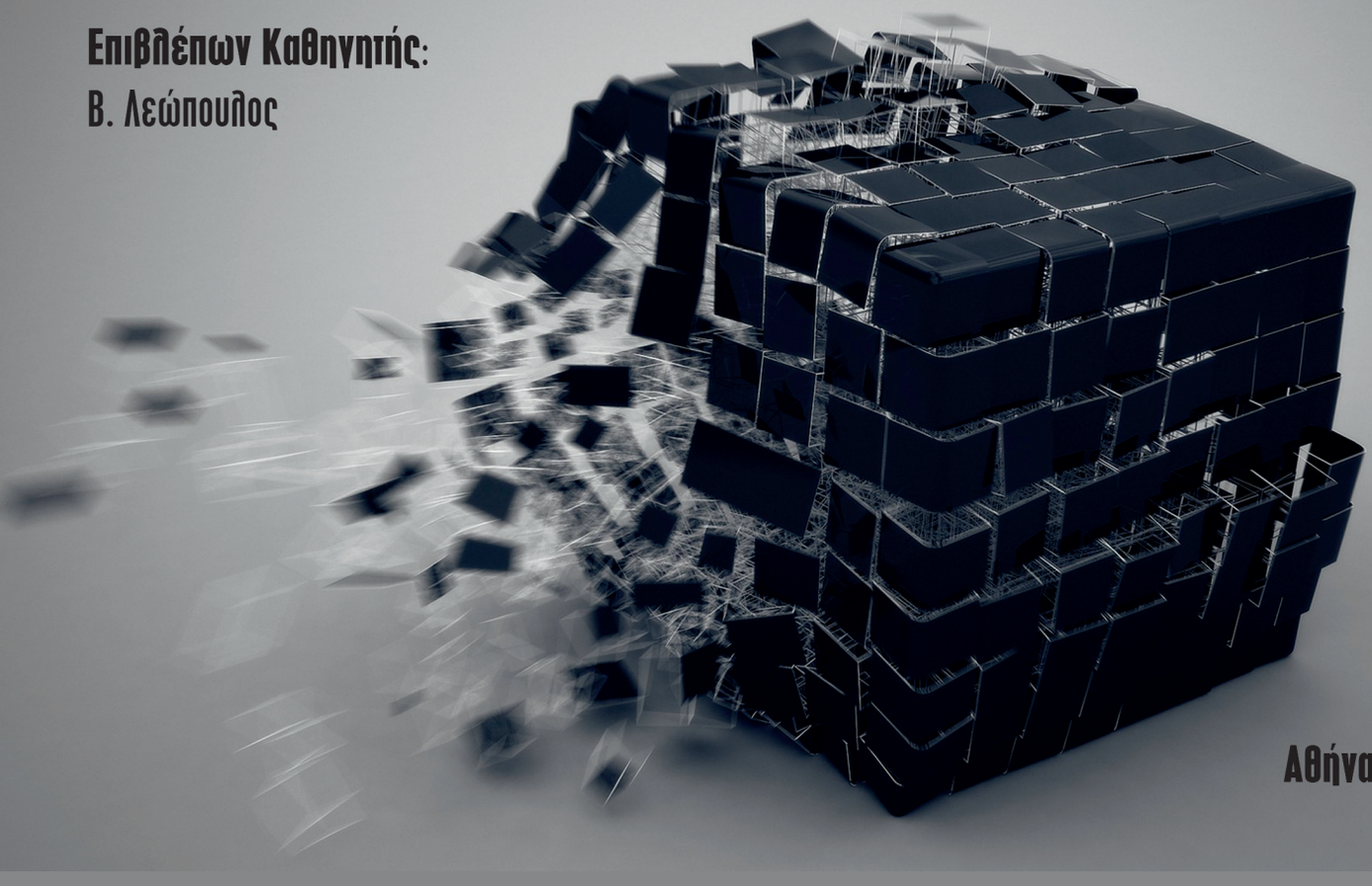
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ & ΕΠΙΧΕΙΡΗΣΙΑΚΗΣ ΕΡΕΥΝΑΣ

Διπλωματική Εργασία

Προγραμματισμός Έργων Προκαθορισμένης Προσπάθειας με Ευέλικτα Προφίλ Πόρων και Γενικευμένες Σχέσεις Προτεραιότητας

Ανδρέας Ι. Γεωργόπουλος

Επιβλέπων Καθηγητής:
Β. Λεώπουλος



Αθήνα 2014

Προγραμματισμός Έργων Προκαθορισμένης Προσπάθειας με Ευέλικτα Προφίλ Πόρων και Γενικευμένες Σχέσεις Προτεραιότητας

Ανδρέας Ι. Γεωργόπουλος

Αθήνα *2014*

"Στις κορυφές των βουνών..."

- *Β.Θ*

Δήλωση περί μη λογοκλοπής

Ο Ανδρέας Ι. Γεωργόπουλος, γνωρίζοντας τις συνέπειες της λογοκλοπής, δηλώνω υπεύθυνα ότι η παρούσα διπλωματική εργασία με τίτλο *«Προγραμματισμός Έργων Προκαθορισμένης Προσπάθειας με Ευέλικτα Προφίλ Πόρων και Γενικευμένες Σχέσεις Προτεραιότητας»* αποτελεί προϊόν αυστηρά προσωπικής εργασίας και όλες οι πηγές που έχω χρησιμοποιήσει έχουν δηλωθεί κατάλληλα στις βιβλιογραφικές παραπομπές και αναφορές. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Σημειώνεται ότι η χρησιμοποιούμενη μορφή του σχεδίου του κειμένου βασίζεται στο πρότυπο των Patrik Further και Pedro De Almeida Patrik.

Ο δηλών,
Α.Γεωργόπουλος

Ευχαριστίες

Αναπολώντας τα πρόσωπα που μου δίδαξαν να λέω «ευχαριστώ» με μεγαλοψυχία και γενναιοδωρία, τους γονείς μου, θα ήθελα να εστιάσω την προσοχή σας στους ανθρώπους που οφείλω τη σύναψη της παρούσας διπλωματικής εργασίας. Από καρδιάς και δίχως ίχνη φαινοτυπικής ανιδιοτέλειας, δηλώνω ευγνώμων στην οικογένειά Γεωργόπουλος που αποτελούν αρωγοί κάθε εγχειρήματός μου, σε όλους τους επιστήθιους φίλους και συναδέλφους για την υπομονή και την εφάμιλλη διάθεση, σε όλους τους καθηγητές στο Εθνικό Μετσόβιο Πολυτεχνείο, στον επιβλέποντα καθηγητή κ. Β. Λεώπουλο για τη στήριξη που παρήχε, στον επίκουρο καθηγητή κ. Κ. Κηρυττόπουλο και στους υποψήφιους διδάκτορες κ. Κ. Χατζόγλου, κ. Ε. Δερμιτζάκη και κ. Ν. Ντάλλα για τη βοήθεια που προσέφεραν οποτεδήποτε κλήθηκαν και τέλος, στην κ. Δρ. Ε. Ρόκου που αποτελεί την γενεσιουργό αιτία της παρούσας εργασίας και μια αέναη πηγή έμπνευσης, στήριξης, βοήθειας και ανιδιοτελούς προσφοράς, χάρη στην υπομονή και επιμονή της οποίας, με περηφάνεια παρουσιάζω την διπλωματική μου εργασία. Μια εργασία που ολοκληρώνει ένα συναρπαστικό πενταετή κύκλο σπουδών στο Εθνικό Μετσόβιο Πολυτεχνείο. Τους ευχαριστώ όλους και ευελπιστώ να κριθώ αντάξιος των προσδοκιών τους. Ευχαριστώ!

Έποψη

Στην παρούσα διπλωματική εργασία παρουσιάζεται ένα νέο εννοιολογικό και μαθηματικό μοντέλο, το οποίο διαχειρίζεται προβλήματα προγραμματισμού έργων υπό περιορισμένους ανανεώσιμους πόρους με ή χωρίς γενικευμένες σχέσεις προτεραιότητας με ελάχιστες και μέγιστες χρονικές καθυστερήσεις, μέσα από την παραγωγή ευέλικτων προφίλ πόρων που ικανοποιούν την προκαθορισμένη απαιτούμενη προσπάθεια κάθε δραστηριότητας του έργου (*Effort Driven Resource Constrained Project Scheduling with Flexible Resource Profiles and Generilized Precedecne Relations with minimal and maximal Time Lags*).

Η κύρια ιδέα έγκειται στη δυνατότητα χρονοπρογραμματισμού και υπολογισμού της κατανομής των πόρων, μέσω της παραγωγής ευέλικτων προφίλ εκτέλεσης που βελτιστοποιούν τον αντικειμενικό σκοπό του έργου, με ταυτόχρονο σεβασμό στις απαιτήσεις και στους ρεαλιστικά διατυπωμένους περιορισμούς. Προτείνεται ένας σύνθετος εξελικτικός αλγόριθμος, ο οποίος υλοποιεί τις απαιτήσεις, τους περιορισμούς και τους στόχους του νέου μοντέλου. Πρόκειται για μια μεταευρετική υπολογιστική διαδικασία βελτιστοποίησης, που βασίζεται στην εξελικτική δυνατότητα δύο εμφωλευμένων γενετικών αλγορίθμων. Ο εξωτερικός γενετικός αλγόριθμος διαχειρίζεται τη δημιουργία προφίλ εκτέλεσης ανά δραστηριότητα και απαιτούμενο πόρο, ενώ ο εσωτερικός γενετικός αλγόριθμος διαχειρίζεται την αναζήτηση του βέλτιστου χρονοπρογράμματος στο χώρο λύσεων του προβλήματος και κατ' επέκταση τη βελτιστοποίηση της συνολικής διάρκειας του έργου. Ο χρονοπρογραμματισμός υλοποιείται μέσα από την ευρετική μέθοδο παραγωγής χρονοπρογραμμάτων με βήμα αποπρογραμματισμού (*serial Schedule Generation Scheme with Unschedule step – sSGSU*). Κατά το πέρας της υπολογιστικής διαδικασίας, δίδονται πέραν από το χρονοπρόγραμμα με τις χρονικές στιγμές έναρξης των δραστηριοτήτων του έργου, οι διάρκειες και τα προφίλ εκτέλεσης των δραστηριοτήτων ανά απαιτούμενο πόρο και χρονική περίοδο.

