριών υποσυνόλων (ανάλογα με το μέγεθος του αρχικού συνόλου): μικρής, μεσαίας και υψηλής εξάρτησης.

Έπειτα πραγματοποιείται κατηγοριοποίηση των έργων εντός των δημιουργημένων συνόλων και υποσυνόλων από τον Moderator ACO αλγόριθμο με βάση το συνολικό βαθμό δυναμικότητας πόρων $RS_{tot_p}$ κάθε έργου, που εκφράζει τη δυνατότητα κάλυψης των αναγκών του έργου σε ανανεώσιμους και μη ανανεώσιμους πόρους από την δεδομένη διαθεσιμότητα ανανεώσιμων και μη ανανεώσιμων πόρων αντίστοιχα. Όταν οι διαφορές των συνολικών βαθμών δυναμικότητας δύο ή περισσότερων έργων δεν είναι αρκετά μεγάλες ώστε να είναι ξεκάθαρη η κατάταξή τους η κατάσταση καλείται «ισοπαλία» και λύνεται με σύγκριση του συνολικού πραγματικού παράγοντα πόρων $ARF_{tot}$ κάθε έργου, ο οποίος δηλώνει τη μέση ποσότητα ανανεώσιμων και μη ανανεώσιμων πόρων που χρησιμοποιούνται και καταναλώνονται αντίστοιχα από κάθε έργο.

Τα έργα έπειτα κατηγοριοποιούνται και προγραμματίζονται με βάση το σύνολο στο οποίο ανήκουν και την σειρά τους εντός του συνόλου αυτού. Έτσι όλα τα έργα που ανήκουν στο σύνολο υψηλής εξάρτησης και έχουν υψηλές απαιτήσεις σε πόρους κατηγοριοποιούνται και προγραμματίζονται πρώτα σε σειρά, έπειτα εκείνα που έχουν επίσης υψηλό βαθμό εξωτερικών εξαρτήσεων αλλά χαμηλές απαιτησεις σε πόρους και ούτω καθεξής. Στη συνέχεια ο Moderator ACO χειρίζεται τη σειρά προγραμματισμού των έργων και τη βελτιστοποίηση των τρόπων εκτέλεσης των δραστηριοτήτων, ενώ ο Activity List ACO καλείται για την βελτιστοποίηση της σειράς των δραστηριοτήτων των έργων.

### *6.2.4 Σειριακή Μέθοδος Παραγωγής Χρονοπρογραμμάτων*

Η πλειονότητα των ευρετικών μεθόδων για προβλήματα προγραμματισμού έργων βασίζονται στη μέθοδο παραγωγής χρονοπρογραμμάτων (schedule generation scheme - SGS) η οποία είναι ένας αλγόριθμος ο οποίος προγραμματίζει μια δραστηριότητα σε κάθε του βήμα μέχρι να παραχθεί ένα πλήρες πρόγραμμα. Στη διαδικασία αυτή, βρίσκεται για κάθε βήμα το σύνολο των δραστηριοτήτων των έργων που είναι επιλέξιμες (eligible) προς προγραμματισμό και υπολογίζεται για την δραστηριότητα που επιλέγεται ένας χρόνος έναρξης, με τέτοιο τρόπο ώστε όλοι οι περιορισμοί σε πόρους και προτεραιότητες να εκπληρώνονται. Η επιλογή της δραστηριότητας σαν διαδικασία δεν είναι κομμάτι του SGS και μπορεί να γίνει για παράδειγμα με βάση έναν κανόνα προτεραιότητας ή ένα διάνυσμα αναπαράστασης ενός μεταευρετικού αλγορίθμου.

Δύο είναι τύποι μεθόδων παραγωγής χρονοπρογραμμάτων που διατίθενται για το κλασικό RCPSP, η παράλληλη μέθοδος παραγωγής χρονοπρογραμμάτων (parallel SGS) και η σειριακή μέθοδος παραγωγής χρονοπρογραμμάτων (serial SGS) (Kolisch and Hartmann, 1999). Εδώ γίνεται εφαρμογή της σειριακής μεθόδου παραγωγής χρονοπρογραμμάτων (SSGS). Αυτή βασίζεται στην προσαύξηση δραστηριοτήτων, κατασκευάζει ενεργά προγράμματα, όπου καμιά δραστηριότητα δεν μπορεί να μετακινηθεί αριστερά χωρίς να καθυστερήσει κάποια άλλη δραστηριότητα και ο χώρος αναζήτησης περιέχει πάντοτε τη βέλτιστη λύση.

Αυτή η μέθοδος αποτελείται από τόσα βήματα όσα και οι δραστηριότητες προς προγραμματισμό. Οι δραστηριότητες χωρίζονται σε δύο σύνολα. Το ένα αποτελείται από τις δραστηριότητες που έχουν ήδη προγραμματιστεί και αποτελούν ένα μερικό πρόγραμμα (partial schedule) και το άλλο από εκείνες που δεν ανήκουν στο μερικό πρόγραμμα, δηλαδή δεν έχουν ακόμα προγραμματιστεί. Σε κάθε βήμα ο SSGS διαλέγει μια επιλέξιμη δραστηριότητα, της οποίας οι προκάτοχοι (predecessors) έχουν ήδη προγραμματιστεί και την τοποθετεί στην νωρίτερη δυνατή θέση του προγράμματος από άποψη διαθεσιμότητας πόρων. Αφού επιλεγεί, η δραστηριότητα περνά στο σύνολο των δραστηριοτήτων που αποτελούν το μερικό πρόγραμμα. Ο αλγόριθμος σταματά όταν έχουν προγραμματιστεί πλέον όλες οι δραστηριότητες. Ο χρόνος νωρίτερης έναρξης μιας δραστηριότητας ορίζεται ως ο χρόνος ολοκλήρωσης της δραστηριότητας που ανήκει στο σύνολο με τις προαπαιτούμενες δραστηριότητές της και έχει τον μεγαλύτερο χρόνο ολοκλήρωσης. Ο χρόνος έναρξής της ορίζεται ως ο μικρότερος χρόνος νωρίτερης έναρξης όπου δεν υπάρχει παράβαση του περιορισμού της διαθεσιμότητας των πόρων σε εκείνη την περίοδο.

Η συμπεριφορά του SSGS αλγορίθμου μπορεί να διαφοροποιηθεί όταν αλλάξουν οι υποθέσεις του κλασικού RCPSP σχετικά με τους τρόπους εκτέλεσης. Όταν έχουμε πολλαπλούς τρόπους εκτέλεσης η διαδικασία επιλογής των τρόπων συνήθως γίνεται ξεχωριστά. Αφού καθοριστούν οι τρόποι εκτέλεσης των δραστηριοτήτων μπορεί να χρησιμοποιηθεί ο κλασικός SGS.

### *6.2.5 Δομή διανύσματος αναπαράστασης έργου*

Ο Moderator ACO χρησιμοποιείται για την κατηγοριοποίηση των έργων σε σύνολα εξαρτήσεων. Αυτό επιτυγχάνεται προσθέτοντας ένα νέο στοιχείο στο κάθε διάνυσμα που αναπαριστά τη λίστα δραστηριοτήτων του κάθε έργου. Το Στοιχείο Υποσυνόλου (Subset Cell) χρησιμοποιείται για τον προσδιορισμό του υποσυνόλου στο οποίο ανήκει το έργο. Επιπρόσθετα, έχοντας πολλαπλούς τρόπους εκτέλεσης ανά δραστηριότητα ανά έργο, απαιτείται μια ακόμη λίστα (ModeList) που περιέχει τους αντίστοιχους τρόπους για κάθε δραστηριότητα κάθε έργου.

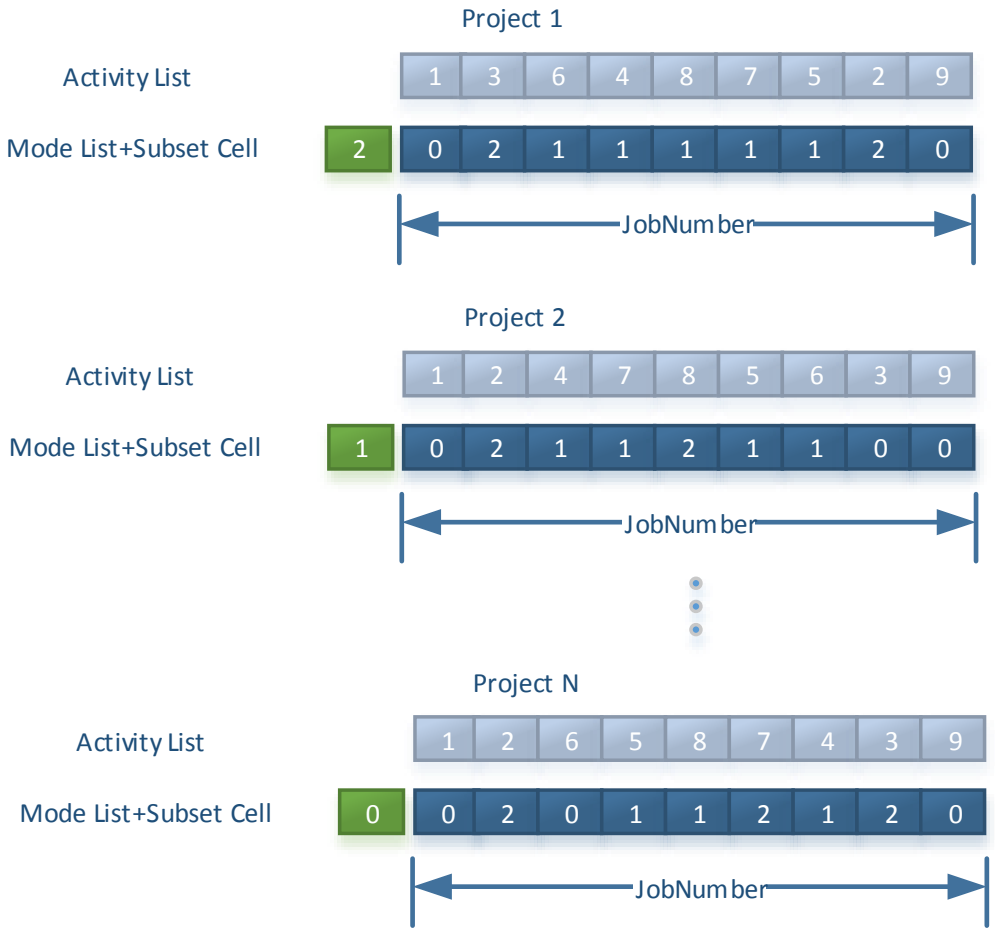
Το διάνυσμα του κάθε έργου παρουσιάζεται στο Σχήμα 6.2 και είναι μια σύνθετη τάξη που συγκροτείται από:

- Το *Στοιχείο Υποσυνόλου*, που ορίζει σε ποιό υποσύνολο ανήκει το έργο.
- Τη *Λίστα Δραστηριοτήτων*, που αποτελείται από τα αναγνωριστικά (IDs) των δραστηριοτήτων του έργου, όπου στο συνολικό αριθμό των δραστηριοτήτων του έργου συμπεριλαμβάνονται οι βοηθητικές δραστηριότητες αρχής και τέλους.
- Τη *Λίστα Τρόπων Εκτέλεσης*, που περιέχει τον επιλεγμένο τρόπο εκτέλεσης κάθε δραστηριότητας.

Όταν φτάσει η σειρά ενός έργου να προγραμματιστεί, σύμφωνα με το *Στοιχείο Υποσυνόλου* του και τον συνολικό βαθμό δυναμικότητας πόρων του $RS_{tot_p}$, για την απόκτηση του προγράμματος που αντιστοιχεί στο διάνυσμά του χρησιμοποιείται ο αλγόριθμος SSGS, όπως περιγράφηκε στην παράγραφο 6.2.4. Στον SSGS αλγόριθμο, αντί για κάποιο κανόνα προτεραιότητας για την εκλογή του ποιά επιλέξιμη δραστηριότητα πρέπει να προγραμματιστεί στη συνέχεια, οι δραστηριότητες λαμβάνονται με τη σειρά που δίνονται στην *Λιστα Δραστηριοτήτων*.

Η *Λίστα Δραστηριοτήτων* είναι μια εφικτή από άποψη προτεραιότητας δραστηριοτήτων μετατεθιμένη λίστα των δραστηριοτήτων του έργου, με την έννοια ότι κάθε δραστηριότητα τοποθετείται μετά από όλους τους άμεσους προκατόχους της (predecessors). Στην προτεινόμενη διαδικασία επίλυσης, η *Λιστα Δραστηριοτήτων* δίνεται από τον Activity List ACO, αφού τοποθετηθούν οι βοηθητικές δραστηριότητες αρχής και τέλους στην πρώτη και την τελευταία θέση αντίστοιχα του διανύσματος.

Η *Λίστα Τρόπων Εκτέλεσης* ορίζει για κάθε δραστηριότητα του έργου ποιός τρόπος εκτέλεσης θα χρησιμοποιηθεί, επομένως καθορίζει την διάρκεια και τις απαιτήσεις σε ανανεώσιμους και μη ανανεώσιμους πόρους της αντίστοιχης δραστηριότητας. Η *Λίστα Τρόπων* είναι ένα διάνυσμα από τρόπους. Η θέση ενός τρόπου στη λίστα αναπαριστά το αναγνωριστικό (ID) της δραστηριότητας με την οποία σχετίζεται. Κάθε τρόπος στη *Λίστα Τρόπων* προσδιορίζει τον τρόπο εκτέλεσης της δραστηριότητας που είναι τοποθετημένη στην αντίστοιχη θέση στη *Λίστα Δραστηριοτήτων*.




**Σχήμα 6.3 Διάνυσμα ACO βελτιστοποίησης**

### *6.2.6 Αρχικός Πλυθησμός*

Στον αρχικό πληθυσμό τα ειδικά *Στοιχεία Υποσυνόλου* κάθε διανύσματος κάθε έργου δίνονται από την αρχική κατηγοριοποίηση της διαδικασίας προεπεξεργασίας, χωρίς να εφαρμοστεί η διαδικασία βελτιστοποίησης του Moderator, έτσι αρχικά κάθε έργο θα προγραμματιστεί σύμφωνα με τη σειρά που ορίστηκε στην παράγραφο

6.2.3, χωρίς να υπάρχει η πιθανότητα να μετακινηθεί από τον Moderator ACO σε άλλο υποσύνολο, στα πλαίσια της βελτιστοποίησης που εφαρμόζει ο Moderator. Οι λίστες δραστηριοτήτων και τρόπων παράγονται με τυχαίο τρόπο. Στην περίπτωση της *Λίστας Δραστηριοτήτων*, στις θέσεις 0 και JobNumber-1 τοποθετούνται οι δραστηριότητες με ID αριθμούς 1 και JobNumber και όλες οι υπόλοιπες θέσεις γεμίζουν από τον Activity List ACO. Κάθε λίστα που παράγεται είναι μια *Λίστα Δραστηριοτήτων* και χρησιμοποιείται για να σχηματιστεί ένα διάνυσμα.

Η *Λίστα Τρόπων* σχηματίζεται παρόμοια. Για κάθε δραστηριότητα, ένας τρόπος επιλέγεται τυχαία από τον Moderator ACO και όλοι μαζί σχηματίζουν τη λίστα από τρόπους.

### 6.2.7 *Λειτουργία Αποικίας Μυρηγκιών (Ant Colony) – Μηχανισμοί Επιλογής, Αξιολόγησης & Εξέλιξης*

Οι δύο ACO αλγόριθμοι που υλοποιήθηκαν για την επίλυση του προβλήματος λειτουργούν με ίδια φιλοσοφία και κανόνες, αναλαμβάνοντας όμως διαφορετικούς ρόλους ο καθένας. Ο Moderator ACO αναλαμβάνει την κατηγοριοποίηση και οργάνωση των έργων, ορίζοντας τη σειρά με την οποία προγραμματίζονται και την βελτιστοποίηση των τρόπων εκτέλεσης των δραστηριοτήτων των έργων. Ο Activity List ACO αναλαμβάνει την βελτιστοποίηση της διάρκειας κάθε λίστας δραστηριοτήτων, δηλαδή κάθε έργου. Παράλληλα, δουλεύουν και οι δύο μαζί για την εύρεση την βέλτιστης διάρκειας του συνόλου των έργων που πρέπει να προγραμματιστούν. Από εκεί και πέρα δεν υπάρχουν διαφοροποιήσεις μεταξύ τους και δουλεύουν και οι δύο με τους ίδιους μηχανισμούς, οι οποίοι είναι οι εξής:

#### *6.2.7.1 Εξέλιξη Μοντέλου Φερομόνης : Αρχικοποίηση, Ανανέωση και Εξάτμιση*

Το στοιχείο κλειδί σε κάθε ACO αλγόριθμο είναι το μοντέλο φερομόνης του, δηλαδή ο πίνακας στον οποίο αποθηκεύονται οι τιμές της φερομόνης για κάθε πιθανή θέση κάθε τμήματος λύσης εντός της επίλυσης. Το μοντέλο φερομόνης του Moderator ACO αναπαριστά την αξία ανάθεσης του τρόπου $j$ στην $i$ δραστηριότητα και την αξία προγραμματισμού του έργου $j$ σαν το $i$ κατά σειρά προτεραιότητας σε σχέση με τα υπόλοιπα έργα. Αντίστοιχα το μοντέλο φερομόνης του Activity List ACO αναπαριστά την αξία τοποθέτησης της δραστηριότητας $j$ στην θέση $i$ της λίστας δραστηριοτήτων.