Η επαλήθευση και η αξιολόγηση της αποδοτικότητας και της αποτελεσματικότητας του προτεινόμενου αλγορίθμου επίλυσης απεδείχθη μέσα από ένα σύνολο πειραματικών διαδικασιών, οι οποίες αφορούν τη σύγκριση των αποτελεσμάτων που δίδει η προτεινόμενη μέθοδος επίλυσης με τα αντίστοιχα δεδομένα βέλτιστα αποτελέσματα, διαθέσιμα στη διεθνή βιβλιογραφία, για κάθε επιμέρους πρόβλημα προγραμματισμού που αφορά το μοντέλο του προβλήματος, δηλαδή RCPSP και RCPSP/max προβλήματα. Επιπρόσθετα, οι πειραματικές διαδικασίες περιλαμβάνουν τη δημιουργία και την επίλυση παραδειγμάτων για το υπό μελέτη πρόβλημα, καθώς και προσπάθειες καθορισμού των τιμών των παραμέτρων του προτεινόμενου αλγορίθμου επίλυσης που βελτιστοποιούν την απόδοση του τελευταίου. Τα πειραματικά αποτελέσματα σκιαγραφούν τη δυναμική και τις προοπτικές που διακατέχουν το προτεινόμενο μοντέλο, το οποίο επιλύει αποτελεσματικά όλα τα προβλήματα προγραμματισμού έργων που δύναται να διαχειριστεί, με δυνατότητα επίλυσης ακόμα και άλυτων, ως τώρα, προβλημάτων.

Abstract

In this diploma thesis a new mathematical formulation for the effort driven resource constrained project scheduling problem with renewable resources with or without generalized precedence relations with minimal and maximal time lags by generating flexible resource profiles is proposed.

The main idea is to generate such flexible resource profiles for each project's activity, that would optimise the project's objective with respect to the problem's constaints. Based on this model and its constraints, requirements and objectives, an evolutionary algorithm is implemented. The proposed algorithm is a metaheuristic combinatorial optimization solution process, consisting of two nested genetic algorithms. The external genetic algorithm generates resource profiles per activity and required resource, while the internal genetic algorithm handles the optimization of the overall project's duration by searching the best schedule in the problem's solution space for the corresponding external chromosome with specified resource profiles. The scheduling part of the internal genetic algorithm is implemented by the heuristic method of serial schedule generation scheme with unschedule step (*sSGSU*). The outputs of the solution process are the scheduled activities with their start times and their total durations and the activities' resource profiles per required resource and time period.

The efficiency and the evaluation of the proposed solution algorithm is substantiated by a set of experimental procedures. The experiments that took place regard firstly a comparison of the best known results for each variation and extention of RCPSP that are incorporated in the model to the results given by the proposed algorithm and secondly the design of specific instances for the under study problem. The experimental results outline the dynamic and the perspectives of the proposed model and algorithm, which has the ability not only to solve effectively RCPSP and RCPSP/max problems, but also to provide solutions to yet unsolved problems.

Πίνακας Περιεχομένων

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Αλγορίθμων

1
Εισαγωγή

Ο χρονικός Προγραμματισμός Έργων (*Project Scheduling*) αποτελεί ένα πρόβλημα υψηλής πολυπλοκότητας, όπου κάθε διαχειριστής (*project manager*) καλείται να διαχειριστεί στο αρχικό στάδιο ενός έργου, ανεξαρτήτου φύσης και αντικειμένου του τελευταίου, ενώ παράλληλα καθιστά έναν από τους πιο κρίσιμους παράγοντες επιτυχίας του έργου, μιας και θεωρείται η πιο κοινή αιτία αποτυχίας του (Goldratt, 1997, Lewis, 1995). Αναντίρρητα, δεν αποτελεί έκπληξη το γεγονός ότι η ελαχιστοποίηση της συνολικής διάρκειας ενός έργου (ή κάποιας άλλης αντικειμενικής συνάρτησης), συναρτήσει περιορισμών πόρων και σχέσεων προτεραιότητας μεταξύ των δραστηριοτήτων του, αποτελεί έναν από τους πρωταρχικούς στόχους στη Διαχείριση Έργων (Demeulemeester and Herroelen, 2002).

Ο χρονοπρογραμματισμός έργων στοχεύει στη δημιουργία ενός εφικτού χρονοπρογράμματος που ελαχιστοποιεί το κριτήριο απόδοσης του προγραμματισμού, με σεβασμό στους περιορισμούς που αφορούν τόσο τη χρήση και τη διαθεσιμότητα των πόρων, όσο και τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων του έργου. Το παραχθέν χρονοπρόγραμμα, αποτελεί το πρόγραμμα αναφοράς, βάσει του οποίου θα εκτελεστεί το έργο, θα κατανεμηθούν οι πόροι, θα προγραμματιστούν εξωτερικές και εσωτερικές δραστηριότητες της εφοδιαστικής αλυσίδας του οργανισμού που αναλαμβάνει το έργο, θα γίνει η σύναψη των απαραίτητων συμβάσεων και θα καθοριστούν τα χρονικά όρια παράδοσης και προθεσμιών των αποτελεσμάτων του έργου. (Herroelen and Leus, 2005).

Πρακτικές εφαρμογές του χρονοπρογραμματισμού έργων, λαμβάνουν χώρα σε διαφορετικών ειδών βιομηχανίες, ενώ παράλληλα αποτελεί ένα συνεχώς αυξανόμενης σημαντικότητας πεδίο, για *make-to-order* εταιρείες (Brucker et al., 1998). Εν αντιστοιχία με τον επιχειρηματικό κόσμο, ο Προγραμματισμός Έργων αποτελεί ένα ελκυστικό ερευνητικό πεδίο ενδιαφέροντος, καθώς η μεγάλη ποικιλία διαφορετικών εκφάνσεων του ίδιου προβλήματος και διαφορετικών αντικειμένων, σε συνδυασμό με τη δυσκολία επίλυσης αυτόνομων και συνδυαστικών προβλημάτων, προσελκύει ολοένα και περισσότερους ερευνητές. Ένα από τα σημαντικότερα προβλήματα προγραμματισμού έργων αποτελεί το πρόβλημα του Προγραμματισμού Έργων υπό Περιορισμένους Πόρους, γνωστό στη διεθνή βιβλιογραφία με το ακρωνύμιο RCPSP (*Resource Constrained Project Scheduling Problem*). Στην κλασική του μορφή, έγκειται στον προγραμματισμό των δραστηριοτήτων ενός έργου με περιορισμούς σχέσεων προτεραιότητας τέλους-αρχής μεταξύ των δραστηριοτήτων με μηδενικές χρονικές καθυστερήσεις, καθώς και περιορισμούς σταθερής διαθεσιμότητας ανανεώσιμων πόρων, με κύριο στόχο την ελαχιστοποίηση της συνολική διάρκειας του έργου (*project makespan*) (Herroelen et al., 1998). Η έκβαση ενός επιλύσιμου προβλήματος, παρέχει ένα χρονοπρόγραμμα που δηλώνει τον ακριβή χρόνο που θα πραγματοποιηθεί κάθε δραστηριότητα του έργου.

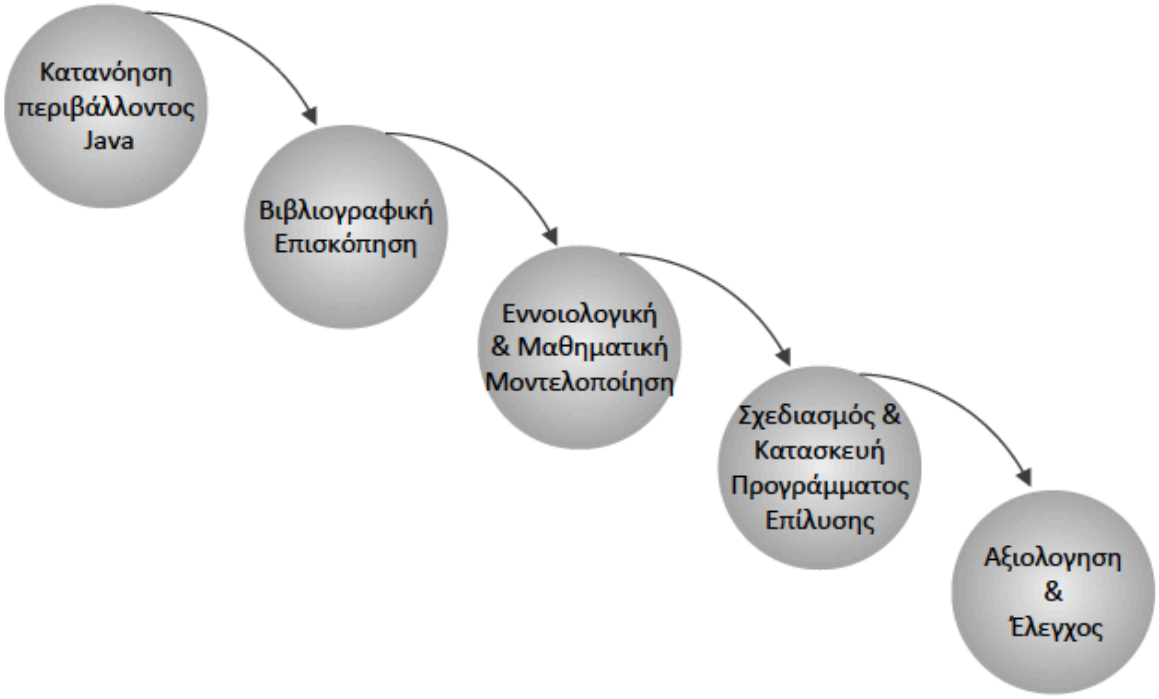
Στην παρούσα διπλωματική εργασία παρουσιάζεται ένα νέο εννοιολογικό και μαθηματικό μοντέλο, το οποίο διαχειρίζεται προβλήματα προγραμματισμού έργων υπό περιορισμένους ανανεώσιμους πόρους με ή χωρίς γενικευμένες σχέσεις προτεραιότητας με ελάχιστες και μέγιστες χρονικές καθυστερήσεις, μέσα από την παραγωγή ευέλικτων προφίλ πόρων που ικανοποιούν την προκαθορισμένη απαιτούμενη προσπάθεια κάθε δραστηριότητας του έργου. Η κύρια ιδέα έγκειται στη δυνατότητα υπολογισμού της βέλτιστης κατανομής πόρων μεταβλητής ποσότητας ανά χρονική περίοδο, με παραγωγή ευέλικτων προφίλ εκτέλεσης για κάθε δραστηριότητα που θα βελτιστοποιούν τον αντικειμενικό σκοπό του έργου, με ταυτόχρονο σεβασμό στις απαιτήσεις και στους ρεαλιστικά διατυπωμένους περιορισμούς. Εν συνεχεία, προτείνεται ένας σύνθετος εξελικτικός αλγόριθμος, ο οποίος αποτελεί μια μεταευρετική υπολογιστική διαδικασία βελτιστοποίησης, που βασίζεται στην εξελικτική δυνατότητα δύο εμφωλευμένων γενετικών αλγορίθμων. Ο εξωτερικός γενετικός αλγόριθμος διαχειρίζεται τη δημιουργία προφίλ εκτέλεσης ανά δραστηριότητα και απαιτούμενο πόρο, ενώ ο εσωτερικός γενετικός πραγματοποιεί βελτιστοποίηση χρόνου, μέσα από την παραγωγή του βέλτιστου χρονοπρογράμματος για τα δεδομένα προφίλ εκτέλεσης του εξωτερικού γενετικού αλγορίθμου.