Το μοντέλο φερομόνης αρχικοποιείται με τρόπο που να δίνει αρχικά ίσες πιθανότητες $\tau_{ij}$ σε όλα τα τμήματα λύσης για τα διανύσματα Activity List και Mode

List. Στο Subset Gene όμως είναι σημαντικό να επιλεγεί το κάθε έργο να προγραμματιστεί με τη σειρά που δίνεται από την αρχική κατηγοριοποίηση με βάση τα σύνολα εξάρτησης και τις απαιτήσεις σε πόρους του κάθε έργου. Έτσι δίνονται μεγάλες αρχικές τιμές στις φερομόνες που αντιπροσωπεύουν αυτή την κατάταξη, αφήνοντας παράλληλα και ένα μικρό περιθώριο για αλλαγές κατά την μετέπειτα εξέλιξη του αλγορίθμου.

Σε κάθε επανάληψη της διαδικασίας τα μοντέλα φερομόνης ανανεώνονται και εξελίσσονται μέσω της σύγκρισης κάθε παραγόμενου προγράμματος-επίλυσης με την βέλτιστη-προς-το-παρόν λύση $s_{bs}$ (BS update rule) αλλά εάν η βέλτιστη-προς-το-παρόν λύση παραμείνει ίδια για πάνω από 10 γενιές-επαναλήψεις, αυτή αντικαθίσταται από την βέλτιστη λύση της τρέχουσας επανάληψης $s_{ib}$ (IB update rule). Η ανανέωση της φερομόνης είναι συνάρτηση του συντελεστή $\rho$ που όπως έχει αναφερθεί καλείται και ρυθμός εξάτμισης (evaporation rate) και της καταλληλότητας $f(s)$ του προγράμματος $T^*$ της $s_{bs}$ ή της $s_{ib}$ ανάλογα με τον κανόνα ανανέωσης που χρησιμοποιείται εκείνη τη στιγμή. Μετά την ανανέωση της φερομόνης εφαρμόζεται η εξάτμισή της, η οποία γίνεται σε όλα τα στοιχεία και των δύο μοντέλων φερομόνης καθολικά με κοινό τρόπο. Η εξάτμιση της φερομόνης είναι συνάρτηση του συντελεστή$\rho$.

#### *6.2.7.2 Μηχανισμός Πιθανοτικής Κατασκευής Λύσης: Αξιολόγηση και Επιλογή*

Κάθε μυρμήγκι κατασκευάζει μέσω της μεθόδου πιθανοτικής κατασκευής λύσης μια εφικτή λύση, επιλέγοντας σε κάθε βήμα του έναν τρόπο εκτέλεσης δραστηριοτήτων και μια σειρά κατάταξης ενός έργου (για τον Moderator ACO) ή τον προγραμματισμό μιας δραστηριότητας ενός έργου που είναι εφικτή από άποψη προτεραιότητας δραστηριοτήτων και περιορισμών σε πόρους (για τον Activity List ACO). Η επιλογή αυτή γίνεται με πιθανότητα $p_{ij}$ η οποία χρησιμοποιεί τις τιμές $\tau_{ij}$ του αντίστοιχου μοντέλου φερομόνης καθώς και τις τιμές $\eta_{ij}$ από τους ευρετικούς κανόνες nLST, nLFT και nMSL. Η σχετική επιρροή φερομονών και ευρετικών κανόνων στην τιμή της πιθανότητας καθορίζονται από δύο συντελεστές $\alpha$ και $\beta$, οι επιτρέπουν στα πρώτα στάδια εκτέλεσης του αλγορίθμου να έχουν μεγαλύτερη επιρροή οι τιμές από τους ευρετικούς κανόνες, στα μεσαία στάδια η κατάσταση να αντιστρέφεται και τελικά να αναλαμβάνουν σχεδόν εξ' ολοκλήρου οι τιμές του μοντέλου φερομόνης.

Αρχικά ο Moderator ACO επιλέγει τους καλύτερους τρόπους στο διάνυσμα Mode List και τη σειρά προγραμματισμού του έργου. Με αυτά τα δεδομένα καλείται ο Activity List ACO να δουλέψει επί του διανύσματος Activity List. Μετά από έναν

προκαθορισμένο αριθμό βημάτων ο Activity List ACO επιστρέφει στον Moderator ACO το βελτιωμένο διάνυσμα δραστηριοτήτων κάθε έργου και τελικά στου συνολικού προγράμματος.

# 7. Αποτελέσματα & Αξιολόγηση

## 7.1 Υλοποίηση & Εφαρμογή

Το προτεινόμενο σύστημα, αποτελούμενο από το μοντέλο, τους αλγορίθμους Moderator ACO και Activity List ACO, τον SSGS καθώς και των διαδικασιών προεπεξεργασίας όπως αυτές περιγράφηκαν στο κεφάλαιο 6, υλοποιήθηκε χρησιμοποιώντας την $C\#.NET$ γλώσσα προγραμματισμού. Η $C\#$ είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που στοχεύει στο συνδυασμό της υπολογιστικής δύναμης της $C++$ και της προγραμματιστικής ευκολίας της Visual Basic.

Ο κώδικας που δημιουργήθηκε παρέχει μια απλή εφαρμογή κονσόλας (console application) που χρησιμοποιήθηκε για τα πειράματα που περιγράφονται στη συνέχεια.

## 7.2 Σχεδιασμός Πειραμάτων

Για την πειραματική επιαλήθευση της αποτελεσματικότητας και της αποδοτικότητας του υλοποιημένου συστήματος, χρησιμοποιήθηκαν τρία διαφορετικά σύνολα περιπτώσεων προβλημάτων, ένα για κάθε παραλλαγή του RCPSP που αντιμετωπίζει το μοντέλο. Τα δύο πρώτα σύνολα προβλημάτων, για το RCPSP και το MRCPSP λήφθηκαν από την διαδικτυακή βιβλιοθήκη προβλημάτων προγραμματισμού έργων PSPLIB, ενώ το τρίτο σύνολο προβλημάτων, δηλαδή των προβλημάτων για το MultiProject MRCPSP δημιουργήθηκε από το σύνολο MRCPSP, συνενώνοντας για κάθε ένα συνολικό πρόβλημα έναν αριθμό από διαφορετικά προβλήματα της MRCPSP βιβλιοθήκης, εισάγοντας με τυχαίο τρόπο συσχετίσεις μεταξύ των έργων και ορίζοντας ως συνολικές διαθεσιμότητες πόρων τα αθροίσματα των διαθεσιμοτήτων των πόρων των επιμέρους έργων που αποτελούν κάθε φορά το υπερ-έργο.

Για το RCPSP χρησιμοποιήθηκε το j30 σύνολο δεδομένων (30 δραστηριότητες προς προγραμματισμό ανά έργο με 4 διαφορετικούς ανανεώσιμους πόρους) ενώ για το MRCPSP χρησιμοποιήθηκαν τα σύνολα j10, j12 και c15.

Όσον αφορά τις αρχικές τιμές των παραμέτρων $\alpha, \beta, \rho$ και $c$ των ACO αλγορίθμων, αυτές παίρνουν τις τιμές $1, 2, 0{,}075$ και $0{,}5$ αντίστοιχα. Οι τιμές των παραμέτρων $\alpha$ και $c$ παραμένουν σταθερές κατά τη διάρκεια εκτέλεσης του αλγορίθμου, όμως οι τιμές του $\rho$ και του $\beta$ μεταβάλλονται,. Στο $\beta$ επιβάλλεται γραμμική μείωση εως τον μηδενισμό του στη μέση της εκτέλεσης του αλγορίθμου με

στόχο να είναι μεν αρχικά ισχυρή η επίδραση της ευρετικής πληροφορίας στην επιλογή των μυρμηγκιών, βοηθώντας στην δημιουργία σταθερών βάσεων για τα μοντέλα φερομόνης, να δίνει δε έπειτα την θέση του στα μοντέλα φερομόνης, αφήνοντας σε αυτά την επίδραση στην επιλογή των μυρμηγκιών. Το $\rho$ κρατά σταθερή τιμή μέχρι τα τελευταία βήματα του αλγορίθμου, οπότε και του δίνεται η τιμή $0{,}075$ με στόχο αρχικά ο αλγόριθμος να μπορεί να ψάξει σε μεγάλο εύρος το χώρο λύσεων, αλλά στα τελευταία στάδιά του να επικεντρωθεί στον έλεγχο της περιοχής γύρω από τις καλύτερες λύσεις που έχουν βρεθεί.

Τέλος, ο συντελεστής βάρους $\alpha$ της ποινής που επιβάλλεται στην αντικειμενική συνάρτηση του προβλήματος, όπως αυτή ορίστηκε στο κεφάλαιο 5 παίρνει ως αρχική τιμή τη μονάδα, οδηγώντας έτσι τον αλγόριθμο στην αναζήτηση εφικτών λύσεων, αλλά μειώνεται σταδιακά φτάνοντας στο $0{,}5$ στη μέση του χρόνου εκτέλεσης του αλγορίθμου και μένοντας σταθερό ως το τέλος της εκτέλεσης, διατηρώντας έτσι ένα βάρος στις λύσεις που υπερβαίνουν τις διαθεσιμότητες σε μη ανανεώσιμους πόρους, αλλά σε χαμηλότερο βαθμό, επιτρέποντας στον αλγόριθμο να αναζητήσει και λύσεις μη εφικτές, από τις οποίες πιθανά να προκύψουν καλύτερης ποιότητας τελικές λύσεις.

## 7.3 Πειραματικά Αποτελέσματα

Στους πίνακες που ακολουθούν απεικονίζεται ένα μέρος των αποτελεσμάτων του πειράματος. Τα αποτελέσματα του συστήματος για το RCPSP είναι η ελάχιστη και η μέγιστη διάρκεια των λύσεων που βρέθηκαν, η απόκλισή τους από τα βέλτιστα αποτελέσματα για κάθε πρόβλημα, όπως αυτά δίνονται τη στιγμή εκπόνησης της εργασίας στην PSPLIB, και η συχνότητα με την οποία το σύστημα βρίσκει τη βέλτιστη λύση. Για το MRCPSP, εκτός από τα παραπάνω, παρουσιάζεται η μέγιστη και η ελάχιστη τιμή ποινής των παραγόμενων από τον αλγόριθμο λύσεων, η μέση τιμή της καθώς και η συχνότητα με την οποία ο αλγόριθμος παρήγαγε λύσεις των οποίων η ποινή ήταν μηδενική. Για το Multi Project MRCPSP παρουσιάζονται η μέγιστη και η ελάχιστη διάρκεια για το συνολικό υπερέργο και τα επιμέρους έργα που το αποτελούν, η μέση απόκλιση των διαρκειών των επιμέρους έργων από τα βέλτιστα αποτελέσματα όπως αυτά δίνονται από την PSPLIB και η τυπική απόκλιση της διάρκειας των παραγόμενων προγραμμάτων για κάθε υπερ-έργο.

Για τα αποτελέσματα του Milti Project MRCPSP δεν υπάρχει προηγούμενο και ως αποτέλεσμα δεν μπορεί να πραγματοποιηθεί συγκριτική ανάλυση των αποτελεσμάτων που προκύπτουν. Μπορεί όμως να εξαχθεί το πιθανά ενδιαφέρον συμπέρασμα οτι λόγω του ορισμού της διαθεσιμότητας πορων του υπερ-έργου ως

άθροισης των πόρων των επιμέρους έργων, παρατηρείται η παραγωγή από τον αλγόριθμο προγραμμάτων για τα επιμέρους έργα, όπου από τη μια έχει διατηρηθεί ο περιορισμός πόρων για το κάθε έργο σε σχέση με τις αρχικές του διαθεσιμότητες, από την άλλη όμως ποινές που πιθανά θα προέκυπταν, απορροφώνται στο συνολικό προγραμμα από τις διαθεσιμότητες των υπολοίπων έργων, δίνοντας έτσι τη δυνατότητα παραγωγής προγραμμάτων αρκετά μικρότερης διάρκειας από τα δεδομένα βέλτιστα της PSPLIB για κάθε επιμέρους έργο.

**Πίνακας 7.1 Αποτελέσματα RCPSP για το σύνολο προβλημάτων *j30***

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Μέση απόκλιση από το βέλτιστο | Συχνότητα εύρεσης βελτίστου |
|---|---|---|---|---|---|
| **J301_1.RCP** | 43 | 43 | 43 | 0,0% | 100% |
| **J301_2.RCP** | 47 | 47 | 47 | 0,0% | 100% |
| **J301_3.RCP** | 47 | 47 | 47 | 0,0% | 100% |
| **J301_4.RCP** | 62 | 63 | 62 | 0,82% | 40% |
| **J301_5.RCP** | 41 | 41 | 39 | 2,11% | 0% |
| **J301_6.RCP** | 49 | 49 | 48 | 1,06% | 0% |
| **J301_9.RCP** | 50 | 52 | 49 | 2,53% | 0% |
| **J301_10.RCP** | 45 | 45 | 45 | 0,0% | 100% |
| **J302_2.RCP** | 51 | 53 | 51 | 1,89% | 20% |
| **J302_7.RCP** | 47 | 47 | 47 | 0,0% | 100% |
| **J302_8.RCP** | 54 | 54 | 54 | 0,0% | 100% |
| **J302_9.RCP** | 54 | 54 | 54 | 0,0% | 100% |
| **J302_10.RCP** | 43 | 43 | 43 | 0,0% | 100% |
| **J303_6.RCP** | 54 | 54 | 54 | 0,0% | 100% |
| **J303_7.RCP** | 48 | 48 | 48 | 0,0% | 100% |
| **J303_8.RCP** | 54 | 54 | 54 | 0,0% | 100% |
| **J303_10.RCP** | 59 | 64 | 59 | 2,94% | 40% |
| **J304_3.RCP** | 47 | 47 | 47 | 0,0% | 100% |
| **J304_8.RCP** | 55 | 55 | 55 | 0,0% | 100% |
| **J304_10.RCP** | 48 | 48 | 48 | 0,0% | 100% |

**Πίνακας 7.2 Αποτελέσματα MRCPSP για το σύνολο προβλημάτων j10**

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Μέση απόκλιση από το βέλτιστο | Συχνότητα εύρεσης βελτίστου | Ελάχιστη ποινή | Μέγιστη Ποινή | Μέση ποινη | Συχνότητα λύσεων χωρις ποινη |
|---|---|---|---|---|---|---|---|---|---|
| **j1010_1.mm** | 17 | 17 | 17 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1010_2.mm** | 22 | 24 | 24 | -1,67% | 100% | 0 | 1 | 0,7 | 30% |
| **j1010_3.mm** | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1010_4.mm** | 15 | 15 | 15 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1010_7.mm** | 13 | 14 | 15 | -2,73% | 100% | 0 | 1 | 0,9 | 10% |
| **j1010_8.mm** | 12 | 15 | 15 | 0,33% | 100% | 0 | 1 | 0,8 | 20% |
| **j1011_1.mm** | 20 | 20 | 20 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1011_2.mm** | 11 | 11 | 11 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1011_4.mm** | 23 | 23 | 23 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1011_5.mm** | 15 | 16 | 15 | 0,47% | 80% | 0 | 0 | 0 | 100% |
| **j1011_6.mm** | 15 | 15 | 15 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1011_7.mm** | 11 | 13 | 11 | 0,67% | 90% | 0 | 0 | 0 | 100% |
| **j1011_8.mm** | 14 | 14 | 14 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1034_2.mm** | 12 | 12 | 12 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1034_3.mm** | 25 | 25 | 25 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1034_4.mm** | 21 | 23 | 23 | -1,63% | 100% | 1 | 1 | 0,8 | 20% |
| **j1034_5.mm** | 22 | 22 | 22 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1034_8.mm** | 19 | 20 | 19 | 0,67% | 60% | 0 | 1 | 0,1 | 90% |
| **j1034_9.mm** | 26 | 27 | 26 | 1,03% | 10% | 0 | 0 | 0 | 100% |
| **j1034_10.mm** | 14 | 15 | 18 | -4,03% | 100% | 1 | 1 | 1 | 0% |