Με αυτό τον τρόπο, ο διαχειριστής του έργου δεν χρειάζεται να εκτιμά το εκάστοτε προφίλ κατανομής με την απαιτούμενη ποσότητα πόρων και την αντίστοιχη διάρκεια εκτέλεσης κάθε δραστηριότητας. Αντίθετα, εκτιμά την απαιτούμενη προσπάθεια (*effort*) κάθε δραστηριότητας ανά απαιτούμενο πόρο και μέσω του παραγόμενου χρονοπρογραμμάτος της υπολογιστικής διαδικασίας λαμβάνει τόσο τη συνολική διάρκεια του έργου, όσο και το προφίλ εκτέλεσης κάθε δραστηριότητας. Τα παραχθέντα προφίλ εκτέλεσης των δραστηριοτήτων σκιαγραφούν την κατανεμημένη ποσότητα του κάθε πόρου ανά χρονική περίοδο και τη συνολική διάρκεια εκτέλεσης της κάθε δραστηριότητας του έργου.

Η διαδικασία που ακολουθήθηκε κατά τη διάρκεια εκπόνησης και ολοκλήρωσης της παρούσας διπλωματικής εργασίας, παρουσιάζεται στο Σχήμα 1. Ειδικότερα, εν αρχή έλαβε χώρα η μελέτη, η κατανόηση και η μάθηση του προγραμματιστικού περιβάλλοντος της Java, το οποίο επιλέχθηκε ως το περιβάλλον προγραμματισμού των αλγορίθμων επίλυσης του υπό μελέτη προβλήματος της παρούσας εργασίας. Έπειτα, ακολούθησε μια εκτεταμένη βιβλιογραφική επισκόπηση επί του προγραμματισμού έργων, δίδοντας ιδιαίτερη έμφαση σε επιμέρους γνωστικά πεδία με άμεση συνάφεια με το αντικείμενο της έρευνας της διπλωματικής μελέτης. Πιο συγκεκριμένα, εντοπίστηκαν οι προσεγγίσεις όλων των αντικειμένων που πραγματεύεται η διπλωματική εργασία στη διεθνή βιβλιογραφία, οι προτεινόμενες διαδικασίες επίλυσης κάθε μιας εξ αυτών, καθώς και οι συγκριτικές μελέτες των τρόπων και μεθόδων επίλυσης των υφιστάμενων αλγορίθμων. Παράλληλα, ορίστηκαν όλα τα επιμέρους συστατικά στοιχεία που θα συνθέσουν το υπό μελέτη πρόβλημα, με απώτερο στόχο τη λογική αιτίαση και συνύπαρξη αυτών αναφορικά με τα υφιστάμενα ρεαλιστικά προβλήματα που ταλανίζουν τόσο τον επιστημονικό όσο και τον πρακτικό χώρο του προγραμματισμού έργων.

Ύστερα, πραγματοποιήθηκε ο ορισμός του υπό μελέτη προβλήματος μέσα από την εννοιολογική και τη μαθηματική μοντελοποίηση όλων των δεδομένων, των περιορισμών και των στόχων, με την κατά το δυνατόν καλύτερη απεικόνιση της πραγματικότητας. Με δεδομένο το υπό μελέτη πρόβλημα, καθώς και όλων των παραμέτρων και στόχων αυτού, διεξήχθη η διαδικασία σχεδιασμού και κατασκευής του αλγορίθμου επίλυσης του προβλήματος. Τέλος, μέσα από παρατεταμένες πειραματικές διαδικασίες, κάνοντας χρήση ενός συνόλου από διαθέσιμα πειραματικά προβλήματα για τη συγκριτική μελέτη των διαδικασιών επίλυσης που έχουν προταθεί στον επιστημονικό χώρο, πραγματοποιήθηκε η επιλογή των τιμών των αρχικών παραμέτρων του αλγορίθμου επίλυσης. Παράλληλα έλαβε

χώρα η συγκριτική αξιολόγηση των αποτελεσμάτων της προτεινόμενης διαδικασίας επίλυσης με τα αντίστοιχα διαθέσιμα βέλτιστα αποτελέσματα των πειραματικών προβλημάτων, με στόχο την επιβεβαίωση της εγκυρότητας και της αποδοτικότητας του αλγορίθμου και την καθιέρωσή του στον επιστημονικό χώρο.



Σχήμα 1. Διαδικασία μεθόδου έρευνας

Ακολουθώντας την άνωθι διαδικασία και βάσει των πεπραχθέντων, προκύπτει η διάρθρωση της παρούσας διπλωματικής εργασίας, η οποία συνοψίζεται ως εξής:

- Στο τρέχον κεφάλαιο, δίδονται εισαγωγικά στοιχεία του αντικειμένου της εύρενας, καθώς και των γενικών στοιχείων αυτής. Παράλληλα παρουσιάζεται η μέθοδος έρευνας που ακολουθήθηκε για την προσέγγιση του υπό μελέτη προβλήματος και την υλοποίηση της διπλωματικής εργασίας.
- Στο κεφάλαιο 2, λαμβάνει χώρα μια εκτενής βιβλιογραφική επισκόπηση όλων των επιστημονικών πεδίων με άμεση συσχέτιση με το αντικείμενο της ερευνητικής μελέτης.
- Στο κεφάλαιο 3, σκιαγραφείται το ευρύτερο περιβάλλον του υπό μελέτη προβλήματος, ο ορισμός των παραμέτρων, των περιορισμών και των στόχων αυτού, καθώς και η ανάλυση των στοιχείων που το απαρτίζουν.
- Στο κεφάλαιο 4, παρουσιάζεται η ενοιολογική και μαθηματική μοντελοποίηση του υπό μελέτη προβλήματος.
- Στο κεφάλαιο 5, πραγματοποιείται αναλυτική περιγραφή της διαδικασίας επίλυσης, καθώς και της δομής και της λειτουργίας του προτεινόμενου αλγορίθμου επίλυσης.
- Στο κεφάλαιο 6, αναπτύσσονται οι πειραματικές διαδικασίες επαλήθευσης για την αξιολόγηση του προτεινόμενου αλγορίθμου, καθώς και τα αποτελέσματα αυτών.
- Στο Κεφάλαιο 7, συνοψίζονται γενικά συμπεράσματα που εξήχθησαν από όλη την ερευνητική μελέτη, καθώς και κατευθύνσεις μελλοντικής έρευνας.

2

Βιβλιογραφική Επισκόπηση

2.1 Διαχείριση Έργων

Στο σύγχρονο ανταγωνιστικό επιχειρηματικό περιβάλλον, η υλοποίηση και η προσφορά προϊόντων και υπηρεσιών με συγκεκριμένα πρότυπα χαρακτηριστικά, καθορισμένες προδιαγραφές ποιότητας, χαμηλό κόστος και εντός συμφωνηθέντων, ενδεχομένως, χρονικών ορίων, αποτελούν το στόχο κάθε οργανισμού. Ο κοινός παρανομαστής κάθε σύγχρονης επιχείρησης έγκειται στη δυνατότητα της αλλαγής και στην προσαρμοστικότητα. Αλλαγή τόσο στον τρόπο υλοποίησης, όσο και στην προσφερόμενη εκροή, ενώ προσαρμοστικότητα στις αλλαγές του ευρύτερου περιβάλλοντος που ενδιαφέρει τον οργανισμό. «*Οι πιο επιτυχημένοι εμπορικοί οργανισμοί είναι όσοι αρίστευσαν στις αλλαγές*» (Maylor, 2003). Η απόδοση παγκόσμιας κλάσης δύναται να επιτευχθεί μέσω της ανάπτυξης άριστου μάνατζμεντ, σημαντικό τμήμα του οποίου αποτελεί η Διαχείριση Έργων (*Project Management*). Το γεγονός αυτό, σκιαγραφεί τη λογική αιτίαση της σημαντικότητας που διακρίνει τη Διαχείριση Έργων.