**Πίνακας 7.3 Αποτελέσματα MRCPSP για το σύνολο προβλημάτων c15**

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Μέση απόκλιση από το βέλτιστο | Συχνότητα εύρεσης βέλτιστου | Ελάχιστη ποινή | Μέγιστη Ποινή | Μέση ποινη | Συχνότητα λύσεων χωρις ποινη |
|---|---|---|---|---|---|---|---|---|---|
| **c1510_1.mm** | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1510_2.mm** | 16 | 17 | 17 | 0,82% | 80% | 0 | 0 | 0 | 100% |
| **c1510_3.mm** | 23 | 23 | 23 | 0,33% | 90% | 0 | 0 | 0 | 100% |
| **c1510_5.mm** | 13 | 15 | 13 | 0,67% | 90% | 0 | 0 | 0 | 100% |
| **c1510_6.mm** | 32 | 32 | 32 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1510_7.mm** | 15 | 16 | 15 | 0,30% | 90% | 0 | 1 | 0,1 | 90% |
| **c1510_8.mm** | 16 | 18 | 16 | 1,53% | 60% | 0 | 3 | 0,6 | 80% |
| **c1511_3.mm** | 30 | 31 | 30 | 0,33% | 100% | 0 | 0 | 0 | 100% |
| **c1511_4.mm** | 14 | 15 | 14 | 0,33% | 90% | 0 | 0 | 0 | 100% |
| **c1511_5.mm** | 11 | 12 | 11 | 0,82% | 80% | 0 | 0 | 0 | 100% |
| **c1511_6.mm** | 16 | 18 | 18 | -1,20% | 100% | 0 | 2 | 0,4 | 80% |
| **c1511_7.mm** | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1511_10.mm** | 30 | 30 | 30 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1542_1.mm** | 25 | 27 | 27 | -2,81% | 100% | 0 | 2 | 1,4 | 30% |
| **c1542_2.mm** | 20 | 23 | 22 | 0,94% | 60% | 0 | 3 | 0,5 | 70% |
| **c1542_3.mm** | 25 | 28 | 28 | -2,05% | 100% | 0 | 4 | 1,9 | 30% |
| **c1542_4.mm** | 23 | 24 | 24 | -0,58% | 100% | 0 | 2 | 1,1 | 20% |
| **c1542_5.mm** | 17 | 22 | 22 | -2,83% | 100% | 0 | 5 | 1,7 | 50% |
| **c1542_9.mm** | 25 | 25 | 25 | 0,00% | 80% | 0 | 0 | 0 | 100% |
| **c1542_10.mm** | 22 | 25 | 25 | -1,45% | 100% | 0 | 3 | 1 | 40% |

## Πίνακας 7.4 Αποτελέσματα Multi Project MRCPSP

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Μέση απόκλιση από το βέλτιστο | Τυπική απόκλιση |
|---|---|---|---|---|---|
| **SP_0_1** | 22 | 24 | - | | 0,36% |
| j1010_1.mm | 17 | 17 | 17 | 0,00% | |
| j1010_2.mm | 22 | 24 | 24 | -2,00% | |
| j1010_3.mm | 21 | 21 | 21 | 0,00% | |
| j1010_4.mm | 15 | 15 | 15 | 0,00% | |
| j1010_5.mm | 22 | 22 | 24 | -2,195 | |
| **SP_1_2** | 87 | 93 | | | 1,52% |
| j1237_6.mm | 36 | 46 | 48 | -11,37% | |
| j1237_7.mm | 36 | 42 | 45 | -7,40% | |
| j1237_8.mm | 28 | 33 | 31 | -2,31% | |
| j1237_9.mm | 24 | 30 | 26 | 2,29% | |
| j1237_10.mm | 23 | 32 | 42 | -17,46% | |
| **SP_2_1** | 43 | 43 | | | 0,00% |
| j1010_1.mm | 17 | 17 | 17 | 0,00% | |
| j1010_2.mm | 22 | 24 | 24 | -2,00% | |
| j1010_3.mm | 21 | 21 | 21 | 0,00% | |
| j1010_4.mm | 15 | 15 | 15 | 0,00% | |
| j1010_5.mm | 22 | 22 | 24 | -2,12% | |
| **SP_2_2** | 28 | 34 | | | 1,50% |
| j1010_6.mm | 13 | 15 | 18 | -4,23% | |
| j1010_7.mm | 13 | 13 | 15 | -2,12% | |
| j1010_8.mm | 11 | 15 | 15 | -2,49% | |
| j1010_9.mm | 10 | 10 | 10 | 0,00% | |
| j1010_10.mm | 14 | 18 | 17 | -1,92% | |
| **SP_3_1** | 43 | 43 | | | 0,00% |
| j1010_1.mm | 17 | 17 | 17 | 0,00% | |
| j1010_2.mm | 22 | 24 | 24 | -1,76% | |
| j1010_3.mm | 21 | 22 | 21 | 0,33% | |
| j1010_4.mm | 15 | 16 | 15 | 0,33% | |
| j1010_5.mm | 22 | 22 | 24 | -2,12% | |
| **SP_3_2** | 55 | 61 | | | 1,41% |
| j1010_6.mm | 13 | 15 | 18 | -4,14% | |
| j1010_7.mm | 13 | 13 | 15 | -2,12% | |
| j1010_8.mm | 11 | 15 | 15 | -2,29% | |
| j1010_9.mm | 10 | 10 | 10 | 0,00% | |
| j1010_10.mm | 14 | 17 | 17 | -2,08% | |
| **SP_3_4** | 117 | 126 | | | 1,64% |
| j1237_6.mm | 35 | 37 | 48 | -12,89% | |
| j1237_7.mm | 35 | 39 | 45 | -9,16% | |
| j1237_8.mm | 28 | 31 | 31 | -2,16% | |
| j1237_9.mm | 23 | 28 | 26 | -1,79% | |
| j1237_10.mm | 23 | 24 | 42 | -19,08% | |

# 8. Συμπεράσματα & Κατευθύνσεις Μελλοντικής Έρευνας

Ο προγραμματισμός έργων αφορά την ανάπτυξη ενός προγράμματος βάσης έργου, το οποίο καθορίζει για κάθε δραστηριότητα τους εφικτούς, από άποψη προτεραιοτήτων δραστηριοτήτων και διαθεσιμότητας πόρων, χρόνους έναρξης και λήξης, τις ποσότητες των διαφόρων τύπων πόρων που χρειάζονται ανά χρονική περίοδο και σαν αποτέλεσμα τον αντίστοιχο προϋπολογισμό που απαιτείται για την πραγματοποίηση του έργου.

Τα προβλήματα προγραμματισμού μελετώνται τα τελευταία εξήντα χρόνια με όλο και μεγαλύτερο ενδιαφέρον, το οποίο πηγάζει από την ανάγκη βελτίωσης και διευκόλυνσης της διοίκησης έργων. Ο προγραμματισμός έργων είναι ένα σύνθετο πρόβλημα που κάθε διαχειριστής έργων αντιμετωπίζει στην αρχή κάθε έργου και οι συνέπειες ενός μη σωστά σχεδιασμένου προγράμματος μπορούν να θέσουν σε κίνδυνο την επιτυχή εκτέλεση και ολοκλήρωση του έργου. Ο προγραμματισμός έργων εφαρμόζεται σε ποικίλες βιομηχανίες όπως είναι η κατασκευές, η ανάπτυξη λογισμικού κτλ. Επίσης είναι ιδιαίτερα ελκυστικός για τους ερευνητές και κυρίως εκείνους που σχετίζονται με την επιχειρησιακή έρευνα, επειδή τα προβλήματα και τα μοντέλα τους σε αυτό τον τομέα είναι πολύπλοκα και πλούσια άρα και δύσκολα επιλύσιμα. Στην εργασία αυτή παρουσιάστηκε ένα νέο εννοιολογικό και μαθηματικό μοντέλο το οποίο επεκτείνει το κλασικό RCPSP μέσω της εισαγωγής στο πρόβλημα πολλαπλών παράλληλων έργων, πολλαπλών τρόπων εκτέλεσης των δραστηριοτήτων κάθε έργου, συσχετίσεων μεταξύ των δραστηριοτήτων του ίδιου αλλά και διαφορετικών έργων και χρήσης ενός συνόλου πόρων, που περιλαμβάνει ανανεώσιμους και μη-ανανεώσιμους πόρους. Με βάση αυτό το μοντέλο προτάθηκε ένας σύνθετος εξελικτικός αλγόριθμος βασισμένος στη βελτιστοποίηση με αποικία μυρμηγκιών, ο οποίος χειρίζεται σε πρώτο επίπεδο την κατηγοριοποίηση και τον προγραμματισμό των έργων και την εκχώρηση των τρόπων εκτέλεσης των δραστηριοτήτων και σε δεύτερο επίπεδο τον προγραμματισμό των δραστηριοτήτων κάθε έργου.

Η προτεινόμενη εργασία αποτελείται λοιπόν από δύο τμήματα, πρώτον το μαθηματικό μοντέλο της, που προτείνει μια πιο γενική οπτική του προβλήματος προγραμματισμού έργων και δεύτερον την αλγόριθμο επίλυσης που χρησιμοποιείται για να λύσει το πρόβλημα αυτό αποδοτικά και αποτελεσματικά. Μέχρι σήμερα ο τρόπος αντιμετώπισης του προβλήματος προγραμματισμού πολλαπλών παράλληλων έργων, ενός προβλήματος που αντιμετωπίζεται ιδιαίτερα συχνά από τους επαγγελματίες διαχειριστές έργων, οδηγούσε σε ιδιαίτερα πολύπλοκα και σύνθετα μοντέλα, που ο χρόνος επίλυσής τους ήταν απαγορευτικά μεγάλος ή το

πρόβλημα που αντιμετωπιζόταν περιοριζόταν σε συγκεκριμένες, προσαρμοσμένες σε ειδικές περιπτώσεις συνθήκες, χάνοντας έτσι τη δυνατότητα εφαρμογής της διαδικασίας επίλυσης και σε άλλες περιπτώσεις. Εδώ, σε μια προσπάθεια κάλυψης του ερευνητικού κενού, προτάθηκε ένα μοντέλο με στόχο την δημιουργία μιας δαδικασίας επίλυσης που θα παράγει προγράμματα με δυνατότητα προσαρμογής σε διαφορετικές περιπτώσεις προβλημάτων προγραμματισμού πολλαπλών έργων. Επιπλέον, προτάθηκε μια διαδικασία επίλυσης η οποία είναι σύντομη και γρήγορη αρκετά ώστε να επιτρέπει επανεκτελέσεις της για διάφορα πιθανά σενάρια, δίνοντας τη δυνατότητα στο διαχειριστή έργου να έχει στη διάθεσή του γρήγορα και αποτελεσματικά ένα πρόγραμμα εφικτό και αποτελεσματικό για το σύνολο των προς εκτέλεση έργων.

Το επόμενο βήμα σε αυτή την ερευνητική προσπάθεια είναι η ενίσχυσή της διαδικασίας επίλυσης με την προσθήκη και άλλων ειδών μετα-ευρετικών αλγορίθμων και σαν αποτέλεσμα τη δημιουργία υβριδικών αλγορίθμων επίλυσης, καθώς κάθε είδος εξελικτικού αλγορίθμου έχει διαφορετικά πλεονεκτήματα και μειονεκτήματα και ο συνδυασμός τους πιθανά να δώσει ακόμη καλύτερα αποτελέσματα, πιο σύντομα και αποδοτικά. Επίσης, επιπλέον πειραματισμοί μπορούν να γίνουν για περεταίρω ανάλυση του τρόπου κατηγοριοποίησης των δραστηριοτήτων σε σύνολα προτεραιότητας, όπως είναι ο έλεγχος για το ποιός είναι ο βέλτιστος αριθμός συνόλων στα οποία χωρίζονται τα έργα, η εύρεση νέων κριτηρίων κατηγοριοποίησης των έργων εντός κάθε συνόλου, η εισαγωγή βαρών στα κριτήρια κατηγοριοποίησης των έργων και των συνόλων των έργων και η ενσωμάτωση περισσότερων στόχων προς επίτευξη για το αντιμετωπιζόμενο πρόβλημα.

Τέλος, ιδιαίτερα σημαντικό είναι να γίνουν προσπάθειες συνδυασμού των διαφόρων μεθόδων και διαδικασιών επίλυσης προβλημάτων προγραμματισμού έργων, με τελικό στόχο τη δημιουργία ολιστικών μεθόδων αντιμετώπισης του προβλήματος, που θα μπορούν να αντιμετωπίσουν ρεαλιστικά προβλήματα προγραμματισμού έργων, χωρίς να απαιτούνται εξεζητημένες γνώσεις μαθηματικών ή προγραμματισμού από τους επαγγελματίες διαχειριστές έργων, ούτε η διαφορετική φύση του κάθε πρακτικού προβλήματος να δημιουργεί εμπόδια στην μέθοδο επίλυσης, αλλά μάλλον να ενισχύει την γενικότητά της, μέσα από διαδικασίες ανάδρασης και βελτιστοποίησης. Τέτοιες ενέργειες έχουν ήδη αρχίσει, με εφαρμογή ολιστικών μαθηματικών μοντέλων πολυστοχικού προγραμματισμού και χρήση υβριδικών αλγορίθμων επίλυσης (Rokou et al., 2013). Παρόλα αυτά, υπάρχουν ακόμα πολλά θέματα που πρέπει να αντιμετωπιστούν και πολλές περιπτώσεις αυτού και παρόμοιων προβλημάτων που πρέπει να εξεταστούν, προτού επιτευχθεί σταθερή σύνδεση μεταξύ των μοντέλων και των μεθόδων επίλυσης που παρέχονται

από την επιστημονική κοινότητα και των πρακτικών προβλημάτων, χωρίς συμβιβασμούς, υποθέσεις και απλοποιήσεις.

# 9. Βιβλιογραφία


AL-FAWZAN, M. A. & HAOUARI, M. 2005. A bi-objective model for robust resource-constrained project scheduling. *International Journal of production economics,* 96**,** 175-187.

ALVAREZ-VALDES, R. & TAMARIT, J. M. 1989. Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. *Advances in project scheduling,* 134.

BALLESTÍN, F. & TRAUTMANN, N. 2008. An iterated-local-search heuristic for the resource-constrained weighted earliness-tardiness project scheduling problem. *International Journal of Production Research,* 46**,** 6231-6249.

BARTUSCH, M., MÖHRING, R. H. & RADERMACHER, F. J. 1988. Scheduling project networks with resource constraints and time windows. *Annals of operations Research,* 16**,** 199-240.

BELL, C. E. & PARK, K. 1990. Solving resource-constrained project scheduling problems by a* search. *Naval Research Logistics (NRL),* 37**,** 61-84.

BIANCO, L., CARAMIA, M. & DELL'OLMO, P. 1998. Solving a preemptive scheduling problem using coloring technique. *Project Scheduling Recent Models, Algorithms and Applications, Kluwer Academic Publishers, Dordrecht*.

BLAZEWICZ, J., LENSTRA, J. K. & KAN, A. H. G. R. 1983. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics,* 5**,** 11-24.

BLUM, C. 2005. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews,* 2**,** 353-373.

BLUM, C. & DORIGO, M. 2004. The hyper-cube framework for ant colony optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* 34**,** 1161-1172.

BOCTOR, F. F. 1990. Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research,* 49**,** 3-13.

BOULEIMEN, K. & LECOCQ, H. 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research,* 149**,** 268-281.

BRUCKER, P., DREXL, A., MÖHRING, R., NEUMANN, K. & PESCH, E. 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research,* 112**,** 3-41.

BULLNHEIMER, B., HARTL, R. F. & STRAUSS, C. 1997. A new rank based version of the Ant System. A computational study.

CALHOUN, K. M., DECKRO, R. F., MOORE, J. T., CHRISSIS, J. W. & VAN HOVE, J. C. 2002. Planning and re-planning in project and production scheduling. *Omega,* 30**,** 155-170.

CERVANTES, M., LOVA, A., TORMOS, P. & BARBER, F. 2008. A dynamic population steady-state genetic algorithm for the resource-constrained project scheduling problem. *New Frontiers in Applied Artificial Intelligence.* Springer.

CHEN, V. Y. 1994. a 0–1 goal programming model for scheduling multiple maintenance projects at a copper mine. *European journal of operational research,* 76**,** 176-191.

CHIANG, C.-W., HUANG, Y.-Q. & WANG, W.-Y. 2008. Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. *Journal of Intelligent and Fuzzy Systems,* 19**,** 345-358.

CHRISTOFIDES, N., ALVAREZ-VALDES, R. & TAMARIT, J. M. 1987. Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research,* 29**,** 262-273.

CLARK, W. 1942. *The Gantt Chart, a Working Tool of Management, second edition,* London, Sir Isaac Pitman & Sons, Ltd.
COLORNI, A., DORIGO, M. & MANIEZZO, V. Distributed optimization by ant colonies. Proceedings of the first European conference on artificial life, 1991. Paris, France, 134-142.
COLORNI, A., DORIGO, M., MANIEZZO, V. & TRUBIAN, M. 1994. Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science,* 34**,** 39-53.
DAVENPORT, T. H., HARRIS, J. G. & CANTRELL, S. 2004. Enterprise systems and ongoing process change. *Business Process Management Journal,* 10**,** 16-26.
DAVIS, E. W. 1973. Project Scheduling under Resource Constraints—Historical Review and Categorization of Procedures. *A I I E Transactions,* 5**,** 297-313.
DAVIS, E. W. & PATTERSON, J. H. 1975. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management science,* 21**,** 944-955.
DEBELS, D. & VANHOUCKE, M. 2005. A bi-population based genetic algorithm for the resource-constrained project scheduling problem. *Computational Science and Its Applications–ICCSA 2005.* Springer.
DEMEULEMEESTER, E. & HERROELEN, W. 1992. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management science,* 38**,** 1803-1818.
DEMEULEMEESTER, E., HERROELEN, W., SIMPSON, W. P., BAROUM, S., PATTERSON, J. H. & YANG, K.-K. 1994. On a paper by Christofides et al. for solving the multiple-resource constrained, single project scheduling problem. *European Journal of Operational Research,* 76**,** 218-228.
DORIGO, M. 1992. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*.
DORIGO, M. 2006. *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings*, Springer-Verlag New York Incorporated.
DORIGO, M. & BLUM, C. 2005. Ant colony optimization theory: A survey. *Theoretical computer science,* 344**,** 243-278.
DORIGO, M. & DI CARO, G. Ant colony optimization: a new meta-heuristic. Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, 1999. IEEE.
DORIGO, M. & GAMBARDELLA, L. M. 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on,* 1**,** 53-66.
DORIGO, M., MANIEZZO, V. & COLORNI, A. 1991. The ant system: An autocatalytic optimizing process. Technical report.
DORIGO, M., MANIEZZO, V. & COLORNI, A. 1996. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* 26**,** 29-41.
DREXL, A. & KIMMS, A. 2001. Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society***,** 340-351.
ELMAGHRABY, S. E. 1964. An algebra for the analysis of generalized activity networks. *Management Science,* 10**,** 494-514.
GAREY, M. R. & JOHNSON, D. S. 1979. *Computers and intractability*, Freeman New York.
GLOVER, F. 1989. Tabu search—part I. *ORSA Journal on computing,* 1**,** 190-206.
GLOVER, F. & LAGUNA, M. 1997. *Tabu search*, Springer.
GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K. & KAN, A. H. G. R. 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling:

a Survey. *In:* P.L. HAMMER, E. L. J. & KORTE, B. H. (eds.) *Annals of Discrete Mathematics.* Elsevier.
HARTMANN, S. 1998. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL),* 45**,** 733-750.
HARTMANN, S. 2001. Project scheduling with multiple modes: a genetic algorithm. *Annals of Operations Research,* 102**,** 111-135.
HARTMANN, S. 2002. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL),* 49**,** 433-448.
HARTMANN, S. & BRISKORN, D. 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research,* 207**,** 1-14.
HARTMANN, S. & DREXL, A. 1998. Project scheduling with multiple modes: A comparison of exact algorithms. *Networks,* 32**,** 283-297.
HARTMANN, S. & KOLISCH, R. 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research,* 127**,** 394-407.
HEIMERL, C. & KOLISCH, R. 2010. Scheduling and staffing multiple projects with a multi-skilled workforce. *OR spectrum,* 32**,** 343-368.
HERROELEN, W. 2005. Project scheduling—Theory and practice. *Production and Operations Management,* 14**,** 413-432.
HERROELEN, W., DE REYCK, B. & DEMEULEMEESTER, E. 1998. Resource-constrained project scheduling: a survey of recent developments. *Computers & Operations Research,* 25**,** 279-302.
HERROELEN, W., DEMEULEMEESTER, E. & DE REYCK, B. 1999. *A classification scheme for project scheduling*, Springer.
HERROELEN, W. S. & DEMEULEMEESTER, E. L. 1996. Project management and scheduling. *European Journal of Operational Research,* 90**,** 197-199.
HERROELEN, W. S., VAN DOMMELEN, P. & DEMEULEMEESTER, E. L. 1997. Project network models with discounted cash flows a guided tour through recent developments. *European Journal of Operational Research,* 100**,** 97-121.
ICMELI, O. & ERENGUC, S. S. 1996a. A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows. *Management Science,* 42**,** 1395-1408.
ICMELI, O. & ERENGUC, S. S. 1996b. The resource constrained time/cost tradeoff project scheduling problem with discounted cash flows. *Journal of Operations Management,* 14**,** 255-275.
ISO 2012. 21500 Guidance on project management. *ICS 03.100.40,* ISO Geneva.
JAISWAL, U. & AGGARWAL, S. 2011. Ant Colony Optimization. *International Journal of Scientific & Engineering Research* 2.
KARP, R. 1975. On the computational complexity. *Networks,* 5**,** 45-68.
KELLEY JR, J. E. & WALKER, M. R. Critical-path planning and scheduling. Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference, 1959. ACM, 160-173.
KIM, K. W., GEN, M. & YAMAZAKI, G. 2003. Hybrid genetic algorithm with fuzzy logic for resource-constrained project scheduling. *Applied Soft Computing,* 2**,** 174-188.
KIMMS, A. 2001. Maximizing the net present value of a project under resource constraints using a Lagrangian relaxation based heuristic with tight upper bounds. *Annals of Operations Research,* 102**,** 221-236.
KIRKPATRICK, S., JR., D. G. & VECCHI, M. P. 1983. Optimization by simmulated annealing. *science,* 220**,** 671-680.
KLEIN, R. 2000. Project scheduling with time-varying resource constraints. *International Journal of Production Research,* 38**,** 3937-3952.

KOBYLAŃSKI, P. & KUCHTA, D. 2007. A note on the paper by MA Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics,* 107**,** 496-501.
KOLISCH, R. 1995. *Project scheduling under resource constraints: efficient heuristics for several problem classes*, Springer Verlag.
KOLISCH, R. 1996. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research,* 90**,** 320-333.
KOLISCH, R. 2000. Integrated scheduling, assembly area-and part-assignment for large-scale, make-to-order assemblies. *International Journal of Production Economics,* 64**,** 127-141.
KOLISCH, R. & DREXL, A. 1997. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE transactions,* 29**,** 987-999.
KOLISCH, R. & HARTMANN, S. 1999. *Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis*, Springer.
KURTULUS, I. & DAVIS, E. 1982. Multi-project scheduling: Categorization of heuristic rules performance. *Management Science,* 28**,** 161-172.
KURTULUS, I. S. & NARULA, S. C. 1985. Multi-project scheduling: Analysis of project performance. *IIE transactions,* 17**,** 58-66.
LAND, A. H. & DOIG, A. G. 1960. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society***,** 497-520.
LEWIS, J. P. 1998. *Mastering project management: Applying advanced concepts of systems thinking, control and evaluation, resource allocation*, McGraw-Hill New York.
LIU, X., JIANG, W., XIE, J. & JIA, Y. A New Resource Constrained Project Scheduling Problem. Information Processing, 2009. APCIP 2009. Asia-Pacific Conference on, 18-19 July 2009 2009. 476-480.
LORENZONI, L. L., AHONEN, H. & ALVARENGA, A. G. D. 2006. A multi-mode resource-constrained scheduling problem in the context of port operations. *Computers & Industrial Engineering,* 50**,** 55-65.
LUMSDEN, P. 1968. *The line-of-balance method*, Pergamon Press Limited.
LUO, S., WANG, C. & WANG, J. Ant colony optimization for resource-constrained project scheduling with generalized precedence relations. Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on, 2003. IEEE, 284-289.
MENDES, J. J. D. M., GONÇALVES, J. F. & RESENDE, M. G. 2009. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research,* 36**,** 92-109.
MERKLE, D. & MIDDENDORF, M. 2000. An ant algorithm with a new pheromone evaluation rule for total tardiness problems. *Real-World Applications of Evolutionary Computing.* Springer.
MERKLE, D., MIDDENDORF, M. & SCHMECK, H. 2002. Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on,* 6**,** 333-346.
MIKA, M., WALIGÓRA, G. & WĘGLARZ, J. 2005. Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. *European Journal of Operational Research,* 164**,** 639-668.
MONTOYA-TORRES, J. R., GUTIERREZ-FRANCO, E. & PIRACHICÁN-MAYORGA, C. 2010. Project scheduling with limited resources using a genetic algorithm. *International Journal of Project Management,* 28**,** 619-628.

NEUMANN, K. & ZIMMERMANN, J. 2000. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research,* 127**,** 425-443.
NONOBE, K. & IBARAKI, T. 2002. Formulation and tabu search algorithm for the resource constrained project scheduling problem. *Essays and surveys in metaheuristics.* Springer.
PINSON, E., PRINS, C. & RULLIER, F. Using tabu search for solving the resource-constrained project scheduling problem. Proceedings of the fourth international workshop on project management and scheduling, 1994. 102-106.
PMI 2012. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). *Project Management Institute*.
PRITSKER, A. A. B., WAITERS, L. J. & WOLFE, P. M. 1969. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science,* 16**,** 93-108.
PROON, S. & JIN, M. 2011. A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. *Naval Research Logistics (NRL),* 58**,** 73-82.
ROKOU, E., ATHENS, G. & KIRYTOPOULOS, K. 2013. MULTICRITERIA DECISION MAKING FOR PROJECT SCHEDULING UNDER RESOURCE CONSTRAINTS.
SHMOYS, D. B. & TARDOS, É. 1993. An approximation algorithm for the generalized assignment problem. *Mathematical Programming,* 62**,** 461-474.
SHTUB, A., LEBLANC, L. J. & CAI, Z. 1996. Scheduling programs with repetitive projects: a comparison of a simulated annealing, a genetic and a pair-wise swap algorithm. *European Journal of Operational Research,* 88**,** 124-138.
SPRECHER, A., HARTMANN, S. & DREXL, A. 1997. An exact algorithm for project scheduling with multiple modes. *Operations-Research-Spektrum,* 19**,** 195-203.
ST¼TZLE, T. & HOOS, H. H. 2000. MAX–MIN ant system. *Future generation computer systems,* 16**,** 889-914.
STINSON, J. P., DAVIS, E. W. & KHUMAWALA, B. M. 1978. Multiple Resource–Constrained Scheduling Using Branch and Bound. *AIIE Transactions,* 10**,** 252-259.
TAVARES, L. V. 2002. A review of the contribution of Operational Research to Project Management. *European Journal of Operational Research,* 136**,** 1-18.
THOMAS, P. R. & SALHI, S. 1998. A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics,* 4**,** 123-139.
TUKEL, O. I. & WASTI, S. N. 2001. Analysis of supplier buyer relationships using resource constrained project scheduling strategies. *European Journal of Operational Research,* 129**,** 271-276.
VANHOUCKE, M. 2006. Work continuity constraints in project scheduling. *Journal of Construction Engineering and Management,* 132**,** 14-25.
VANHOUCKE, M., COELHO, J., DEBELS, D., MAENHOUT, B. & TAVARES, L. V. 2008. An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research,* 187**,** 511-524.
VANHOUCKE, M., DEMEULEMEESTER, E. & HERROELEN, W. 2002. Discrete time/cost trade-offs in project scheduling with time-switch constraints. *Journal of the Operational Research Society***,** 741-751.
VANHOUCKE, M., DEMEULEMEESTER, E. & HERROELEN, W. 2003. Progress payments in project scheduling problems. *European Journal of Operational Research,* 148**,** 604-620.

VIANA, A. & PINHO DE SOUSA, J. 2000. Using metaheuristics in multiobjective resource constrained project scheduling. *European Journal of Operational Research,* 120**,** 359-374.
WANG, H., LI, T. & LIN, D. 2010. Efficient genetic algorithm for resource-constrained project scheduling problem. *Transactions of Tianjin University,* 16**,** 376-382.

# Παράρτημα Α **Μορφή αρχείου δεδομένων**

**********

| | |
|---|---|
| file with basedata: | mm10_.bas |
| initial value random generator: | 19747 |
| projects: | 1 |
| jobs (incl. supersource/sink : | 12 |
| horizon: | 77 |
| RESOURCES | |
| - renewable: | 2 R |
| - nonrenewable: | 2 N |
| - doubly constrained: | 0 D |

**********

**PROJECT INFORMATION:**

| Pronr. | #jobs | rel.date | duedate | tardcost | MPM-Time |
|---|---|---|---|---|---|
| **1** | 10 | 0 | 13 | 3 | 13 |

**PRECEDENCE RELATIONS:**

| jobnr. | #modes | #successors | successors |
|---|---|---|---|
| **1** | 1 | 3 | 2 3 4 |
| **2** | 3 | 2 | 5 6 |
| **3** | 3 | 2 | 10 11 |
| **4** | 3 | 1 | 9 |
| **5** | 3 | 2 | 7 8 |
| **6** | 3 | 2 | 10 11 |
| **7** | 3 | 2 | 9 10 |
| **8** | 3 | 1 | 9 |
| **9** | 3 | 1 | 12 |
| **10** | 3 | 1 | 12 |
| **11** | 3 | 1 | 12 |
| **12** | 1 | 0 | 2 3 4 |

**********

**REQUESTS/DURATIONS:**

| jobnr | mode | duration | R 1 | R 2 | N 1 | N 2 |
|---|---|---|---|---|---|---|
| **1** | 1 | 0 | 0 | 0 | 0 | 0 |
| **2** | 1 | 3 | 6 | 0 | 9 | 0 |
| | 2 | 9 | 5 | 0 | 0 | 8 |
| | 3 | 10 | 0 | 6 | 0 | 6 |
| **3** | 1 | 1 | 0 | 4 | 0 | 8 |
| | 2 | 1 | 7 | 0 | 0 | 8 |
| | 3 | 5 | 0 | 4 | 0 | 5 |
| **4** | 1 | 3 | 10 | 0 | 0 | 7 |
| | 2 | 5 | 7 | 0 | 2 | 0 |
| | 3 | 8 | 6 | 0 | 0 | 7 |
| **5** | 1 | 4 | 0 | 9 | 8 | 0 |
| | 2 | 6 | 2 | 0 | 0 | 7 |
| | 3 | 10 | 0 | 5 | 0 | 5 |
| **6** | 1 | 2 | 2 | 0 | 8 | 0 |
| | 2 | 4 | 0 | 8 | 5 | 0 |
| | 3 | 6 | 2 | 0 | 0 | 1 |
| **7** | 1 | 3 | 5 | 0 | 10 | 0 |
| | 2 | 6 | 0 | 7 | 10 | 0 |
| | 3 | 8 | 5 | 0 | 0 | 10 |
| **8** | 1 | 4 | 6 | 0 | 0 | 1 |
| | 2 | 10 | 3 | 0 | 10 | 0 |
| | 3 | 10 | 4 | 0 | 0 | 1 |
| **9** | 1 | 2 | 2 | 0 | 6 | 0 |
| | 2 | 7 | 1 | 0 | 0 | 8 |
| | 3 | 10 | 1 | 0 | 0 | 7 |
| **10** | 1 | 1 | 4 | 0 | 4 | 0 |
| | 2 | 1 | 0 | 2 | 0 | 8 |
| | 3 | 9 | 4 | 0 | 0 | 5 |
| **11** | 1 | 6 | 0 | 2 | 0 | 10 |
| | 2 | 9 | 0 | 1 | 0 | 9 |
| | 3 | 10 | 0 | 1 | 0 | 7 |
| **12** | 1 | 0 | 0 | 0 | 0 | 0 |

**********

**RESOURCEAVAILABILITIES:**

| R 1 | R 2 | N 1 | N 2 |
|---|---|---|---|
| 9 | 4 | 29 | 40 |

**********

# Παράρτημα Β **Πειραματικά αποτελέσματα**

**Πίνακας Β.9.1 Απόσπασμα αναλυτικών αποτελεσμάτων RCPSP για το σύνολο j30**

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Τυπική Απόκλιση | Συχνότητα εύρεσης βελτίστου |
|---|---|---|---|---|---|
| J301_1.RCP | 43 | 43 | 43 | 0,0% | 100% |
| J301_2.RCP | 47 | 47 | 47 | 0,0% | 100% |
| J301_3.RCP | 47 | 47 | 47 | 0,0% | 100% |
| J301_4.RCP | 62 | 63 | 62 | 0,82% | 40% |
| J301_5.RCP | 41 | 41 | 39 | 2,11% | 0% |
| J301_6.RCP | 49 | 49 | 48 | 1,06% | 0% |
| J301_7.RCP | 60 | 60 | 60 | 0,0% | 100% |
| J301_8.RCP | 53 | 53 | 53 | 0,0% | 100% |
| J301_9.RCP | 50 | 52 | 49 | 2,53% | 0% |
| J301_10.RCP | 45 | 45 | 45 | 0,0% | 100% |
| J302_1.RCP | 38 | 38 | 38 | 0,0% | 100% |
| J302_2.RCP | 51 | 53 | 51 | 1,89% | 20% |
| J302_3.RCP | 43 | 43 | 43 | 0,0% | 100% |
| J302_4.RCP | 43 | 43 | 43 | 0,0% | 100% |
| J302_5.RCP | 51 | 51 | 51 | 0,0% | 100% |
| J302_6.RCP | 47 | 47 | 47 | 0,0% | 100% |
| J302_7.RCP | 47 | 47 | 47 | 0,0% | 100% |
| J302_8.RCP | 54 | 54 | 54 | 0,0% | 100% |
| J302_9.RCP | 54 | 54 | 54 | 0,0% | 100% |
| J302_10.RCP | 43 | 43 | 43 | 0,0% | 100% |
| J303_1.RCP | 72 | 72 | 72 | 0,0% | 100% |
| J303_2.RCP | 40 | 40 | 40 | 0,0% | 100% |
| J303_3.RCP | 57 | 57 | 57 | 0,0% | 100% |
| J303_4.RCP | 98 | 98 | 98 | 0,0% | 100% |
| J303_5.RCP | 53 | 53 | 53 | 0,0% | 100% |
| J303_6.RCP | 54 | 54 | 54 | 0,0% | 100% |
| J303_7.RCP | 48 | 48 | 48 | 0,0% | 100% |
| J303_8.RCP | 54 | 54 | 54 | 0,0% | 100% |
| J303_9.RCP | 65 | 65 | 65 | 0,0% | 100% |
| J303_10.RCP | 59 | 64 | 59 | 2,94% | 40% |
| J304_1.RCP | 49 | 49 | 49 | 0,0% | 100% |
| J304_2.RCP | 60 | 60 | 60 | 0,0% | 100% |
| J304_3.RCP | 47 | 47 | 47 | 0,0% | 100% |
| J304_4.RCP | 57 | 57 | 57 | 0,0% | 100% |
| J304_5.RCP | 59 | 59 | 59 | 0,0% | 100% |
| J304_6.RCP | 45 | 45 | 45 | 0,0% | 100% |
| J304_7.RCP | 56 | 56 | 56 | 0,0% | 100% |
| J304_8.RCP | 55 | 55 | 55 | 0,0% | 100% |
| J304_9.RCP | 38 | 38 | 38 | 0,0% | 100% |
| J304_10.RCP | 48 | 48 | 48 | 0,0% | 100% |