Τι είναι, όμως, ένα έργο και ποιά τα χαρακτηριστικά που το διέπουν;

Ορισμός & Χαρακτηριστικά ενός Έργου

Ως έργο ορίζεται, σύμφωνα με το ISO, «*μια μοναδική δέσμη συντονισμένων και ελεγχόμενων δραστηριοτήτων με διακριτό σημείο έναρξης και λήξης, που λαμβάνουν χώρα για την επίτευξη του σκοπού του έργου που περιλαμβάνει συγκεκριμένα παραδοτέα σε συμφωνία με συγκεκριμένες απαιτήσεις, συμπεριλαμβανομένων στις τελευταίες και πολλαπλούς περιορισμούς, όπως χρόνου, κόστους και πόρων*» (ISO, 2012). Ένα έργο αποτελεί μια προσωρινή συντονισμένη προσπάθεια σχεδιασμού ενός νέου μοναδικού προϊόντος, υπηρεσίας ή ενός γενικότερου αποτελέσματος υπό τη φύση της εκροής, που με τη χρήση συγκεκριμένων πόρων ολοκληρώνουν έναν αντικειμενικό σκοπό, σε περιορισμένο χρόνο, με συγκεκριμένα κεφάλαια και με καθορισμένες προδιαγραφές ποιότητας (PMI, 2013). Η έκβαση ενός έργου μπορεί να αφορά ένα τελικό ή ημιέτοιμο προϊόν, μια υπηρεσία ή τη δυνατότητα παροχής αυτής, μια βελτίωση σε ένα υπάρχον σύστημα παραγωγής ή εξυπηρέτησης, ένα αποτέλεσμα, όπως ένα συμπέρασμα ενός ερευνητικού έργου ή ένα συγκεκριμένο αρχείο. Τέλος, η έκβαση του έργου μπορεί να είναι αισθητή ή μη.

Ένα έργο διακρίνεται από την προσωρινή και την μοναδική του φύση. Η προσωρινή φύση του, υποδεικνύει τη σαφή έναρξη και την περάτωσή του. Η πραγμάτωση της τελευταίας προκύπτει όταν οι στόχοι του έργου έχουν υλοποιηθεί ή όταν οι στόχοι δεν γίνεται να πραγματοποιηθούν είτε βραχυπρόθεσμα είτε μακροπρόθεσμα ή, τέλος, όταν η ανάγκη για την

ολοκλήρωση και την υλοποίηση του έργου πάψει να υφίσταται. Επιπρόσθετα, η περάτωση ενός έργου μπορεί να λάβει χώρα, σε περίπτωσεις όπου ο πελάτης –αιτών για πραγματοποίηση και αποδέκτης του αποτελέσματος- ζητήσει για οποιοδήποτε λόγο την διακοπή του (PMI, 2013). Η μοναδικότητά του, υποδεικνύει ότι παρότι κάποιοι παράγοντες επαναλαμβάνονται σε διαφορετικών έργων παραδοτέα και δραστηριότητες, κάθε έργο έχει τα ιδιαίτερα χαρακτηριστικά που του δίνουν αυτή την ιδιότητα. Επί παραδείγματι, οι κατασκευές κτιρίων μπορούν να πραγματοποιηθούν με τα ίδια ή παρόμοια υλικά και με τις ίδιες ή διαφορετικές ομάδες έργου. Πάραυτα, κάθε έργο κατασκευής κτιρίου παραμένει μοναδικό με ξεχωριστή τοποθεσία και σχεδιασμό, διαφορετικές υπάρχουσες καταστάσεις, διαφορετικούς ενδιαφερόμενους, κ.ό.κ.

Όλα τα έργα, έχουν κάποια κοινά χαρακτηριστικά τα οποία συνοψίζονται ως εξής (de Vilder, 2004):

- Αντικειμενικό σκοπό:

  Τα έργα έχουν σαφώς καθορισμένους στόχους και υλοποιούνται για την παραγωγή σαφώς καθορισμένων αποτελεσμάτων. Σκοπός τους είναι να επιλύσουν ένα πρόβλημα, να βελτιώσουν μια υφαστάμενη κατάσταση ή να ικανοποιήσουν μια ανάγκη, μέσα από ένα ορισμένο, και σύμφωνα με τις εκάστοτε απαιτήσεις, παραγόμενο αποτέλεσμα.

- Μοναδική & Προσωρινή υπόσταση :

  όπως σκιαγραφείται και στην άνωθι παράγραφο, όλα τα έργα είναι μοναδικά και προσωρινά, καθώς παρέχουν μια συγκεκριμένη ανάδραση στην εκάστοτε γενεσιουργό τους ανάγκη, υλοποιούνται σε ένα συγκεκριμένο περιβάλλον εφαρμογής με μοναδικές συνθήκες, ενώ παράλληλα έχουν σαφή σημεία έναρξης και λήξης, γεγονός που τα καθιστά περιορισμένα χρονικά και χωρικά.

- Πολυπλοκότητα:

  Η υλοποίηση των έργων έγκειται στην ικανότητα πολλαπλού σχεδιασμού και εφαρμογής δεξιοτήτων, καθώς και στη συνεργασία πολλών ανθρώπων με διαφορετικά συμφέροντα. Αβίαστα, η πολυπλοκότητα ενός έργου, ανεξαρτήτου δυσκολίας πραγματοποίησης των στόχων του, υπάρχει λόγω της φύσης των σχέσεων των εργασιών και των ατόμων που σχετίζονται με αυτό.

- Συλλογικότητα:

  Κάθε έργο είναι αποτέλεσμα συλλογικής δράσης, σχετίζει διαφορετικές ομάδες ενδιαφερομένων και ικανοποιεί ανάγκες τρίτων.

- Στοχαστικότητα:

  Το στοιχείο της στοχαστικότητας αφορά τη φύση του προγραμματισμού κάθε έργου, καθώς γίνονται παραδοχές και εκτιμήσεις για την υλοποίηση του. Χαρακτηριστικό στοιχείο στοχαστικότητας του έργου είναι οι διάρκειες σε επιμέρους δραστηριότητες του προς επίτευξη τελικού αποτελέσματος.

- Αβεβαιότητα:

  Σε κάθε έργο εγκυμονούν κίνδυνοι και το στοιχείο της αβεβαιότητας είναι παρόν, δεδομένου του προγραμματισμού των δράσεων πριν την έναρξη της υλοποίησής του και της στοχαστικότητας.

- Κύκλος ζωής:

  Τα έργα έχουν διακριτά και αναγνωρίσιμα στάδια από τα οποία υπεισέρχονται, ανεξάρτητα από τη φύση και τους στόχους τους. Τα στάδια αυτά συνοψίζονται ως εξής:

  1. *Έναρξης:*
     Με τη γένεση της ανάγκης υλοποίησης του έργου, γίνεται η πρώτη διαμόρφωσή του, με τον καθορισμό των αναγκών των χρηστών, το στρατηγικό σχεδιασμό της φάσης της υλοποίησης και τις μελέτες σκοπιμότητας. Όλα τα προηγούμενα έχουν στόχο την έγκριση του έργου και τη μετάβασή του στο επόμενο στάδιο του σχεδιασμού.
  2. *Σχεδιασμού:*
     Με την είσοδο ενός εγκεκριμένου έργου στη φάση του σχεδιασμού, λαμβάνει χώρα ο βασικός σχεδιασμός και ο οικονομικός υπολογισμός του κόστους των επιμέρους συνιστωσών για την υλοποίησή του, η σύνταξη των χρονοδιαγραμμάτων, ενώ παράλληλα ορίζεται η ομάδα υλοποίησής του και καθορίζονται οι όροι της σύμβασης, αν υπάρχει.
  3. *Εκτέλεσης:*
     Κατά το στάδιο της εκτέλεσης του έργου, κυριαρχούν προεργασίες εργασίες, δοκιμές και αναθεωρήσεις με στόχο την υλοποίησή του σύμφωνα με τις προσυμφωνηθείσες απαιτήσεις. Με τεχνικές προγραμματισμού, παρακολούθησης και ελέγχου κάθε επιμέρους δραστηριότητας, επιδιώκεται η σωστή διαχείριση των πόρων, η αποτελεσματική αντιμετώπιση προβλημάτων και τέλος, η αναθεώρηση του συνολικού προγράμματος του έργου σε κάθε απόκλιση από τα προσχεδιασμένα. Με το πέρας της φάσης, γίνεται αξιολόγηση των παραδοτέων ως προς τις συμφωνηθείσες ιδιότητες.
  4. *Κλεισίματος:*
     Το έργο αξιολογείται ως προς τα παραδοτέα (ως κατασκευάστηκε), γίνεται η εκκαθάριση οικονομικών και άλλων εκκρεμοτήτων, με απώτερο στόχο το κλείσιμο του έργου μέσα από τη σύναψη των αρχείων του έργου.

- Ρεαλισμός:

  Κρίσιμος παράγοντας για κάθε έργο είναι το στοιχείο του ρεαλισμού, το οποίο προϋποθέτει την αντικειμενικά επιτεύξιμη υλοποίηση των στόχων του κατά τη θέσπισή τους. Επιπέον, απαιτεί τον εκ των προτέρων συνυπολογισμό των απαιτήσεων σε οικονομικούς και ανθρώπινους πόρους, καθώς και της υφιστάμενης διαθεσιμότητας των τελευταίων.

- Δυνατότητα εκτίμησης:

  Τα έργα προγραμματίζονται και εν συνεχεία αναλύονται σε μετρήσιμους στόχους, οι οποίοι υπόκεινται σε επιμέρους και συνολική αξιολόγηση.

Τέλος, το πιο σημαντικό κοινό χαρακτηριστικό όλων των έργων είναι ο χαρακτηρισμός τους ως «επιτυχημένο», ο οποίος αποδίδεται σε ένα έργο που ολοκληρώθηκε εντός των προκαθορισμένων χρονοδιαγραμμάτων και του προϋπολογισμού, που ικανοποίησε τις προσδοκίες και τις ανάγκες των ενδιαφερομένων, ενώ παράλληλα τα χαρακτηριστικά του

παραδοτέου του είναι σε συμφωνία με τις προκαθορισμένες και συμφωνηθείσες προδιαγραφές και απαιτήσεις (Cooke-Davies, 2002, Atkinson, 1999).

Χαρακτηριστικά της Διαχείρισης Έργων

Ένα έργο, αποτελεί ένα ζωντανό οργανισμό, εντός του οργανισμού που υλοποιείται, με συνεχή εξέλιξη και δεν λειτουργεί σαν ένα κλειστό σύστημα. Αντίθετα, απαιτεί στοιχεία εισόδων προερχόμενα από τα έγκατα του οργανισμού και μεταφέρει προς αυτόν τα παραγόμενα αποτελέσματά του (PMI, 2013). Οι διαδικασίες που ακολουθούνται με απώτερο στόχο την απόδοση του άνωθι χαρακτηρισμού του επιτυχημένου έργου, αφορούν τη διαχείριση και διοίκηση έργων. Ως Διαχείριση Έργων ορίζεται η εφαρμογή γνώσεων, μεθόδων, εργαλείων, τεχνικών και ικανοτήτων στις δραστηριότητες ενός έργου, με απώτερο στόχο την ικανοποίηση των απαιτήσεων και την υλοποίηση των στόχων του (PMI, 2013, ISO, 2012). Η αναφερόμενη εφαρμογή των γνώσεων απαιτεί έναν αποτελεσματικό χειρισμό των διαδικασιών που αφορούν τη Διαχείριση Έργων.

Στη Διαχείριση Έργων, μια διαδικασία (*process*) αποτελεί ένα σύνολο αλληλοσυσχετιζόμενων ενεργειών και δραστηριοτήτων, που δημιουργεί ένα προκαθορισμένο προϊόν, υπηρεσία ή αποτέλεσμα . Κάθε διαδικασία χαρακτηρίζεται από τις εισόδους, τα εργαλεία και τις τεχνικές που δύνανται να εφαρμοστούν, και τα παραχθέντα αποτελέσματα ως εκροές. Οι διακασίες της Διαχείρισης Έργων κατηγοριοποιούνται σε πέντε βασικές ομάδες διαδικασιών (*Project Management Process Groups*), αναφορικά με τη φύση της κάθε μίας, τις μεταξύ τους εξαρτήσεις και τους σκοπούς που επιτελούν, ως εξής (PMI, 2013):

1. Ομάδα Διαδικασιών Έναρξης:

   προσδιορίζουν ένα νέο έργο ή μια νέα φάση/στάδιο ενός υφιστάμενου έργου, μέσα από διεργασίες.

2. Ομάδα Διαδικασιών Προγραμματισμού:

   καθορίζουν το εύρος-πεδίο εφαρμογής του έργου, βελτιώνουν τους επικείμενους στόχους και προσδιορίζουν την πορεία δράσης για την επιτυχημένη επίτευξη των στόχων του έργου.

3. Ομάδα Διαδικασιών Εκτέλεσης:

   υλοποιούν την ολοκλήρωση των προκαθορισμένων και απαιτούμενων δραστηριοτήτων μέσα από την εκτέλεση των απαιτούμενων εργασιών για την ικανοποίηση των απαιτήσεων του έργου.

4. Ομάδα Διαδικασιών Παρακολούθησης και Ελέγχου:

   παρακολουθούν, αναθεωρούν και εξασφαλίζουν την ομαλή εξέλιξη και πρόοδο εκτέλεσης του έργου. Μέσα από τον εντοπισμό τυχόν προβλημάτων, επιλέγονται οι κατάλληλες διορθωτικές ενέργειες, μέσα από την έναρξη εφαρμογής των οποίων, θα συνεχιστεί η εκτέλεση του έργου συμμορφώνοντας όποιες αποκλίσεις από τις προκαθορισμένες απαιτήσεις.

5. Ομάδα Διαδικασιών Κλεισίματος:

   πραγματοποιούνται με στόχο την ολοκλήρωση όλων των δραστηριοτήτων σε όλες τις ομάδες διαδικασιών, για την τελική ολοκλήρωση και το καθολικό κλείσιμο του έργου ή της φάσης του έργου.

Οι άνωθι διαδικασίες κάθε ομάδας και μεταξύ διαφορετικών ομάδων, βρίσκονται σε συνεχή αλληλεπίδραση και αλληλεξάρτηση. Οι ενέργειες που γίνονται κατά την εκτέλεση

μιας διαδικασίας επηρεάζουν όχι μόνο αυτή καθ' αυτή τη διαδικασία που αφορούν, αλλά και τις υπόλοιπες σχετικές διαδικασίες. Η άμεση επικοινωνία μεταξύ των διαδικασιών της Διαχείρισης Έργων, χρήζει σωστής διαχείρισης των αλληλοσυσχετίζόμενων διαδικασιών και ενεργειών, μέσω ενδεχόμενων αναθεωρήσεων των απαιτήσεων και στόχων του έργου, ώστε να αποφευχθούν μελλοντικά προβλήματα και να εκπληρωθούν οι απαιτήσεις των χορηγών, των πελατών και των υπόλοιπων εμπλεκομένων του έργου.

Μια διαφορετική κατηγοριοποίηση των διαδικασιών της Διαχείρισης Έργων αφορά της περιοχές γνώσης (*Knowledge Areas*). Μια περιοχή γνώσης αναπαριστά *«ένα πλήρες σύνολο σεναρίων, όρων και δραστηριοτήτων που αφορούν έναν επαγγελματικό τομέα, τον τομέα της Διαχείρισης Έργων ή μια συγκεκριμένη περιοχή εξειδίκευσης».* Η κατηγοριοποίηση, λοιπόν, των διαδικασιών, έγκειται σε δέκα περιοχές γνώσης, η χρήση των οποίων πρέπει να γίνεται με κριτική σκοπιά από την ομάδα έργου για την εφαρμογή εκείνων που σχετίζονται και αφορούν άμεσα το έργο ανάλογα με τη φύση και τα ιδιαίτερα χαρακτηριστικά του τελευταίου. Οι δέκα περιοχές γνώσης είναι οι ακόλουθες (PMI, 2013):

1. Διαχείριση Εύρους *(Project Scope Management)*:

   αφορά διαδικασίες για την εγγύηση της σωστής και αποτελεσματικής ολοκλήρωσης του έργου, εξασφαλίζοντας ότι το έργο περιλαμβάνει όλες τις απαραίτητες εργασίες για την αποτελεσματική υλοποίησή του, χωρίς περιττές δραστηριότητες. Πρωταρχικός στόχος είναι η συγκέντρωση των απαιτήσεων, ο καθορισμός, των απαραίτητων για το έργο, στοιχείων ύπαρξης εντός του έργου, η τελική επικύρωση και έλεγχος του πεδίου δράσης.

2. Διαχείριση Χρόνου *(Project Time Management)*:

   αφορά τις διαδικασίες που απαιτούνται για τη διαχείριση της χρονικής ολοκλήρωσης του έργου, μέσω του καθορισμού των πολιτικών, των διαδικασιών και της τεκμηρίωσης για το σχεδιασμό, την ανάπτυξη, τη διαχείριση, την εκτέλεση και τον έλεγχο του χρονοπρογράμματος του έργου. Πιο συγκεκριμένα, περιλαμβάνει διαδικασίες καθορισμού των -προς εκτέλεση- δραστηριοτήτων και την αλληλουχία αυτών, εκτίμησης της χρονικής διάρκειας και της απαίτησης σε πόρους κάθε δραστηριότητας, ανάπτυξης προγράμματος, όπου μέσα από την ανάλυση των αλληλουχιών, διαρκειών, απαιτήσεων σε πόρους των δραστηριοτήτων και των υφιστάμενων περιορισμών δημιουργείται το χρονοπρόγραμμα του έργου, και τέλος ελέγχου του χρονοπρογράμματος για τη διαχείριση των αλλαγών και την αναθεώρηση αυτού.

3. Διαχείριση Κόστους *(Project Cost Management)*:

   αφορά τις διαδικασίες που σχετίζονται με το σχεδιασμό, την εκτίμηση, τον προϋπολογισμό, τη χρηματοδότηση, τη διαχείριση και τον έλεγχο όλων των υφιστάμενων κοστών, με απώτερο στόχο τη δυνατότητα ολοκλήρωσης του έργου εντός του εγκεκριμένου προϋπολογισμού. Περιλαμβάνει την προσεγγιστική εκτίμηση των απαιτούμενων οικονομικών πόρων για την υλοποίηση των δραστηριοτήτων του έργου, τη συνολική ανάπτυξη του προϋπολογισμού και τον έλεγχο, μέσα από τη συνεχή παρακολούθηση, της κατάστασης του έργου για την ανανέωση των κοστών και τη διαχείριση των αλλαγών ώστε να υπάρχει συμφωνία με τον αρχικό προϋπολογισμό.

4. Διαχείριση Ποιότητας *(Project Quality Management)*:

αφορά τις διαδικασίες και δραστηριότητες για τον καθορισμό των πολιτικών, των στόχων και των ευθυνών ποιότητας, ώστε το έργο να ικανοποιήσει τις ανάγκες για τις οποίες κλήθηκε. Περιλαμβάνει την καταγραφή των απαιτήσεων ή/και των προτύπων ποιότητας του έργου και των παραδοτέων, την τεκμηρίωση του τρόπου συμμόρφωσης του έργου για την αποτελεσματική ανταπόκριση στις απαιτήσεις ποιότητας, τη διασφάλιση υλοποίησης των προαναφερθέντων δραστηριοτήτων, και τον έλεγχο ποιότητας, όπου μέσω της αέναης παρακολούθησης και καταγραφής κάθε παραχθέντος αποτελέσματος γίνεται συνεχής αξιολόγηση και προτείνονται οι απαραίτητες αλλαγές για τη συμμόρφωση του έργου και τη διασφάλιση επικύρωσης των απαιτήσεων αυτού.