**Πίνακας Β.9.2 Απόσπασμα αναλυτικών αποτελεσμάτων MRCPSP για το σύνολο j10**

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Τυπική Απόκλιση | Συχνότητα εύρεσης | Ελάχιστη ποινή | Μέγιστη Ποινή | Μέση ποινη | Συχνότητα λύσεων χωρις ποινη |
|---|---|---|---|---|---|---|---|---|---|
| j1010_1.mm | 17 | 17 | 17 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1010_2.mm | 22 | 24 | 24 | -1,67% | 100% | 0 | 1 | 0,7 | 30% |
| j1010_3.mm | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1010_4.mm | 15 | 15 | 15 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1010_5.mm | 22 | 22 | 24 | -2,11% | 100% | 1 | 1 | 1 | 0% |
| j1010_6.mm | 15 | 15 | 18 | -3,16% | 100% | 1 | 1 | 1 | 0% |
| j1010_7.mm | 13 | 14 | 15 | -2,73% | 100% | 0 | 1 | 0,9 | 10% |
| j1010_8.mm | 12 | 15 | 15 | 0,33% | 100% | 0 | 1 | 0,8 | 20% |
| j1010_9.mm | 10 | 11 | 10 | -0,99% | 90% | 0 | 1 | 0,3 | 70% |
| j1010_10.mm | 16 | 17 | 17 | 0,47% | 100% | 1 | 1 | 1 | 0% |
| j1011_1.mm | 20 | 20 | 20 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1011_2.mm | 11 | 11 | 11 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1011_3.mm | 12 | 12 | 12 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1011_4.mm | 23 | 23 | 23 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1011_5.mm | 15 | 16 | 15 | 0,47% | 80% | 0 | 0 | 0 | 100% |
| j1011_6.mm | 15 | 15 | 15 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1011_7.mm | 11 | 13 | 11 | 0,67% | 90% | 0 | 0 | 0 | 100% |
| j1011_8.mm | 14 | 14 | 14 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1011_9.mm | 16 | 16 | 16 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1011_10.mm | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1034_1.mm | 23 | 23 | 23 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1034_2.mm | 12 | 12 | 12 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1034_3.mm | 25 | 25 | 25 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1034_4.mm | 21 | 23 | 23 | -1,63% | 100% | 1 | 1 | 0,8 | 20% |
| j1034_5.mm | 22 | 22 | 22 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1034_6.mm | 30 | 32 | 30 | 0,67% | 90% | 0 | 0 | 0 | 100% |
| j1034_7.mm | 26 | 26 | 26 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| j1034_8.mm | 19 | 20 | 19 | 0,67% | 60% | 0 | 1 | 0,1 | 90% |
| j1034_9.mm | 26 | 27 | 26 | 1,03% | 10% | 0 | 0 | 0 | 100% |
| j1034_10.mm | 14 | 15 | 18 | -4,03% | 100% | 1 | 1 | 1 | 0% |

**Πίνακας Β.9.3 Απόσπασμα αναλυτικών αποτελεσμάτων MRCPSP για το σύνολο j12**

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Τυπική Απόκλιση | Συχνότητα εύρεσης | Ελάχιστη ποινή | Μέγιστη Ποινή | Μέση ποινη | Συχνότητα λύσεων χωρις ποινη |
|---|---|---|---|---|---|---|---|---|---|
| **j1210_1.mm** | 20 | 20 | 20 | 0,00% | 100% | 0 | 2 | 1,2 | 30% |
| **j1210_2.mm** | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1210_3.mm** | 15 | 15 | 15 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1210_4.mm** | 20 | 20 | 20 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1210_5.mm** | 16 | 16 | 16 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1210_6.mm** | 26 | 27 | 27 | -0,67% | 100% | 0 | 2 | 0,6 | 50% |
| **j1210_7.mm** | 13 | 14 | 15 | -1,94% | 100% | 1 | 1 | 1 | 0% |
| **j1210_8.mm** | 11 | 12 | 11 | 0,33% | 90% | 0 | 1 | 0,5 | 50% |
| **j1210_9.mm** | 21 | 23 | 22 | -0,88% | 80% | 0 | 1 | 0,5 | 50% |
| **j1210_10.mm** | 17 | 17 | 17 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1211_1.mm** | 17 | 17 | 17 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1211_2.mm** | 22 | 22 | 22 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1211_3.mm** | 13 | 14 | 14 | -0,67 % | 100% | 0 | 2 | 0,8 | 60% |
| **j1211_4.mm** | 17 | 18 | 17 | 0,33% | 90% | 0 | 1 | 0,1 | 90% |
| **j1211_5.mm** | 19 | 21 | 19 | -1,63% | 100% | 0 | 2 | 1,2 | 40% |
| **j1211_6.mm** | 13 | 13 | 13 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1211_7.mm** | 17 | 19 | 17 | 0,67% | 90% | 0 | 0 | 0 | 100% |
| **j1211_8.mm** | 14 | 16 | 18 | -3,89% | 100% | 0 | 2 | 1,6 | 20% |
| **j1211_9.mm** | 17 | 17 | 17 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **j1211_10.mm** | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **J1237_1.mm** | 18 | 22 | 33 | -14,06 % | 100% | 7 | 12 | 9,5 | 0% |
| **j1237_2.mm** | 25 | 28 | 37 | -12,08% | 100% | 4 | 7 | 6,4 | 0% |
| **j1237_3.mm** | 38 | 40 | 42 | -3,35% | 100% | 1 | 4 | 2,4 | 0% |
| **j1237_4.mm** | 28 | 38 | 38 | -7,63% | 100% | 0 | 10 | 6 | 10% |
| **j1237_5.mm** | 20 | 25 | 31 | -10,11% | 100% | 2 | 7 | 5,4 | 0% |
| **j1237_6.mm** | 37 | 38 | 48 | -11,20% | 100% | 5 | 7 | 6 | 0% |
| **j1237_7.mm** | 36 | 40 | 45 | -8,45% | 100% | 5 | 9 | 7,8 | 0% |
| **j1237_8.mm** | 28 | 30 | 31 | -2,45% | 100% | 0 | 2 | 0,9 | 50% |
| **j1237_9.mm** | 24 | 28 | 26 | -1,45% | 80% | 0 | 5 | 1,7 | 40% |
| **j1237_10.mm** | 24 | 27 | 42 | -18,58% | 100% | 9 | 10 | 9,8 | 0% |

**Πίνακας Β.9.4 Απόσπασμα αναλυτικών αποτελεσμάτων MRCPSP για το σύνολο c15**

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Τυπική Απόκλιση | Συχνότητα εύρεσης | Ελάχιστη ποινή | Μέγιστη Ποινή | Μέση ποινη | Συχνότητα λύσεων χωρις ποινη |
|---|---|---|---|---|---|---|---|---|---|
| **c1510_1.mm** | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1510_2.mm** | 16 | 17 | 17 | 0,82% | 80% | 0 | 0 | 0 | 100% |
| **c1510_3.mm** | 23 | 23 | 23 | 0,33% | 90% | 0 | 0 | 0 | 100% |
| **c1510_4.mm** | 38 | 41 | 39 | 1,20% | 60% | 0 | 2 | 0,8 | 40% |
| **c1510_5.mm** | 13 | 15 | 13 | 0,67% | 90% | 0 | 0 | 0 | 100% |
| **c1510_6.mm** | 32 | 32 | 32 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1510_7.mm** | 15 | 16 | 15 | 0,30% | 90% | 0 | 1 | 0,1 | 90% |
| **c1510_8.mm** | 16 | 18 | 16 | 1,53% | 60% | 0 | 3 | 0,6 | 80% |
| **c1510_9.mm** | 12 | 14 | 12 | 1,15% | 10% | 0 | 1 | 0,1 | 90% |
| **c1510_10.mm** | 14 | 15 | 14 | 0,47% | 80% | 0 | 1 | 0,1 | 90% |
| **c1511_1.mm** | 25 | 25 | 25 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1511_2.mm** | 29 | 29 | 29 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1511_3.mm** | 30 | 31 | 30 | 0,33% | 100% | 0 | 0 | 0 | 100% |
| **c1511_4.mm** | 14 | 15 | 14 | 0,33% | 90% | 0 | 0 | 0 | 100% |
| **c1511_5.mm** | 11 | 12 | 11 | 0,82% | 80% | 0 | 0 | 0 | 100% |
| **c1511_6.mm** | 16 | 18 | 18 | -1,20% | 100% | 0 | 2 | 0,4 | 80% |
| **c1511_7.mm** | 21 | 21 | 21 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1511_8.mm** | 19 | 23 | 20 | 1,29% | 40% | 0 | 1 | 0,4 | 60% |
| **c1511_9.mm** | 27 | 27 | 27 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1511_10.mm** | 30 | 30 | 30 | 0,00% | 100% | 0 | 0 | 0 | 100% |
| **c1542_1.mm** | 25 | 27 | 27 | -2,81% | 100% | 0 | 2 | 1,4 | 30% |
| **c1542_2.mm** | 20 | 23 | 22 | 0,94% | 60% | 0 | 3 | 0,5 | 70% |
| **c1542_3.mm** | 25 | 28 | 28 | -2,05% | 100% | 0 | 4 | 1,9 | 30% |
| **c1542_4.mm** | 23 | 24 | 24 | -0,58% | 100% | 0 | 2 | 1,1 | 20% |
| **c1542_5.mm** | 17 | 22 | 22 | -2,83% | 100% | 0 | 5 | 1,7 | 50% |
| **c1542_6.mm** | 22 | 24 | 24 | -1,67% | 100% | 0 | 3 | 1,4 | 30% |
| **c1542_7.mm** | 21 | 22 | 21 | 0,67% | 60% | 0 | 1 | 0,2 | 80% |
| **c1542_8.mm** | 26 | 28 | 27 | 0,67% | 100% | 0 | 1 | 0,4 | 50% |
| **c1542_9.mm** | 25 | 25 | 25 | 0,00% | 80% | 0 | 0 | 0 | 100% |
| **c1542_10.mm** | 22 | 25 | 25 | -1,45% | 100% | 0 | 3 | 1 | 40% |

**Πίνακας Β.5 Απόσπασμα αναλυτικών αποτελεσμάτων Multi Project MRCPSP**

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Μέση απόκλιση από το βέλτιστο | τυπική απόκλιση |
|---|---|---|---|---|---|
| **SP_0_1** | 22 | 24 | - | | 0,36% |
| j1010_1.mm | 17 | 17 | 17 | 0,00% | |
| j1010_2.mm | 22 | 24 | 24 | -2,00% | |
| j1010_3.mm | 21 | 21 | 21 | 0,00% | |
| j1010_4.mm | 15 | 15 | 15 | 0,00% | |
| j1010_5.mm | 22 | 22 | 24 | -2,195 | |
| **SP_0_2** | 52 | 66 | | | 2,72% |
| j1237_1.mm | 25 | 35 | 33 | -5,91% | |
| j1237_2.mm | 28 | 43 | 37 | -6,36% | |
| j1237_3.mm | 43 | 52 | 42 | 5,495 | |
| j1237_4.mm | 29 | 43 | 38 | 4,48% | |
| j1237_5.mm | 23 | 39 | 31 | 4,855 | |
| **SP_1_1** | 23 | 27 | - | | 1,00% |
| j1010_6.mm | 13 | 16 | 18 | -4,09% | |
| j1010_7.mm | 13 | 14 | 15 | -1,94% | |
| j1010_8.mm | 11 | 17 | 15 | -2,91% | |
| j1010_9.mm | 10 | 11 | 10 | 0,33% | |
| j1010_10.mm | 13 | 17 | 17 | -3,07% | |
| **SP_1_2** | 87 | 93 | | | 1,52% |
| j1237_6.mm | 36 | 46 | 48 | -11,37% | |
| j1237_7.mm | 36 | 42 | 45 | -7,40% | |
| j1237_8.mm | 28 | 33 | 31 | -2,31% | |
| j1237_9.mm | 24 | 30 | 26 | 2,29% | |
| j1237_10.mm | 23 | 32 | 42 | -17,46% | |
| **SP_2_1** | 43 | 43 | | | 0,00% |
| j1010_1.mm | 17 | 17 | 17 | 0,00% | |
| j1010_2.mm | 22 | 24 | 24 | -2,00% | |
| j1010_3.mm | 21 | 21 | 21 | 0,00% | |
| j1010_4.mm | 15 | 15 | 15 | 0,00% | |
| j1010_5.mm | 22 | 22 | 24 | -2,12% | |
| **SP_2_2** | 28 | 34 | | | 1,50% |
| j1010_6.mm | 13 | 15 | 18 | -4,23% | |
| j1010_7.mm | 13 | 13 | 15 | -2,12% | |
| j1010_8.mm | 11 | 15 | 15 | -2,49% | |
| j1010_9.mm | 10 | 10 | 10 | 0,00% | |
| j1010_10.mm | 14 | 18 | 17 | -1,92% | |

| Όνομα αρχείου | Ελάχιστη διάρκεια | Μέγιστη διάρκεια | Βέλτιστο | Μέση απόκλιση από το βέλτιστο | τυπική απόκλιση |
|---|---|---|---|---|---|
| **SP_2_3** | 97 | 109 | | | 2,60% |
| j1237_1.mm | 18 | 21 | 33 | -14,50% | |
| j1237_2.mm | 22 | 29 | 37 | -13,32% | |
| j1237_3.mm | 35 | 45 | 42 | -4,46% | |
| j1237_4.mm | 26 | 45 | 38 | -9,75% | |
| j1237_5.mm | 20 | 27 | 31 | -10,13% | |
| **SP_2_4** | 87 | 94 | | | 1,90% |
| j1237_6.mm | 35 | 41 | 48 | -11,82% | |
| j1237_7.mm | 35 | 48 | 45 | -7,02% | |
| j1237_8.mm | 28 | 30 | 31 | -2,56% | |
| j1237_9.mm | 24 | 29 | 26 | -1,91% | |
| j1237_10.mm | 24 | 35 | 42 | -17,24% | |
| **SP_3_1** | 43 | 43 | | | 0,00% |
| j1010_1.mm | 17 | 17 | 17 | 0,00% | |
| j1010_2.mm | 22 | 24 | 24 | -1,76% | |
| j1010_3.mm | 21 | 22 | 21 | 0,33% | |
| j1010_4.mm | 15 | 16 | 15 | 0,33% | |
| j1010_5.mm | 22 | 22 | 24 | -2,12% | |
| **SP_3_2** | 55 | 61 | | | 1,41% |
| j1010_6.mm | 13 | 15 | 18 | -4,14% | |
| j1010_7.mm | 13 | 13 | 15 | -2,12% | |
| j1010_8.mm | 11 | 15 | 15 | -2,29% | |
| j1010_9.mm | 10 | 10 | 10 | 0,00% | |
| j1010_10.mm | 14 | 17 | 17 | -2,08% | |
| **SP_3_3** | 111 | 120 | | | 2,72% |
| j1237_1.mm | 18 | 23 | 33 | -13,20% | |
| j1237_2.mm | 22 | 32 | 37 | -13,16% | |
| j1237_3.mm | 36 | 42 | 42 | -3,48% | |
| j1237_4.mm | 26 | 39 | 38 | -8,63% | |
| j1237_5.mm | 20 | 24 | 31 | -10,12% | |
| **SP_3_4** | 117 | 126 | | | 1,64% |
| j1237_6.mm | 35 | 37 | 48 | -12,89% | |
| j1237_7.mm | 35 | 39 | 45 | -9,16% | |
| j1237_8.mm | 28 | 31 | 31 | -2,16% | |
| j1237_9.mm | 23 | 28 | 26 | -1,79% | |
| j1237_10.mm | 23 | 24 | 42 | -19,08% | |

# Παράρτημα Γ **Βασικά τμήματα κώδικα**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ACO
{
    class ALACO
    {
        public CommonVariables.CurrentInstance
ActivityListACO(CommonVariables.ProblemData ProblemData,
CommonVariables.CurrentInstance CurrentIn)
        {
            #region aux
            ProblemData.ALACO.Generations = 25;
            ProblemData.ALACO.Population = 5;
            ProblemData.ALACO.gMax = 10;
            ProblemData.ALACO.ρ = 0.025;
            ProblemData.ALACO.α = 1;
            ProblemData.ALACO.β = 2;
            ProblemData.ALACO.γ = 1;
            ProblemData.ALACO.c = 0.5;
            int Generations = ProblemData.ALACO.Generations;
            int Population = ProblemData.ALACO.Population;
            double βStart = 2;
            double ρStart = 0.025;
            #endregion