5. Διαχείριση Ανθρώπινων Πόρων *(Project Human Resource Management)*:

   αφορά τις διαδικασίες οργάνωσης, διαχείρισης και καθοδήγησης της ομάδας έργου. Περιλαμβάνει διαδικασίες αναγνώρισης και τεκμηρίωσης απαιτούμενων δεξιοτήτων, ανάληψης ευθυνών, διαμοιρασμού ρόλων και σχέσεων αναφοράς, απόκτησης του απαραίτητου ανθρώπινου δυναμικού σύμφωνα με τις ανάγκες για τη σύμπραξη της ομάδας του έργου, ενίσχυσης του συλλογικού πνεύματος μεταξύ των μελών της ομάδας και βελτίωσης των ατομικών δεξιοτήτων. Τέλος, αφορά διαδικασίες παρακολούθησης της απόδοσης των μελών της ομάδας, συνεχής ενημέρωσης, επίλυσης προβλημάτων και πραγματοποίησης απαραίτητων αλλαγών, με στόχο τη βελτιστοποίηση της συνολικής απόδοσης του έργου.

6. Διαχείριση Επικοινωνίας *(Project Communications Management)*:

   αφορά διαδικασίες που απαιτούνται για την εξασφάλιση της έγκαιρης και κατάλληλης σχεδίασης, συλλογής, δημιουργίας, διανομής, αποθήκευσης, διαχείρισης, παρακολούθησης, επίβλεψης και τελικής διάθεσης των πληροφοριών του έργου, μεταξύ της διοίκησης, της ομάδας έργου και των ενδιαφερομένων. Η αποτελεσματική επικοινωνία γεφυρώνει τις διαφορές μεταξύ των ενδιαφερομένων που ενδέχεται να έχουν διαφορετικές πολιτισμικές και οργανωτικές δομές, επίπεδα εμπειρίας, οπτικές προσέγγισης και συμφέροντα, που ενδεχομένως να επηρεάσουν την εκτέλεση και το τελικό αποτέλεσμα του έργου (Allen et al., 1980).

7. Διαχείριση Κινδύνου *(Project Risk Management)*:

   αφορά διαδικασίες διεξαγωγής του σχεδιασμού της διαχείρισης κινδύνων, αναγνώρισης και ανάλυσης -ποιοτικά και ποσοτικά- ενδεχόμενων κινδύνων που ελλοχεύουν, εντοπισμού και περιορισμού αυτών, παρακολούθησης των εναπομένοντων κινδύνων, ανάπτυξης εναλλακτικών προτάσεων και δράσεων για την ενίσχυση των ευκαιριών, σχεδιασμού αντίδρασης στις αρνητικές καταστάσεις, καθώς και αξιολόγησης της αποτελεσματικότητας των διαδικασιών αντιμετώπισης σε όλη τη διάρκεια ζωής του έργου. Οι στόχοι της Διαχείρισης Κινδύνων συνοψίζονται στην αύξηση των πιθανοτήτων και των επιπτώσεων όλων των θετικών ενδεχόμενων γεγονότων και στην ταυτόχρονη αντίστοιχη μείωση των αρνητικών, για την αξιοποίηση των ευκαιριών, τον περιορισμό των απειλών και την αποτελεσματική αντιμετώπιση των κινδύνων.

8. Διαχείριση Προμηθειών *(Project Procurement Management)*:

αφορά διαδικασίες που απαιτούνται για την αγορά ή την απόκτηση προϊόντων, υπηρεσιών ή αποτελεσμάτων από τρίτους εκτός της ομάδας έργου, με τον οργανισμό να βρίσκεται στη θέση είτε του αγοραστή είτε του πωλητή. Περιλαμβάνει διαδικασίες διαχείρισης των συμβάσεων, αλλαγής των διαδικασιών που απαιτούνται για την ανάπτυξη και σύμπραξη μιας σύμβασης ή ενός δελτίου παραγγελίας, ελέγχου κάθε εκδοθέντος συμβολαίου από εξωτερικό οργανισμό και χόρηγησης συμβατικών υποχρεώσεων στην ομάδα έργου.

9. Διαχείριση Ενδιαφερομένων *(Project Stakeholder Management)*:

   αφορά διαδικασίες για τον εντοπισμό ατόμων, ομάδων ή οργανισμών που δύναται να επηρεάσουν το έργο ή να επηρεαστούν από αυτό, την ανάλυση των προσδοκιών και των χαρακτηριστικών συμφερόντων των ενδιαφερομένων του έργου, την ανάπτυξη κατάλληλων στρατηγικών για την αποτελεσματική συμμετοχή των ενδιαφερομένων στις αποφάσεις και στην εκτέλεση του έργου, καθώς και την παρακολούθηση των σχέσεων των ενδιαφερομένων καθόλη τη διάρκεια του έργου. Επιπρόσθετα, επικεντρώνεται στη συνεχή επικοινωνία της διοίκησης με τους ενδιαφερόμενους, ώστε να κατανοηθούν πλήρως οι προσδοκίες και οι ανάγκες τους, να αντιμετωπιστούν επιτυχώς τυχόν ζητήματα και να διαχειριστούν αντικρουόμενα συμφέροντα. Η ικανοποίηση των ενδιαφερομένων αποτελεί ένα κρίσιμο στοιχείο επιτυχίας του έργου (Cooke-Davies, 2002).

10. Διαχείριση Ενοποίησης-Ολοκλήρωσης *(Project Integration Management)*:

    αφορά διαδικασίες και δραστηριότητες με στόχο τον εντοπισμό, τον καθορισμό, το συνδυασμό, την ενοποίηση και το συντονισμό των πολλαπλών και διαφορετικών διαδικασιών και δραστηριοτήτων της Διαχείρισης Έργων μεταξύ των διαφορετικών ομάδων, για την ελεγχόμενη εκτέλεση του έργου ως την καθολική του ολοκλήρωση. Επιπλέον, περιλαμβάνει τη λήψη αποφάσεων σχετικά με την κατανομή πόρων, την πραγματοποίηση συμβιβασμών μεταξύ αλληλοσυγκρουόμενων και ανταγωνιστικών στόχων και εναλλακτικών σχεδίων και λύσεων, και τέλος, τη διαχείριση των αλληλεξαρτήσεων μεταξύ των περιοχών γνώσης της Διαχείρισης Έργων.

Σαν υποστηρικτικά στοιχεία, οι περιοχές γνώσης παρέχουν μία αναλυτική περιγραφή των εισόδων και εξόδων κάθε διαδικασίας, σε συνδυασμό με μια περιγραφική επεξήγηση των εργαλείων και τεχνικών που συνήθως χρησιμοποιούνται σε κάθε διαδικασία την παραγωγή της εξόδου.

Εν κατακλείδι, σημειώνεται ότι η σωστή εφαρμογή των διαδικασιών της Διαχείρισης Έργων θα ενισχύσει την πιθανότητα επιτυχίας ενός έργου, όμως δεν έχει τη δυνατότητα να αποτρέψει ένα έργο που αποτυγχάνει, ώστε να επιτύχει. Ένα επιτυχημένο έργο μπορεί να λάβει χώρα χωρίς την απόλυτη επιτυχία της Διαχείρισης Έργων, όμως μια επιτυχημένη εφαρμογή των διαδικασιών της, αναντίρρητα, θα ενισχύσει την επιτυχία ενός έργου (Munns and Bjeirmi, 1996). Στη συνέχεια, η ανάλυση εστιάζεται στη Διαχείριση Χρόνου μέσω του χρονικού Προγραμματισμού Έργων (*Project Schedule/Time Management*), καθώς αποτελεί μια από τις πιο συνήθεις αιτίες αποτυχίας έργων (Goldratt, 1997, Lewis, 1995).

2.2 Προγραμματισμός Έργων

Ο χρονικός Προγραμματισμός Έργων (*Project Scheduling*), αποτελεί έναν από τους πιο κρίσιμους παράγοντες επιτυχίας ενός έργου, μιας και αποτελεί την πιο κοινή αιτία αποτυχίας του (Goldratt, 1997, Lewis, 1995). Ιδιαίτερα σε περιπτώσεις όπου αφορά την ανάπτυξη καινοτόμων και νέων προϊόντων, οι καθυστερήσεις ολοκλήρωσης των έργων υπόκεινται σε βαρύτατες ποινές, ακόμα μεγαλύτερες και από την υπέρβαση του ανώτατου ορίου του προϋπολογισμού (Goldratt, 1997). Χαρακτηριστικά αναφέρεται η μελέτη της Hewlett-Packard σε μια αναπτυγμένη αγορά (Demeulemeester and Herroelen, 2002), στην οποία διαπιστώθηκε ότι σε περίπτωση που η ολοκλήρωση ενός έργου, που αφορά ένα νέο καινοτόμο προϊόν, υποστεί μια καθυστέρηση έξι μηνών, μπορεί να προκαλέσει απώλειες κερδών 33%, ενώ αν το συνολικό κόστος του έργου υπερβεί κατά 50% τον προϋπολογισμό, η αντίστοιχη απώλεια έγκειται σε ένα 3.5% (Suri, 1998). Ως εκ τούτου, δεν αποτελεί έκπληξη το γεγονός ότι π.χ. η ελαχιστοποίηση της συνολικής διάρκειας ενός έργου (ή κάποιας άλλης αντικειμενικής συνάρτησης), συναρτήσει περιορισμών πόρων και σχέσεων προτεραιότητας μεταξύ των δραστηριοτήτων, αποτελεί έναν από τους πρωταρχικούς στόχους στη Διαχείριση Έργων (Demeulemeester and Herroelen, 2002).