            #region ActivityList ACO variables initialization
            int JobNum =
ProblemData.SuperProject.Projects[CurrentIn.ProjectID].Activities.Count;
            int[] ModeList = new int[JobNum];
            List<int> ScheduledSet = new List<int>();
            int[,] ActivityList = new int[Population, JobNum];
            int[,] ProjectSchedule = new int[Population, JobNum];
            int[] BS = new int[JobNum];
            int[] IB = new int[JobNum];
            for (int i = 0; i < JobNum; i++)
            {
                BS[i] = i;
                IB[i] = -1;
                for (int j = 0; j < Population; j++)
                {
                    ActivityList[j, i] = -1;
                    ProjectSchedule[j, i] = -1;
                }
            }
            ModeList = CurrentIn.ModeList;
            int[] ListFitness = new int[Population];
            for (int i = 0; i < Population; i++)
                ListFitness[i] = -1;
            double[,] τ = new double[JobNum, JobNum];
            for (int i = 0; i < JobNum; i++)
            {
                for (int j = 0; j < JobNum; j++)
                {
                    τ[i, j] = 0.1;
```

```
                }
            }
            #endregion

            #region ACO preproccess
            SSGS SSGS_Ant = new SSGS();
            AuxiliaryFunctions Aux = new AuxiliaryFunctions();
            //Using CPM for ES,EF,LS,LF for Heuristics
            int[] ESACTLIST = Aux.CPM(ref ProblemData, CurrentIn);
            int[] Duration = Aux.Duration(ProblemData, CurrentIn);
            //Heuristics
            int[] LST = LSTHeuristic(ProblemData, CurrentIn, JobNum);
            int[] LFT = LFTHeuristic(ProblemData, CurrentIn, JobNum);
            int[] MSL = MSLHeuristic(ProblemData, CurrentIn, JobNum, LST, LFT);
            //Using Preds & SSGS for EligibleSet Calculation
            List<int>[] PredsSet = Aux.PredsSet(ProblemData, CurrentIn);
            CurrentIn.ActivityList = new int[JobNum];
            //Initiallizing BS
            CommonVariables.CurrentInstance BSInstance = new
CommonVariables.CurrentInstance();
            BSInstance.ActivityList = BS;
            BSInstance.ModeList = CurrentIn.ModeList;
            BSInstance.ProjectID = CurrentIn.ProjectID;
            BSInstance = SSGS_Ant.SSGSFunction(ProblemData, BSInstance);
            int BSCounter = 0;
            int BS_Fitness = Generations * Population * ProblemData.ALACO.gMax
* 1000;
            int[] BSSchedule = new int[JobNum];
            int SolutionCounter = 0;
            #endregion

            for (int Gens = 0; Gens < Generations; Gens++)
            {
                for (int Pop = 0; Pop < Population; Pop++)
                {
                    #region OneAnt
                    ScheduledSet.Add(0);
                    ActivityList[Pop, 0] = 0;
                    CurrentIn.ActivityList[0] = 0;
                    for (int i = 1; i < JobNum; i++)
                    {
                        List<int> EligibleSet =
SSGS_Ant.EligibleSetCalc(ProblemData, CurrentIn, ScheduledSet, PredsSet);
                        int[] nLST = Normalize(LST, EligibleSet);
                        int[] nLFT = Normalize(LFT, EligibleSet);
                        int[] nMSL = Normalize(MSL, EligibleSet);
                        double[] Prob = Probability(CurrentIn,
ProblemData.ALACO, EligibleSet, τ, nLST, nLFT, nMSL, Pop, i);
                        int jSelected = ActivityPSM(EligibleSet, Prob);
                        ScheduledSet.Add(jSelected);
                        ActivityList[Pop, i] = ScheduledSet[i];
                        CurrentIn.ActivityList[i] = ScheduledSet[i];
                    }
                    CurrentIn = SSGS_Ant.SSGSFunction(ProblemData, CurrentIn);

                    for (int i = 0; i < JobNum; i++)
                    {
                        ProjectSchedule[Pop, i] = CurrentIn.ProjectSchedule[i];
                    }
                    ListFitness[Pop] = CurrentIn.Duration;
                    CurrentIn = ReInstance(CurrentIn);
                    ScheduledSet.Clear();
```

```
                #endregion
            }

            #region IB/BS Update Rule
            int[] IBplusPosition = new int[JobNum + 1];
            int[] IBSchedule = new int[JobNum];
            IBplusPosition = IBFunction(ActivityList, ListFitness, JobNum,
Population);
            int IB_Fitness = ListFitness[IBplusPosition[JobNum]];
            for (int i = 0; i < JobNum; i++)
            {
                IBSchedule[i] = ProjectSchedule[IBplusPosition[JobNum], i];
            }

            for (int i = 0; i < JobNum; i++)
                IB[i] = IBplusPosition[i];
            if (BSCounter < ProblemData.ALACO.gMax)
            {
                BSCounter++;
                if (IB_Fitness < BS_Fitness)
                {
                    BS = IB;
                    BS_Fitness = IB_Fitness;
                    BSSchedule = IBSchedule;
                    BSCounter = 0;
                }
            }
            else
            {
                BS = IB;
                BS_Fitness = IB_Fitness;
                BSSchedule = IBSchedule;
                BSCounter = 0;
            }
            if (BS_Fitness <= BSInstance.Duration)
            {
                if (BS_Fitness < BSInstance.Duration)
                {
                    SolutionCounter = 0;
                }
                else
                {
                    SolutionCounter++;
                }
                BSInstance.Duration = BS_Fitness;
                BSInstance.ActivityList = BS;
                BSInstance.ProjectSchedule = BSSchedule;
            }
            else
            {
                SolutionCounter++;
            }
            #endregion

            #region ACO var update
            UpdateACO(ref ProblemData, Gens, βStart);
            #endregion

            #region Pheromone Model Update
            UpdateFunction(ref ActivityList, ref ProjectSchedule, ref
ListFitness, BS, BS_Fitness, IB, IB_Fitness, JobNum, Population,
ProblemData.ALACO, ref τ);
```

```
                #endregion

                #region Convergence Condition
            // if (SolutionCounter >= (ProblemData.ALACO.Generations / 2))
            //        Gens = Generations;
                #endregion
            }

            #region Checking Usage for Mistakes in Algo
            int[,] NonRenUsage = new
int[ProblemData.ResAvail.NonRenResAvailable.Count, JobNum + 1];
            int[,] RenUsage = new
int[ProblemData.ResAvail.RenResAvailable.Count, BSInstance.Duration + 1];

            for (int U = 0; U < ProblemData.ResAvail.RenResAvailable.Count;
U++)
            {
                int MAX = 0;
                for (int d = 0; d < BSInstance.Duration; d++)
                {
                    RenUsage[U, d] = RResUsageMatrix(ProblemData, BSInstance,
U, d, Duration);
                    if (MAX < RenUsage[U, d])
                        MAX = RenUsage[U, d];
                }
                RenUsage[U, BSInstance.Duration] = MAX;
            }

            for (int U = 0; U < ProblemData.ResAvail.NonRenResAvailable.Count;
U++)
            {
                int SUM = 0;
                for (int d = 0; d < JobNum; d++)
                {
                    NonRenUsage[U, d] =
ProblemData.SuperProject.Projects[BSInstance.ProjectID].Activities[d].Modes[BSI
nstance.ModeList[d]].ResNeed.NonRenResNeed[U].ResMatrix[0];
                    SUM = SUM + NonRenUsage[U, d];
                }
                NonRenUsage[U, JobNum] = SUM;
                if (NonRenUsage[U, JobNum] >
ProblemData.SuperProject.Projects[BSInstance.ProjectID].ResAvail.NonRenResAvail
able[U].ResMatrix[0])
                {
                    BSInstance.Penalty = BSInstance.Penalty + (NonRenUsage[U,
JobNum] -
ProblemData.SuperProject.Projects[BSInstance.ProjectID].ResAvail.NonRenResAvail
able[U].ResMatrix[0]);
                }
            }
            #endregion

            ProblemData.ALACO.β = βStart;
            ProblemData.ALACO.ρ = ρStart;
            return BSInstance;
        }

        public int[] Normalize(int[] Heuristic, List<int> Eligibles)
        {
            int[] nHeuristic = new int[Eligibles.Count];
            int[] TEMP = new int[Eligibles.Count];
            for (int i = 0; i < Eligibles.Count; i++)
```

```
            {
                TEMP[i] = Heuristic[Eligibles[i]];
            }
            int Hmax = TEMP.Max();
            for (int i = 0; i < TEMP.Length; i++)
            {
                nHeuristic[i] = Hmax - TEMP[i] + 1;
            }
            return nHeuristic;
        }
        public int[] LSTHeuristic(CommonVariables.ProblemData ProblemData,
CommonVariables.CurrentInstance CurrentIn, int JobNum)
        {
            int[] LST = new int[JobNum];
            for (int i = 0; i < JobNum; i++)
            {
                LST[i] =
ProblemData.SuperProject.Projects[CurrentIn.ProjectID].Activities[i].LS;
            }
            return LST;
        }
        public int[] LFTHeuristic(CommonVariables.ProblemData ProblemData,
CommonVariables.CurrentInstance CurrentIn, int JobNum)
        {
            int[] LFT = new int[JobNum];
            for (int i = 0; i < JobNum; i++)
            {
                LFT[i] =
ProblemData.SuperProject.Projects[CurrentIn.ProjectID].Activities[i].LF;
            }
            return LFT;
        }
        public int[] MSLHeuristic(CommonVariables.ProblemData ProblemData,
CommonVariables.CurrentInstance CurrentIn, int JobNum, int[] LST, int[] LFT)
        {
            int[] MSL = new int[JobNum];
            for (int i = 0; i < JobNum; i++)
            {
                MSL[i] = LFT[i] - LST[i];
            }
            return MSL;
        }

        public double[] Probability(CommonVariables.CurrentInstance CurrentIn,
CommonVariables.ActivityListACO ACO, List<int> Eligibles, double[,] τ, int[]
nLST, int[] nLFT, int[] nMSL, int Pop, int i)
        {
            double[] newτ = new double[Eligibles.Count];
            double Xi = 0;
            double Yi = 0;
            double GT = 0;
            for (int h = 0; h < Eligibles.Count; h++)
            {
                for (int k = 1; k <= i; k++)
                {
                    Xi = Xi + Math.Pow(ACO.γ, i - k) * τ[k, h];

                }
                Yi = Yi + τ[i, h];
            }
            for (int j = 0; j < Eligibles.Count; j++)
            {
```

```
                for (int k = 1; k <= i; k++)
                {
                    GT = GT + Math.Pow(ACO.γ, i - k) * τ[k, j];
                }

                newτ[j] = ACO.c * Xi * τ[i, j] + (1 - ACO.c) * Yi * GT;
            }
            double SumEligible_ητ = 0;
            //Setting the heuristics:
            int[] HeuristicsValue = new int[Eligibles.Count];
            for (int j = 0; j < Eligibles.Count; j++)
            {
                if (Pop < 5)
                    HeuristicsValue[j] = nLST[j];
                else if ((5 <= Pop) && (Pop < 10))
                    HeuristicsValue[j] = nLFT[j];
                else if ((10 <= Pop) && (Pop < 15))
                    HeuristicsValue[j] = nMSL[j];

                SumEligible_ητ = SumEligible_ητ + (Math.Pow(newτ[j], ACO.α) *
Math.Pow(HeuristicsValue[j], ACO.β));

            }
            double[] PROB = new double[Eligibles.Count];

            for (int j = 0; j < Eligibles.Count; j++)
            {
                PROB[j] = (Math.Pow(newτ[j], ACO.α) *
Math.Pow(HeuristicsValue[j], ACO.β)) / SumEligible_ητ;
            }
            return PROB;
        }
        public int ActivityPSM(List<int> Eligibles, double[] Prob)
        {
            int jSelected = -1;
            double[] DimensionArray = new double[Eligibles.Count];
            double SUM = 0;
            for (int i = 0; i < Eligibles.Count; i++)
            {
                SUM = SUM + Prob[i];
                DimensionArray[i] = SUM;
            }
            DimensionArray[Eligibles.Count - 1] = 1;
            Random RandFun = new Random();
            double Ball = RandFun.NextDouble();
            int j = 0;
            do
            {
                if (Ball <= DimensionArray[j])
                    jSelected = Eligibles[j];
                j++;
            } while (jSelected == -1);
            return jSelected;
        }
        public CommonVariables.CurrentInstance
ReInstance(CommonVariables.CurrentInstance CurrentIn)
        {
            for (int i = 0; i < CurrentIn.ActivityList.Length; i++)
            {
                CurrentIn.ActivityList[i] = -1;
                CurrentIn.ProjectSchedule[i] = -1;
            }
```

```
            CurrentIn.Duration = 0;
            CurrentIn.Penalty = 0;
            return CurrentIn;
        }

        public int[] IBFunction(int[,] ActivityLists, int[] ListFitness, int
JobNum, int Population)
        {
            int[] IB = new int[JobNum + 1];
            int Counter = 0;
            for (int i = 0; i < Population; i++)
            {
                if (ListFitness[i] < ListFitness[Counter])
                    Counter = i;
            }
            for (int i = 0; i < JobNum; i++)
            {
                IB[i] = ActivityLists[Counter, i];
            }
            IB[JobNum] = Counter;
            return IB;
        }

        public void UpdateACO(ref CommonVariables.ProblemData ProblemData, int
Gens, double bStart)
        {
            if (Gens > (ProblemData.ALACO.Generations * 2 / 3))
                ProblemData.ALACO.ρ = 0.075;
            ProblemData.ALACO.β = (-4 / ProblemData.ALACO.Generations) * Gens +
bStart;
            if (ProblemData.ALACO.β <= 0)
                ProblemData.ALACO.β = 0;
        }
        public void UpdateFunction(ref int[,] ActivityLists, ref int[,]
ProjectSchedules, ref int[] ListFitness, int[] BS, int BSFitness, int[] IB, int
IBFitness, int JobNum, int Population, CommonVariables.ActivityListACO ACO, ref
double[,] τ)
        {
            //WEIGHT
            double BSFitnessValue = (double)(1) / (double)(2 * BSFitness);
            double IBFitnessValue = (double)(1) / (double)(2 * IBFitness);
            for (int i = 0; i < JobNum; i++)
            {
                for (int j = 0; j < JobNum; j++)
                {
                    τ[i, j] = (1 - ACO.ρ) * τ[i, j];
                }
            }
            for (int i = 0; i < JobNum; i++)
            {
                for (int j = 0; j < JobNum; j++)
                {
                    if (j == BS[i])
                    {
                        τ[i, j] = τ[i, j] + ACO.ρ * BSFitnessValue;
                    }

                    if (j == IB[i])
                    {
                        τ[i, j] = τ[i, j] + ACO.ρ * IBFitnessValue;
                    }
```

```
                    if (τ[i, j] < BSFitnessValue * ACO.ρ)
                    {
                        τ[i, j] = BSFitnessValue * ACO.ρ;
                    }

                    if (τ[i, j] > (BSFitnessValue / ACO.ρ))
                    {
                        τ[i, j] = BSFitnessValue / ACO.ρ;
                    }
                }
            }

            for (int i = 0; i < Population; i++)
            {
                ListFitness[i] = -1;
                for (int j = 0; j < JobNum; j++)
                {
                    ActivityLists[i, j] = -1;
                    ProjectSchedules[i, j] = -1;
                }
            }
        }

        public int RResUsageMatrix(CommonVariables.ProblemData ProblemData,
CommonVariables.CurrentInstance BSIn, int RenResID, int tCheck, int[] Duration)
        {
            int ProjectID = BSIn.ProjectID;
            int JobNum =
ProblemData.SuperProject.Projects[ProjectID].Activities.Count;
            int[] StartTimes = new int[JobNum];
            int[] FinishTimes = new int[JobNum];
            for (int i = 0; i < JobNum; i++)
            {
                StartTimes[i] = BSIn.ProjectSchedule[i];
                FinishTimes[i] = StartTimes[i] + Duration[i];
            }
            int RenResUsage = 0;

            for (int i = 0; i < JobNum; i++)
            {
                if (tCheck >= StartTimes[i] && tCheck < FinishTimes[i])
                {
                    RenResUsage = RenResUsage +
ProblemData.SuperProject.Projects[ProjectID].Activities[i].Modes[BSIn.ModeList[
i]].ResNeed.RenResNeed[RenResID].ResMatrix[0];
                }
            }
            return RenResUsage;
        }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ACO
{
    class MODACO
    {
        public void ModeratorACO(ref CommonVariables.ProblemData ProblemData)
        {
            #region ModeratorACO Variables Initialization
            #region aux
            ProblemData.MODACO.Generations = 249;
            ProblemData.MODACO.Population = 5;
            ProblemData.MODACO.gMax = 10;
            ProblemData.MODACO.ρ = 0.025;
            ProblemData.MODACO.PenaltyWeight = 1;
            int Generations = ProblemData.MODACO.Generations;
            int Population = ProblemData.MODACO.Population;
            int gMax = ProblemData.MODACO.gMax;
            double PenaltyWeightStart = 1;
            int ProjectNum = ProblemData.SuperProject.Projects.Count;
            int SubsetNum = ProblemData.SuperProject.Subsets.Count;
            #endregion
            //Mode Pheromone Model Initialization & Subset Pheromone Model
Initialization

            double[][][] τ = new double[ProjectNum][][];
            double[,] τSubset = new double[ProjectNum, SubsetNum];

            for (int P = 0; P < ProjectNum; P++)
            {
                int JobNum =
ProblemData.SuperProject.Projects[P].Activities.Count;
                τ[P] = new double[JobNum][];
                for (int i = 0; i < JobNum; i++)
                {
                    int ModeNum =
ProblemData.SuperProject.Projects[P].Activities[i].Modes.Count;
                    τ[P][i] = new double[ModeNum];
                    for (int j = 0; j < ModeNum; j++)
                    {
                        τ[P][i][j] = 0.1;
                    }
                }
                for (int S = 0; S < SubsetNum; S++)
                {
                    if (ProblemData.SuperProject.Projects[P].Subset == S)
                    {
                        τSubset[P, S] = 0.95;
                    }
                    else
                    {
                        τSubset[P, S] = 0.05;
                    }
                }
            }
            //Fitness Vectors Initialization
            //For Subset Prob Update
            double[] SolFitness = new double[Population];
```

```
            for (int i = 0; i < Population; i++)
                SolFitness[i] = -1;
            //For Mode Prob Update
            double[,] SolFitnessPerP = new double[Population, ProjectNum];
            for (int i = 0; i < Population; i++)
            {
                for (int j = 0; j < ProjectNum; j++)
                {
                    SolFitnessPerP[i, j] = -1;
                }
            }
            #endregion

            #region ModeratorACO Preprocess
            //MULTIPROJECT
            //OVERALL BEST SOLUTION
            CommonVariables.CurrentInstance BSSuperInstance_Tot = new
CommonVariables.CurrentInstance();
            CommonVariables.Project BSSuperP_Tot = new
CommonVariables.Project();
            List<CommonVariables.CurrentInstance> BSProjectInstaTot_Tot = new
List<CommonVariables.CurrentInstance>();
            double BSSuperInstance_Tot_Fitness = Generations * Population *
gMax * 1000; //Fitness' Safety Initialization
            //BEST SOLUTION
            CommonVariables.CurrentInstance BSSuperInstance = new
CommonVariables.CurrentInstance();
            CommonVariables.Project BSSuperP = new CommonVariables.Project();
            List<CommonVariables.CurrentInstance> BSProjectInstaTot = new
List<CommonVariables.CurrentInstance>();
            double BSSuperInstance_Fitness = Generations * Population * gMax *
1000; //Fitness' Safety Initialization
            //ITETRATION BEST
            CommonVariables.CurrentInstance IBSuperInstance = new
CommonVariables.CurrentInstance();
            CommonVariables.Project IBSuperP = new CommonVariables.Project();
            List<CommonVariables.CurrentInstance> IBProjectInstaTot = new
List<CommonVariables.CurrentInstance>();
            double IBSuperInstance_Fitness = -1;
            int BSCounter = 0;
            //MULTIMODE
            //OVERALL BEST SOLUTION
            List<CommonVariables.CurrentInstance> BSProjectInstaTot_Tot_2 = new
List<CommonVariables.CurrentInstance>();
            double[] BSforProject_Tot_2 = new double[ProjectNum];
            //BEST SOLUTION
            List<CommonVariables.CurrentInstance> BSProjectInstaTot_2 = new
List<CommonVariables.CurrentInstance>();
            double[] BSforProject_2 = new double[ProjectNum];
            //ITERATION BEST
            List<CommonVariables.CurrentInstance> IBProjectInstaTot_2 = new
List<CommonVariables.CurrentInstance>();
            double[] IBforProject_2 = new double[ProjectNum];
            for (int i = 0; i < ProjectNum; i++)
            {
                BSforProject_Tot_2[i] = Generations * Population * gMax * 1000;
//Fitness' Safety Initialization
                BSforProject_2[i] = Generations * Population * gMax * 1000;
//Fitness' Safety Initialization
                IBforProject_2[i] = -1;
                CommonVariables.CurrentInstance CuIn = new
CommonVariables.CurrentInstance();
```

```
                CuIn.Duration = Generations * Population * gMax * 1000;
//Fitness' Safety Initialization
                CuIn.Penalty = Generations * Population * gMax * 1000;
//Fitness' Safety Initialization
                BSProjectInstaTot_Tot_2.Add(CuIn);
                BSProjectInstaTot_2.Add(CuIn);
                IBProjectInstaTot_2.Add(CuIn);
            }
            int[] BSCounter_2 = new int[ProjectNum];
            for (int i = 0; i < ProjectNum; i++)
            {
                BSCounter_2[i] = 0;
            }

            //RES USE
            int ResNum = ProblemData.ResAvail.RenResAvailable.Count +
ProblemData.ResAvail.NonRenResAvailable.Count;
            int[,] ResUsage = new int[ProjectNum, ResNum];
            #endregion

            for (int Gens = 0; Gens < Generations; Gens++)
            {
                List<List<CommonVariables.CurrentInstance>> ProjectInstaTot =
new List<List<CommonVariables.CurrentInstance>>();
                List<CommonVariables.CurrentInstance> SuperInstances = new
List<CommonVariables.CurrentInstance>();
                List<CommonVariables.Project> SuperPs = new
List<CommonVariables.Project>();
                for (int Pop = 0; Pop < Population; Pop++)
                {
                    #region OneAnt
                    List<CommonVariables.CurrentInstance> ProjectInstances =
new List<CommonVariables.CurrentInstance>();
                    for (int P = 0; P < ProjectNum; P++)
                    {
                        CommonVariables.CurrentInstance CurrentIn = new
CommonVariables.CurrentInstance();
                        CurrentIn.ProjectID = P;
                        CurrentIn.ModeList = ModeListCalculator(ProblemData, P,
τ);

                        ALACO ActivityACO = new ALACO();
                        CurrentIn = ActivityACO.ActivityListACO(ProblemData,
CurrentIn);

                        CurrentIn.Subset = SubsetPSM(ProblemData, P, τSubset);
                        ResUse(ref ResUsage, ProblemData, ref CurrentIn, P);
                        ProjectInstances.Add(CurrentIn);
                    }

                    ProjectInstaTot.Add(ProjectInstances);
                    for (int PrIn = 0; PrIn < ProjectInstances.Count; PrIn++)
                    {
                        SolFitnessPerP[Pop, PrIn] =
ProjectInstances[PrIn].Duration + (ProjectInstances[PrIn].Penalty *
ProblemData.MODACO.PenaltyWeight);
                    }

                    List<CommonVariables.Subset> SubsetsNew = new
List<CommonVariables.Subset>();
                    SubsetsNew = SubsetsReranking(ProblemData,
ProjectInstances);
```

```
                    CommonVariables.CurrentInstance SuperInstance = new
CommonVariables.CurrentInstance();
                    CommonVariables.Project SuperP = new
CommonVariables.Project();
                    Transformation(ProblemData, ProjectInstances, ref SuperP,
ref SuperInstance, ResUsage, SubsetsNew);
                    SuperInstances.Add(SuperInstance);
                    SuperPs.Add(SuperP);
                    SolFitness[Pop] = SuperInstances[Pop].Duration +
SuperInstances[Pop].Penalty * ProblemData.MODACO.PenaltyWeight;
                    #endregion
                }
                #region IB/BS Update Rule
                IBFunction(SolFitness, SolFitnessPerP, SuperInstances, SuperPs,
ProjectInstaTot, ref IBSuperInstance, ref IBSuperP, ref IBProjectInstaTot, ref
IBProjectInstaTot_2, ref IBSuperInstance_Fitness, ref IBforProject_2);

                #region MultiMode Optimization
                //Using the _2 variables
                for (int i = 0; i < ProjectNum; i++)
                {
                    if (BSCounter_2[i] < ProblemData.MODACO.gMax)
                    {
                        BSCounter_2[i]++;
                        if (IBProjectInstaTot_2[i].Duration <=
BSProjectInstaTot_2[i].Duration)
                        {
                            if (IBProjectInstaTot_2[i].Duration ==
BSProjectInstaTot_2[i].Duration)
                            {
                                if (IBProjectInstaTot_2[i].Penalty <=
BSProjectInstaTot_2[i].Penalty)
                                {
                                    BSforProject_2[i] = IBforProject_2[i];
                                    BSProjectInstaTot_2[i] =
IBProjectInstaTot_2[i];
                                    BSCounter_2[i] = 0;
                                }
                            }
                            else
                            {
                                BSforProject_2[i] = IBforProject_2[i];
                                BSProjectInstaTot_2[i] =
IBProjectInstaTot_2[i];
                                BSCounter_2[i] = 0;
                            }
                        }
                    }
                    else
                    {
                        BSforProject_2[i] = IBforProject_2[i];
                        BSProjectInstaTot_2[i] = IBProjectInstaTot_2[i];
                        BSCounter_2[i] = 0;
                    }
                    if (BSProjectInstaTot_2[i].Duration <=
BSProjectInstaTot_Tot_2[i].Duration)
                    {
                        if (BSProjectInstaTot_2[i].Duration ==
BSProjectInstaTot_Tot_2[i].Duration)
                        {
```

```
                            if (BSProjectInstaTot_2[i].Penalty <=
BSProjectInstaTot_Tot_2[i].Penalty)
                            {
                                BSforProject_Tot_2[i] = BSforProject_2[i];
                                BSProjectInstaTot_Tot_2[i] =
BSProjectInstaTot_2[i];
                            }
                        }
                        else
                        {
                            BSforProject_Tot_2[i] = BSforProject_2[i];
                            BSProjectInstaTot_Tot_2[i] =
BSProjectInstaTot_2[i];
                        }
                    }
                }

                #endregion

                #region MultiProject Optimization
                if (BSCounter < ProblemData.MODACO.gMax)
                {
                    BSCounter++;
                    if (IBSuperInstance_Fitness < BSSuperInstance_Fitness)
                    {
                        BSSuperInstance_Fitness = IBSuperInstance_Fitness;
                        BSSuperInstance = IBSuperInstance;
                        BSSuperP = IBSuperP;
                        BSProjectInstaTot = IBProjectInstaTot;
                        BSCounter = 0;
                    }
                    else
                    {
                        BSSuperInstance_Fitness = IBSuperInstance_Fitness;
                        BSSuperInstance = IBSuperInstance;
                        BSSuperP = IBSuperP;
                        BSProjectInstaTot = IBProjectInstaTot;
                        BSCounter = 0;
                    }
                    if (BSSuperInstance_Fitness <= BSSuperInstance_Tot_Fitness)
                    {
                        BSSuperInstance_Tot_Fitness = BSSuperInstance_Fitness;
                        BSSuperInstance_Tot = BSSuperInstance;
                        BSSuperP_Tot = BSSuperP;
                        BSProjectInstaTot_Tot = BSProjectInstaTot;
                    }
                }
                #endregion
                #endregion

                #region ACO var update
                ModeratorUpdateACO(ref ProblemData, Gens, PenaltyWeightStart);
                #endregion

                #region Update Function
                ModeUpdateFunction(ref τ, ProblemData, BSProjectInstaTot_Tot_2,
BSforProject_Tot_2, IBProjectInstaTot_2, IBforProject_2);
                SubsetUpdateFunction(ref τSubset, ProblemData, BSSuperP_Tot,
IBSuperP, BSSuperInstance_Tot, BSSuperInstance_Tot_Fitness, IBSuperInstance,
IBSuperInstance_Fitness);
                #endregion
            }
```

```
            ProblemData.SuperProject.TotalDuration =
(int)BSSuperInstance_Tot_Fitness;
            for (int i = 0; i < ProjectNum; i++)
                ProblemData.SuperProject.Projects[i].Duration =
BSProjectInstaTot_Tot[i].Duration;
        }
        public int[] ModeListCalculator(CommonVariables.ProblemData
ProblemData, int ProjectID, double[][][] τ)
        {
            int JobNum =
ProblemData.SuperProject.Projects[ProjectID].Activities.Count;
            int[] ModeList = new int[JobNum];
            for (int J = 0; J < JobNum; J++)
            {
                int ModeNum =
ProblemData.SuperProject.Projects[ProjectID].Activities[J].Modes.Count;
                ModeList[J] = ModePSM(ProjectID, J, ModeNum, τ);
            }
            return ModeList;
        }
        public int ModePSM(int ProjectID, int JobID, int ModeNum, double[][][]
τ)
        {
            int Mode = -1;

            //Probability caculation for each mode
            double SumMode_τ = 0;
            for (int j = 0; j < ModeNum; j++)
            {
                SumMode_τ = SumMode_τ + τ[ProjectID][JobID][j];
            }
            double[] PROB = new double[ModeNum];
            for (int j = 0; j < ModeNum; j++)
            {
                PROB[j] = τ[ProjectID][JobID][j] / SumMode_τ;
            }

            //Probabilistic Selection Mechanism
            //roulette floor
            double[] DimensionArray = new double[ModeNum];
            double SUM = 0;
            for (int m = 0; m < ModeNum; m++)
            {
                SUM = SUM + PROB[m];
                DimensionArray[m] = SUM;
            }
            DimensionArray[ModeNum - 1] = 1;
            //roulette ball
            double Ball = RandFun.NextDouble();

            //winner check
            int i = 0;
            do
            {
                if (Ball <= DimensionArray[i])
                    Mode = i;
                i++;
            } while (Mode == -1);

            return Mode;
        }
```

```
        public int SubsetPSM(CommonVariables.ProblemData ProblemData, int
ProjectID, double[,] τSubset)
        {
            int Subset = -1;
            int ProjectNum = ProblemData.SuperProject.Projects.Count;
            int SubsetNum = ProblemData.SuperProject.Subsets.Count;
            //Probability caculation for each mode
            double Sum_τSubset = 0;
            for (int j = 0; j < SubsetNum; j++)
            {
                Sum_τSubset = Sum_τSubset + τSubset[ProjectID, j];
            }
            double[] PROB = new double[SubsetNum];
            for (int j = 0; j < SubsetNum; j++)
            {
                PROB[j] = τSubset[ProjectID, j] / Sum_τSubset;
            }

            //Probabilistic Selection Mechanism
            //roulette floor
            double[] DimensionArray = new double[SubsetNum];
            double SUM = 0;
            for (int m = 0; m < SubsetNum; m++)
            {
                SUM = SUM + PROB[m];
                DimensionArray[m] = SUM;
            }
            DimensionArray[SubsetNum - 1] = 1;

            //roulette ball
            Random RandFun = new Random();
            double Ball = RandFun.NextDouble();

            //winner check
            int i = 0;
            do
            {
                if (Ball <= DimensionArray[i])
                    Subset = i;
                i++;
            } while (Subset == -1);
            return Subset;
        }

        public List<CommonVariables.Subset>
SubsetsReranking(CommonVariables.ProblemData ProblemData,
List<CommonVariables.CurrentInstance> ProjectInstances)
        {

            AuxiliaryFunctions Aux = new AuxiliaryFunctions();
            CommonVariables.Superproject SP = new
CommonVariables.Superproject();
            SP = ProblemData.SuperProject;
            for (int i = 0; i < SP.Subsets.Count; i++)
            {
                SP.Subsets[i].SubsetProjectsRanked = null;
                SP.Subsets[i].SubsetProjectsNum = 0;
            }

            for (int i = 0; i < ProjectInstances.Count; i++)
            {
                SP.Projects[i].Subset = ProjectInstances[i].Subset;
```

```
                SP.Subsets[ProjectInstances[i].Subset].SubsetProjectsNum++;
            }

            for (int i = 0; i < SP.Subsets.Count; i++)
            {
                SP.Subsets[i].SubsetProjectsRanked =
Aux.SubsetRankingFunction(SP, i);
            }
            ProblemData.SuperProject = SP;
            return SP.Subsets;

        }
        public void ResUse(ref int[,] ResUsage, CommonVariables.ProblemData
ProblemData, ref CommonVariables.CurrentInstance CurrentIn, int P)
        {
            CurrentIn.Penalty = 0;
            AuxiliaryFunctions Aux = new AuxiliaryFunctions();
            ALACO ActivityACO = new ALACO();
            int JobNum = ProblemData.SuperProject.Projects[P].Activities.Count;
            int[] duration = new int[JobNum];
            duration = Aux.Duration(ProblemData, CurrentIn);

            int[,] NonRenUsage = new
int[ProblemData.ResAvail.NonRenResAvailable.Count, JobNum + 1];
            int[,] RenUsage = new
int[ProblemData.ResAvail.RenResAvailable.Count, CurrentIn.Duration + 1];
            for (int U = 0; U < ProblemData.ResAvail.RenResAvailable.Count;
U++)
            {
                int MAX = 0;
                for (int d = 0; d < CurrentIn.Duration; d++)
                {
                    RenUsage[U, d] = ActivityACO.RResUsageMatrix(ProblemData,
CurrentIn, U, d, duration);
                    if (MAX < RenUsage[U, d])
                        MAX = RenUsage[U, d];
                }
                RenUsage[U, CurrentIn.Duration] = MAX;
                ResUsage[P, U] = RenUsage[U, CurrentIn.Duration];
            }
            for (int U = 0; U < ProblemData.ResAvail.NonRenResAvailable.Count;
U++)
            {
                int SUM = 0;
                for (int d = 0; d < JobNum; d++)
                {
                    NonRenUsage[U, d] =
ProblemData.SuperProject.Projects[CurrentIn.ProjectID].Activities[d].Modes[Curr
entIn.ModeList[d]].ResNeed.NonRenResNeed[U].ResMatrix[0];
                    SUM = SUM + NonRenUsage[U, d];
                }
                NonRenUsage[U, JobNum] = SUM;
                if (NonRenUsage[U, JobNum] >
ProblemData.SuperProject.Projects[CurrentIn.ProjectID].ResAvail.NonRenResAvaila
ble[U].ResMatrix[0])
                {
                    CurrentIn.Penalty = (NonRenUsage[U, JobNum] -
ProblemData.SuperProject.Projects[CurrentIn.ProjectID].ResAvail.NonRenResAvaila
ble[U].ResMatrix[0]) + CurrentIn.Penalty;
                }
                ResUsage[P, ProblemData.ResAvail.RenResAvailable.Count + U] =
NonRenUsage[U, JobNum];
```