Ο Προγραμματισμός Έργων στοχεύει στη δημιουργία ενός εφικτού -υπό την έννοια του σεβασμού στους περιορισμούς που αφορούν τη χρήση και διαθεσιμότητα των πόρων, καθώς και των σχέσεων προτεραιότητας μεταξύ των δραστηριοτήτων του έργου- χρονοπρογράμματος, που ελαχιστοποιεί το κριτήριο απόδοσης του προγραμματισμού. Το παραχθέν χρονοπρόγραμμα, αποτελεί το πρόγραμμα αναφοράς (*baseline schedule*), βάσει του οποίου θα εκτελεστεί το έργο, θα κατανεμηθούν οι πόροι, θα προγραμματιστούν εξωτερικές και εσωτερικές δραστηριότητες της εφοδιαστικής αλυσίδας του οργανισμού που αναλαμβάνει το έργο, όπως προμήθεια μηχανημάτων, προληπτική συντήρηση και παράδοση εσωτερικών και εξωτερικών παραγγελιών, θα γίνει η σύναψη των απαραίτητων συμβάσεων και θα καθοριστούν τα χρονικά όρια παράδοσης και προθεσμιών των αποτελεσμάτων του έργου (Herroelen and Leus, 2005).

Πρακτικές εφαρμογές του, λαμβάνουν χώρα σε διαφορετικών ειδών βιομηχανίες, όπως κατακευαστικών, ανάπτυξης λογισμικών, μεταποιητικών και παραγωγικών εταιριών κ.ό.κ. Παράλληλα, αποτελεί ένα συνεχώς αυξανόμενης σημαντικότητας πεδίο, για *make-to-order* εταιρείες, όπου ύστερα από την τοποθέτηση κάποιας παραγγελίας κατασκευάζεται το τελικό προϊόν, με κύριο χαρακτηριστικό την περιορισμένη διαθέσιμη ποσότητα σε πόρους λόγω της φύσης της διαχείρισης που εξυπηρετεί (Brucker et al., 1998). Εν αντιστοιχία με τον επιχειρηματικό κόσμο, ο Προγραμματισμός Έργων αποτελεί ένα ελκυστικό ερευνητικό πεδίο ενδιαφέροντος, καθώς η μεγάλη ποικιλία διαφορετικών εκφάνσεων του ίδου προβλήματος και διαφορετικών αντικειμένων, σε συνδυασμό με τη δυσκολία επίλυσης αυτόνομων και συνδυαστικών προβλημάτων, προσελκύει ολοένα και περισσότερους ερευνητές.

2.2.1 Συστατικά Στοιχεία Προβλημάτων Προγραμματισμού Έργων

Τα προβλήματα Προγραμματισμού Έργων (*Project Scheduling Problems- PSP*) ενδέχεται να αποτελούνται από γεγονότα (events/*milestones*), δραστηριότες (*activities*), πόρους (*resources*), σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων (*precedence relations*), και κριτήρια απόδοσης (*performance measures*) (Kolisch and Padman, 1997). Στόχος είναι η δημιουργία ενός εφικτού χρονοπρογράμματος (*feasible schedule*), με σεβασμό

στους περιορισμούς διαθεσιμότητας των πόρων και στις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, μέσα από τη βελτιστοποίηση του επιλεγμένου κριτηρίου απόδοσης υπό τη μορφή αντικειμενικής συνάρτησης. Ακολούθως, παρουσιάζονται κάποια συνήθη συστατικά στοιχεία ενός έργου, που επρόκειτω να προγραμματιστεί, με την προϋπόθεση ότι όλα τα απαραίτητα δεδομένα είναι διαθέσιμα, ντετερμινιστικά και ακέραιης τιμής.

Γεγονότα

Ένα γεγονός (event/*milestone*) αποτελεί ένα συμβάν που χρήζει ιδιαίτερου ενδιαφέροντος. Τοποθετείται σε μια συγκεκριμένη χρονική στιγμή, όπως στην έναρξη ή στη λήξη κάποιας δραστηριότητας ή μιας φάσης εκτέλεσης του έργου και έχει μηδενική διάρκεια και απαιτήσεις σε πόρους. Επιπρόσθετα, αποτελεί μια κατευθυντήρια γραμμή όσον αφορά την πορεία εκτέλεσης του έργου για να γίνουν οι πιθανές απαραίτητες διορθωτικές ενέργειες, ενώ παράλληλα ενδέχεται να αποτελεί τη χρονική στιγμή λήψης μια κομβικής απόφασης ή μιας πληροφορίας που θα επηρεάσει σημαντικά την πορεία εξέλιξης και ολοκλήρωσης του έργου (Eisenhardt and Tabrizi, 1995).

Δραστηριότητες

Ένα έργο αποτελείται από ένα συγκεκριμένο αριθμό δραστηριοτήτων (*activities, jobs, tasks, operations*), κάθε μια εκ των οποίων, πρέπει να εκτελεστεί με έναν συγκεκριμένο τρόπο (*mode*) από ενδεχόμενους πολλούς, για την επιτυχή ολοκλήρωση του έργου. Κάθε τρόπος εκτέλεσης αντιπροσωπεύει ένα σαφή και ξεχωριστό τρόπο και μέθοδο υλοποίησης της δραστηριότητας, καθορίζοντας τη διάρκειά της, την απαιτούμενη ποσότητα για κάθε τύπο, απαραίτητων για την υλοποίησή της, πόρων, καθώς και πιθανές ταμειακές εισροές και εκροές κατά την έναρξη, τη διάρκεια εκτέλεσης, ή/και τη λήξη της (Kolisch and Padman, 1997).

Κάθε δραστηριότητα, χαρακτηρίζεται από:

- τη διάρκειά της (*duration*), μετρούμενη σε ακέραιο αριθμό χρονικών περιόδων υποδεικνύοντας τον απαιτούμενο χρόνο ως την ολοκλήρωσή της. Η διάρκεια μιας δραστηριότητας μπορεί να έχει σταθερή και συγκεκριμένη τιμή ή να είναι στοχαστική (*fuzzy numbers*) όταν υπάρχει το αίσθημα της ανακρίβειας στην πρόβλεψη της τιμής της,.
- την απαιτούμενη ποσότητα και το απαραίτητο είδος από κάθε διαθέσιμο πόρο (*resource requirement*) για την εκτέλεσή της κάθε χρονική στιγμή.
- τις ταμειακές εισροές ή εκροές που επιφέρει σε κάποιο στάδιό της (*cash flows*)
- τον τρόπο εκτέλεσης. Σε περίπτωση που υπάρχουν πολλοί διαφορετικοί τρόποι εκτέλεσης σε μια δραστηριότητα, η τελευταία χαρακτηρίζεται ως *πολλαπλών τρόπων εκτέλεσης* (*multi-mode*), ενώ σε αντίθετη περίπτωση, όπου υπάρχει ένας διακριτός τρόπος εκτέλεσης, *απλού τρόπου εκτέλεσης* (*single-mode*).
- τις σχέσεις προτεραιότητας, ως το σύνολο των δραστηριοτήτων των προαπαιτούμενων (*predecessors*), όπου για να λάβει χώρα η έναρξη μιας δραστηριότητας πρέπει να έχουν υλοποιηθεί όλες οι προαπαιτούμενες δραστηριότητές της, και των διαδόχων (*successors*), όπου η ολοκλήρωση της συγκεκριμένης δραστηριότητας δίνει τη δυνατότητα στους διαδόχους να ξεκινήσουν την εκτέλεσή τους.

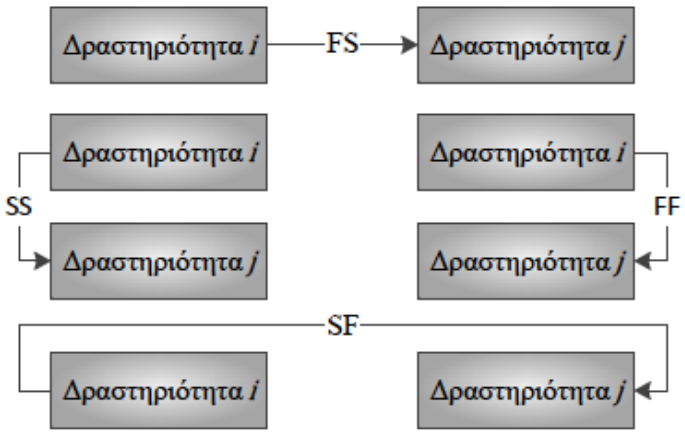
- τη δυνατότητα προεκχώρησης (preemption), όπου επιτρέπεται η διακοπή της εκτέλεσης της δραστηριότητας και η συνέχισή της είτε από το σημείο όπου διεκόπη (*preempt-resume*) είτε από την αρχή υλοποιώντας ξανά και το στάδιο πριν την εκχώρηση (*preempt-repeat*), ή τη μη δυνατότητα προεκχώρησης (no-preemption), όπου κάθε δραστηριότητα πρέπει να ολοκληρωθεί από τη στιγμή της έναρξής της, χωρίς καμία διακοπή.

Τέλος, σε κάθε έργο υπάρχουν δύο *βοηθητικές δραστηριότητες* (*dummy activities*), οι απαιτήσεις των οποίων σε πόρους, καθώς και οι διάρκειές τους λαμβάνουν μηδενική τιμή, ενώ αναπαριστούν την έναρξη και τη λήξη του έργου. Ορίζεται λοιπόν, η βοηθητική δραστηριότητα αρχής (*dummy source activity*) και η βοηθητική δραστηριότητα τέλους (*dummy sink activity*).