```
            }
        }
        public void Transformation(CommonVariables.ProblemData ProblemData,
List<CommonVariables.CurrentInstance> ProjectInstances, ref
CommonVariables.Project SuperP, ref CommonVariables.CurrentInstance SuperIn,
int[,] ResNeed, List<CommonVariables.Subset> NewSubs)
        {
            #region Resource Allocation
            SuperP.ProjectID = ProjectInstances.Count;
            SuperP.ProjectName = "Hyper Project";
            SuperP.Subset = 0;
            SuperP.ResourceFactor = 0;
            SuperP.ResourceStrength = 0;
            SuperP.ResAvail = new CommonVariables.ResourcesAvailable();
            SuperP.ResAvail.RenResAvailable = new
List<CommonVariables.Resource>();
            SuperP.ResAvail.NonRenResAvailable = new
List<CommonVariables.Resource>();
            SuperP.ORNumber = 0;
            SuperP.Duration = 0;
            int RenResNum = ProblemData.ResAvail.RenResAvailable.Count;
            int NonRenResNum = ProblemData.ResAvail.NonRenResAvailable.Count;
            SuperP.ResourceS = new double[RenResNum + NonRenResNum];

            for (int i = 0; i < (RenResNum + NonRenResNum); i++)
            {
                CommonVariables.Resource Res = new CommonVariables.Resource();
                Res.Weight = 0;
                if (i < RenResNum)
                {
                    Res.ResID = i;
                    Res.ResMatrix = new int[1];
                    Res.ResMatrix[0] =
ProblemData.ResAvail.RenResAvailable[i].ResMatrix[0];
                    SuperP.ResAvail.RenResAvailable.Add(Res);
                }
                else
                {
                    Res.ResID = i - RenResNum;
                    Res.ResMatrix = new int[1];
                    Res.ResMatrix[0] =
ProblemData.ResAvail.NonRenResAvailable[i - RenResNum].ResMatrix[0];
                    SuperP.ResAvail.NonRenResAvailable.Add(Res);
                }
                SuperP.ResourceS[i] = 0;
            }
            #endregion

            #region Projects to Activities
            SuperP.Activities = new List<CommonVariables.Activity>();
            //Super Dummy Start Activity
            CommonVariables.Activity ActO = new CommonVariables.Activity();
            ActO.ActivityID = 0;
            ActO.ES = 0;
            ActO.EF = 0;
            ActO.LS = 0;
            ActO.LF = 0;
            ActO.Modes = new List<CommonVariables.Mode>();
            CommonVariables.Mode ActOMode = new CommonVariables.Mode();
            ActOMode.ModeID = 0;
            ActOMode.Duration = 0;
            ActOMode.ResNeed = new CommonVariables.ResourcesNeed();
```

```
            ActOMode.ResNeed.RenResNeed = new List<CommonVariables.Resource>();
            ActOMode.ResNeed.NonRenResNeed = new
List<CommonVariables.Resource>();
            for (int j = 0; j < (RenResNum + NonRenResNum); j++)
            {
                CommonVariables.Resource Res = new CommonVariables.Resource();
                Res.Weight = 0;
                if (j < RenResNum)
                {
                    Res.ResID = j;
                    Res.ResMatrix = new int[1];
                    Res.ResMatrix[0] = 0;
                    ActOMode.ResNeed.RenResNeed.Add(Res);
                }
                else
                {
                    Res.ResID = j - RenResNum;
                    Res.ResMatrix = new int[1];
                    Res.ResMatrix[0] = 0;
                    ActOMode.ResNeed.NonRenResNeed.Add(Res);
                }
            }
            ActO.Modes.Add(ActOMode);
            ActO.Successors = new int[ProjectInstances.Count + 1];
            for (int i = 0; i < ProjectInstances.Count; i++)
            {
                ActO.Successors[i] = ProjectInstances[i].ProjectID + 1;
            }
            ActO.Successors[ProjectInstances.Count] = ProjectInstances.Count +
1;
            SuperP.Activities.Add(ActO);

            //Projects --> Activities
            for (int i = 0; i < ProjectInstances.Count; i++)
            {
                CommonVariables.Activity Act = new CommonVariables.Activity();
                Act.ActivityID = i + 1;
                Act.ES = 0;
                Act.EF = 0;
                Act.LS = 0;
                Act.LF = 0;
                Act.Modes = new List<CommonVariables.Mode>();
                CommonVariables.Mode ActMode = new CommonVariables.Mode();
                ActMode.ModeID = ProjectInstances[i].Subset;
                ActMode.Duration = ProjectInstances[i].Duration;
                ActMode.ResNeed = new CommonVariables.ResourcesNeed();
                ActMode.ResNeed.RenResNeed = new
List<CommonVariables.Resource>();
                ActMode.ResNeed.NonRenResNeed = new
List<CommonVariables.Resource>();
                for (int j = 0; j < (RenResNum + NonRenResNum); j++)
                {
                    CommonVariables.Resource Res = new
CommonVariables.Resource();
                    Res.Weight = 0;
                    if (j < RenResNum)
                    {
                        Res.ResID = j;
                        Res.ResMatrix = new int[1];
                        Res.ResMatrix[0] = ResNeed[i, j];
                        ActMode.ResNeed.RenResNeed.Add(Res);
                    }
```

```
                    else
                    {
                        Res.ResID = j - RenResNum;
                        Res.ResMatrix = new int[1];
                        Res.ResMatrix[0] = ResNeed[i, j];
                        ActMode.ResNeed.NonRenResNeed.Add(Res);
                    }
                }
                Act.Modes.Add(ActMode);
                int SuccCounter = 0;
                for (int por = 0; por <
ProblemData.SuperProject.ProjectsOuterRelations.Count; por++)
                {
                    if
(ProblemData.SuperProject.ProjectsOuterRelations[por].SourceProj == i)
                    {
                        SuccCounter++;
                    }
                }
                Act.Successors = new int[SuccCounter + 1];
                int k = 0;
                for (int por = 0; por <
ProblemData.SuperProject.ProjectsOuterRelations.Count; por++)
                {
                    if
((ProblemData.SuperProject.ProjectsOuterRelations[por].SourceProj) == i)
                    {
                        Act.Successors[k] =
ProblemData.SuperProject.ProjectsOuterRelations[por].DestinationProj + 1;
                        k++;
                    }
                }
                Act.Successors[SuccCounter] = ProjectInstances.Count + 1;
                SuperP.Activities.Add(Act);
            }
            //Supersink Activity
            CommonVariables.Activity ActN = new CommonVariables.Activity();
            ActN.ActivityID = 0;
            ActN.ES = 0;
            ActN.EF = 0;
            ActN.LS = 0;
            ActN.LF = 0;
            ActN.Modes = new List<CommonVariables.Mode>();
            CommonVariables.Mode ActNMode = new CommonVariables.Mode();
            ActNMode.ModeID = 0;
            ActNMode.Duration = 0;
            ActNMode.ResNeed = new CommonVariables.ResourcesNeed();
            ActNMode.ResNeed.RenResNeed = new List<CommonVariables.Resource>();
            ActNMode.ResNeed.NonRenResNeed = new
List<CommonVariables.Resource>();
            for (int j = 0; j < (RenResNum + NonRenResNum); j++)
            {
                CommonVariables.Resource Res = new CommonVariables.Resource();
                Res.Weight = 0;
                if (j < RenResNum)
                {
                    Res.ResID = j;
                    Res.ResMatrix = new int[1];
                    Res.ResMatrix[0] = 0;
                    ActNMode.ResNeed.RenResNeed.Add(Res);
                }
                else
```

```
                {
                    Res.ResID = j - RenResNum;
                    Res.ResMatrix = new int[1];
                    Res.ResMatrix[0] = 0;
                    ActNMode.ResNeed.NonRenResNeed.Add(Res);
                }
            }
            ActN.Modes.Add(ActOMode);
            ActN.Successors = new int[0];
            SuperP.Activities.Add(ActN);
            #endregion

            #region ModeList & Activity List
            SuperIn.ProjectID = SuperP.ProjectID;
            SuperIn.Subset = SuperP.Subset;
            SuperIn.ModeList = new int[SuperP.Activities.Count];
            SuperIn.ActivityList = new int[SuperP.Activities.Count];
            for (int m = 0; m < SuperP.Activities.Count; m++)
            {
                SuperIn.ModeList[m] = 0;
            }
            int a = 0;
            SuperIn.ActivityList[a] = 0;
            a++;
            for (int i = ProblemData.SuperProject.Subsets.Count - 1; i > -1; i-
-)
            {
                for (int j = 0; j <
ProblemData.SuperProject.Subsets[i].SubsetProjectsRanked.Length; j++)
                {
                    SuperIn.ActivityList[a] =
ProblemData.SuperProject.Subsets[i].SubsetProjectsRanked[j] + 1;
                    a++;
                }
            }
            SuperIn.ActivityList[a] = a;
            #endregion

            ProblemData.SuperProject.Projects.Add(SuperP);
            SSGS SSGS_SP = new SSGS();
            SuperIn = SSGS_SP.SSGSFunction(ProblemData, SuperIn);

            #region Penalty
            int JobNum = SuperIn.ActivityList.Length;
            int[,] NonRenUsage = new
int[ProblemData.ResAvail.NonRenResAvailable.Count, JobNum + 1];

            for (int U = 0; U < ProblemData.ResAvail.NonRenResAvailable.Count;
U++)
            {
                int SUM = 0;
                for (int d = 0; d < JobNum; d++)
                {
                    NonRenUsage[U, d] =
ProblemData.SuperProject.Projects[SuperIn.ProjectID].Activities[d].Modes[SuperI
n.ModeList[d]].ResNeed.NonRenResNeed[U].ResMatrix[0];
                    SUM = SUM + NonRenUsage[U, d];
                }
                NonRenUsage[U, JobNum] = SUM;
                if (NonRenUsage[U, JobNum] >
ProblemData.SuperProject.Projects[SuperIn.ProjectID].ResAvail.NonRenResAvailabl
e[U].ResMatrix[0])
```

```
                {
                    SuperIn.Penalty = SuperIn.Penalty + (NonRenUsage[U, JobNum]
-
ProblemData.SuperProject.Projects[SuperIn.ProjectID].ResAvail.NonRenResAvailabl
e[U].ResMatrix[0]);
                }
            }
            #endregion

            //ALACO ACO = new ALACO();
            //SuperIn = ACO.ActivityListACO(ProblemData, SuperIn);

            ProblemData.SuperProject.Projects.Remove(SuperP);
        }

        public void IBFunction(double[] SolFitness, double[,] SolFitnessPerP,
List<CommonVariables.CurrentInstance> SuperInstances,
List<CommonVariables.Project> SuperPs,
List<List<CommonVariables.CurrentInstance>> ProjectInstaTot, ref
CommonVariables.CurrentInstance IBSuperInstance, ref CommonVariables.Project
IBSuperP, ref List<CommonVariables.CurrentInstance> IBProjectInstaTot, ref
List<CommonVariables.CurrentInstance> IBProjectInstaTot_2, ref double
IBSuperInFitness, ref double[] IBforProject_2)
        {
            int ProjNum = ProjectInstaTot[0].Count;
            int Marker = 0;
            for (int i = 0; i < SuperInstances.Count; i++)
            {
                if (SolFitness[i] < SolFitness[Marker])
                    Marker = i;
            }
            IBSuperInstance = SuperInstances[Marker];
            IBSuperP = SuperPs[Marker];
            IBSuperInFitness = SolFitness[Marker];
            IBProjectInstaTot = ProjectInstaTot[Marker];
            IBProjectInstaTot_2 = ProjectInstaTot[Marker];
            //_2 for Mode Optimization
            for (int j = 0; j < ProjNum; j++)
            {
                int InstaMarker = 0;
                for (int i = 0; i < ProjectInstaTot.Count; i++)
                {
                    if (SolFitnessPerP[i, j] < SolFitnessPerP[InstaMarker, j])
                        InstaMarker = i;
                }
                IBforProject_2[j] = SolFitnessPerP[InstaMarker, j];
                IBProjectInstaTot_2[j] = ProjectInstaTot[InstaMarker][j];
            }

        }

        public void ModeratorUpdateACO(ref CommonVariables.ProblemData
ProblemData, int Gens, double PenaltyWeightStart)
        {
            if (Gens > (ProblemData.MODACO.Generations * 3 / 4))
                ProblemData.MODACO.ρ = 0.075;
            ProblemData.MODACO.PenaltyWeight = -((double)1 /
(double)ProblemData.MODACO.Generations) * Gens + PenaltyWeightStart;
            if (ProblemData.MODACO.PenaltyWeight <= 0.5)
                ProblemData.MODACO.PenaltyWeight = 0.5;
        }
```

```
        public void ModeUpdateFunction(ref double[][][] τ,
CommonVariables.ProblemData ProblemData, List<CommonVariables.CurrentInstance>
BSProjectInstaTot_Tot_2, double[] BSforProject_Tot_2,
List<CommonVariables.CurrentInstance> IBProjectInstaTot_2, double[]
IBforProject_2)
        {
            int ProjectNum = IBProjectInstaTot_2.Count;
            for (int P = 0; P < ProjectNum; P++)
            {
                int JobNum = IBProjectInstaTot_2[0].ActivityList.Length;
                double BSFitnessValue = (double)1 / (double)2 /
BSforProject_Tot_2[P];
                double IBFitnessValue = (double)1 / (double)2 /
IBforProject_2[P];
                for (int i = 0; i < JobNum; i++)
                {
                    int ModeNum =
ProblemData.SuperProject.Projects[P].Activities[i].Modes.Count;
                    for (int j = 0; j < ModeNum; j++)
                    {
                        τ[P][i][j] = (1 - ProblemData.MODACO.ρ) * τ[P][i][j];
                    }
                }
                for (int i = 0; i < JobNum; i++)
                {
                    int ModeNum =
ProblemData.SuperProject.Projects[P].Activities[i].Modes.Count;
                    for (int j = 0; j < ModeNum; j++)
                    {
                        if (j == BSProjectInstaTot_Tot_2[P].ModeList[i])
                        {
                            τ[P][i][j] = τ[P][i][j] + ProblemData.MODACO.ρ *
BSFitnessValue;
                        }
                        if (j == IBProjectInstaTot_2[P].ModeList[i])
                        {
                            τ[P][i][j] = τ[P][i][j] + ProblemData.MODACO.ρ *
IBFitnessValue;
                        }
                        if (τ[P][i][j] < ProblemData.MODACO.ρ * BSFitnessValue)
                        {
                            τ[P][i][j] = ProblemData.MODACO.ρ * BSFitnessValue;
                        }
                        if (τ[P][i][j] > (BSFitnessValue /
ProblemData.MODACO.ρ))
                        {
                            τ[P][i][j] = BSFitnessValue / ProblemData.MODACO.ρ;
                        }
                    }
                }


            }
        }
        public void SubsetUpdateFunction(ref double[,] τSubset,
CommonVariables.ProblemData ProblemData, CommonVariables.Project BSSuperP_Tot,
CommonVariables.Project IBSuperP, CommonVariables.CurrentInstance
BSSuperInstance_Tot, double BSSuperInstance_Tot_Fitness,
CommonVariables.CurrentInstance IBSuperInstance, double
IBSuperInstance_Fitness)
        {
            int ProjNum = ProblemData.SuperProject.Projects.Count;
```

```
            int SubsetNum = ProblemData.SuperProject.Subsets.Count;
            double BSFitnessValue = (double)1 / (double)2 /
BSSuperInstance_Tot_Fitness;
            double IBFitnessValue = (double)1 / (double)2 /
IBSuperInstance_Fitness;

            for (int i = 0; i < ProjNum; i++)
            {
                for (int j = 0; j < SubsetNum; j++)
                {
                    τSubset[i, j] = τSubset[i, j] * (1 - ProblemData.MODACO.ρ);
                }
            }

            for (int i = 0; i < ProjNum; i++)
            {
                for (int j = 0; j < SubsetNum; j++)
                {
                    if (BSSuperP_Tot.Activities[i + 1].Modes[0].ModeID == j)
                    {
                        τSubset[i, j] = τSubset[i, j] + (BSFitnessValue *
ProblemData.MODACO.ρ);
                    }
                    if (IBSuperP.Activities[i + 1].Modes[0].ModeID == j)
                    {
                        τSubset[i, j] = τSubset[i, j] + (IBFitnessValue *
ProblemData.MODACO.ρ);
                    }
                    if (τSubset[i, j] < BSFitnessValue * ProblemData.MODACO.ρ)
                    {
                        τSubset[i, j] = BSFitnessValue * ProblemData.MODACO.ρ;
                    }
                    if (τSubset[i, j] > (BSFitnessValue /
ProblemData.MODACO.ρ))
                    {
                        τSubset[i, j] = BSFitnessValue / ProblemData.MODACO.ρ;
                    }
                }
            }
        }

        Random RandFun = new Random();
    }
}
```