Σχέσεις Προτεραιότητας

Τεχνολογικοί περιορισμοί καθιστούν κάποιες σχέσεις προτεραιότητας (*precedence relations*) μεταξύ των δραστηριοτήτων ενός έργου, οι οποίες καθορίζουν την έναρξη ή την ολοκλήρωση μιας δραστηριότητας ανάλογα με την έναρξη ή την ολοκλήρωση των συνδεόμενων με αυτή, δραστηριοτήτων. Οι σχέσεις προτεραιότητας που δύνανται να αναπτυχθούν μεταξύ των δραστηριοτήτων είναι:

- *Σχέση τέλους-αρχής* (*Finish-Start, FS*), όπου η δραστηριότητα *j* δεν μπορεί να ξεκινήσει, αν δεν έχει προηγηθεί η ολοκλήρωση της δραστηριότητας *i*. Η δεδομένη σχέση τέλους-αρχής αποτελεί την παραδοσιακή σχέση προτεραιότητας των τεχνικών CPM/PERT, με μηδενική χρονική καθυστέρηση.
- *Σχέση τέλους-τέλους* (*Finish-Finish, FF*), όπου η δραστηριότητα *j* δεν μπορεί να ολοκληρωθεί αν δεν έχει προηγηθεί η ολοκλήρωση της δραστηριότητας *i*.
- *Σχέση αρχής-αρχής* (*Start-Start, SS*), όπου η δραστηριότητα *j* δεν μπορεί να ξεκινήσει αν δεν έχει προηγηθεί η έναρξη της δραστηριότητας *i*.
- *Σχέση αρχής-τέλους* (*Start-Start, SS*), όπου η δραστηριότητα *j* δεν μπορεί να ολοκληρωθεί αν δεν έχει προηγηθεί η έναρξη της δραστηριότητας *i*.



Σχήμα 2. Διαγραμματική απεικόνιση των σχέσεων προτεραιότητας (PMI, 2013)

Παράλληλα με τις σχέσεις προτεραιότητας, ενδέχεται να υπάρχουν και ελάχιστα/μέγιστα χρονικά διαστήματα καθυστέρησης (*minimum/maximum time lags*) μεταξύ των γεγονότων έναρξης ή λήξης που ορίζουν οι σχέσεις προτεραιότητας, ορίζοντας

γενικευμένες σχέσεις προτεραιότητας (*Generalized Precedence Relations, GPR*), οι οποίες θα παρουσιαστούν στη συνέχεια στο αντίστοιχο πρόβλημα προγραμματισμού που χαρακτηρίζουν (Κεφάλαιο 2.3.2.1).

Πόροι

Οι πόροι που χρησιμοποιούνται από τις δραστηριότητες του έργου, ταξινομούνται αναφορικά με τον τύπο που αντιπροσωπεύουν στις εξής κατηγορίες (Demeulemeester and Herroelen, 2002, Kolisch and Padman, 1997) :

Ανανεώσιμοι πόροι (*renewable resources*), οι οποίοι είναι διαθέσιμοι για κάθε χρονική περίοδο, με τον περιορισμό να αφορά τη συνολική διαθεσιμότητα κάθε πόρου ανά χρονική περίοδο και αντίστοιχα τη συνολική χρησιμοποίηση αυτού. Οι ανανεώσιμοι πόροι δεν καταναλώνονται, απλά δεσμεύονται για συγκεκριμένες χρονικές περιόδους και στη συνέχεια είναι ξανά διαθέσιμοι προς χρήση. Χαρακτηριστικά παραδείγματα ανανεώσιμων πόρων αποτελούν το ανθρώπινο δυναμικό, τα μηχανήματα, ο τεχνολογικός εξοπλισμός, ο διαθέσιμος χώρος αποθήκευσης κ.ό.κ. Το σύνολο των ανανεώσιμων πόρων ορίζεται ως R^{ρ}, η διαθεσιμότητα του ανανεώσιμου πόρου k ανά χρονική περίοδο t είναι R_{kt}^{ρ}, ενώ σε περίπτωση που η διαθεσιμότητα του πόρου k είναι σταθερή καθόλη τη διάρκεια του έργου, τότε ορίζεται ως R_{k}^{ρ}. Τέλος, η ανά περίοδο απαιτούμενη ποσότητα σε μονάδες πόρου k από την δραστηριότητα i, ενδέχεται να είναι σταθερή r_{ik}^{ρ} (π.χ. πέντε εργάτες για κάθε μία από τις τρεις μέρες εκτέλεσης της δραστηριότητας), ή μεταβλητή ανά χρονική περίοδο r_{ikt}^{ρ} (π.χ. πέντε κατά την πρώτη, έξι κατά τη δεύτερη και τρεις κατά την τρίτη μέρα εκτέλεσης της δραστηριότητας).

Μη Ανανεώσιμοι πόροι (*non renewable resources*), οι οποίοι είναι διαθέσιμοι όχι για κάθε χρονική περίοδο με ένα περιορισμένο συνολικό μέγεθος, αλλά για όλη τη διάρκεια του έργου με ένα περιορισμένο διαθέσιμο συνολικό μέγεθος κατανάλωσης. Οι πόροι αυτής της κατηγορίας πρακτικά καταναλώνονται, καθώς η χρησιμοποιούμενη ποσότητα αφαιρείται μόνιμα από το διαθέσιμο συνολικό μέγεθος κατανάλωσης. Χαρακτηριστικά παραδείγματα μη ανανεώσιμων πόρων αποτελούν τα χρήματα, οι πρώτες ύλες, η ενέργεια κ.ά. Το σύνολο των μη ανανεώσιμων πόρων ορίζεται ως R^{ν}, η διαθεσιμότητα του μη ανανεώσιμου πόρου k είναι R_{k}^{ν}, και είναι σταθερή καθόλη τη διάρκεια του έργου. Τέλος, η ανά περίοδο απαιτούμενη ποσότητα σε μονάδες πόρου k από την δραστηριότητα i, ενδέχεται να είναι σταθερή r_{ik}^{ν}, ή μεταβλητή ανά χρονική περίοδο r_{ikt}^{ν}.

Διπλά περιορισμένοι πόροι (*doubly-constrained resources*), οι οποίοι είναι περιορισμένοι τόσο για κάθε χρονική περίοδο, όσο και για τη συνολική διάρκεια του έργου. Τυπικά παραδείγματα διπλά περιορισμένων πόρων αποτελούν το κεφάλαιο με περιορισμένες ταμειακές ροές ανά χρονική περίοδο και περιορισμένο συνολικό διαθέσιμο χρηματικό ποσό, καθώς και οι ανθρωποώρες κάθε ημέρας σε συνδυασμό με ένα περιορισμένο αριθμό διαθέσιμων ανθρωποωρών για το σύνολο του έργου. Οι διπλά περιορισμένοι πόροι μπορούν να διαχειριστούν σαν έναν ανανεώσιμο και ένα μη ανανεώσιμο πόρο, $k \in R^{\nu} \cap R^{\rho}$, με κάθε έναν να λαμβάνει τις αντίστοιχες τιμές ανάλογα με την απαίτηση που εξυπηρετεί, (Talbot, 1982).

Μερικώς Ανανεώσιμοι Πόροι (*partially renewable resources*), όπως ορίζονται από τους (Böttcher et al., 1999), οι οποίοι έχουν περιορισμένη διαθεσιμότητα μέσα σε συγκεκριμένα χρονικά διαστήματα εντός του χρονικού ορίζοντα ολοκλήρωσης του έργου. Κάθε τέτοιου είδους πόρος χαρακτηρίζεται από έναν αριθμό υποσυνόλων-διαστημάτων χρονικών περιόδων, σε κάθε ένα από τα οποία ο πόρος λαμβάνει μια συγκεκριμένη τιμή

διαθεσιμότητας. Οι μερικώς ανανεώσιμοι πόροι αποτελούν μια γενική κατηγορία που εμπεριέχει ανανεώσιμους και μη ανανεώσιμους πόρους, και κατά συνέπεια και διπλά περιορισμένους (ένας μερικώς ανανεώσιμος πόρος με συγκεκριμένη διαθεσιμότητα σε χρονικό διάστημα ίσο με μία χρονική περίοδο αποτελεί έναν ανανεώσιμο πόρο, και αντίστοιχα με συγκεκριμένη διαθεσιμότητα σε χρονικό διάστημα ίσο με το χρονικό ορίζοντα του έργου αποτελεί έναν μη ανανεώσιμο πόρο). Το σύνολο των μερικώς ανανεώσιμων πόρων ορίζεται ως ως R^{π}, ενώ για κάθε πόρο $k \in R^{\pi}$, ορίζεται ένα σύνολο Π_k, που ορίζει τον αριθμό των διαστημάτων $P_{kj} \in \Pi_k$, κάθε ένα από τα οποία αποτελεί ένα υποσύνολο χρονικών περιόδων. Για κάθε υποσύνολο-διάστημα P_{kj}, ορίζεται η αντίστοιχη διαθέσιμη ποσότητα του μερικώς ανανεώσιμου πόρου k ως R_{kj}^{π}. Τέλος, η ανά περίοδο απαιτούμενη ποσότητα σε μονάδες πόρου k από την δραστηριότητα i, είναι r_{ik}^{π} (Böttcher et al., 1999, Demeulemeester and Herroelen, 2002).

Κριτήρια Απόδοσης

Στόχος του Προγραμματισμού Έργων είναι η δημιουργία ενός εφικτού χρονοπρογράμματος (*feasible schedule*), με σεβασμό στους περιορισμούς διαθεσιμότητας των πόρων και στις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, παραχθέν από τη βελτιστοποίηση του επιλεγμένου κριτηρίου απόδοσης υπό τη μορφή αντικειμενικής συνάρτησης. Κάποια τυπικά κριτήρια απόδοσης είναι τα εξής (Kolisch and Padman, 1997):

1. *Ελαχιστοποίηση Διάρκειας Έργου (Makespan Minimization).* Αποτελεί το πιο σύνηθες και εφαρμόσιμο κριτήριο απόδοσης στον τομέα του προγραμματισμού έργων. Ως διάρκεια έργου ορίζεται το χρονικό διάστημα μεταξύ της έναρξης και της λήξης του έργου. Δεδομένου ότι η αρχή του έργου συνήθως θεωρείται τη χρονική στιγμή $t = 0$, η ελαχιστοποίηση της διάρκειας του έργου, έγκειται στην ελαχιστοποίηση της μέγιστης χρονικής στιγμής ολοκλήρωσης κάθε δραστηριότητας του έργου, ώστε να ελαχιστοποιηθεί και η συνολική διάρκεια του έργου.
2. *Μεγιστοποίηση Καθαρής Παρούσας Αξίας (Maximize Net Present Value).* Όταν στο υπό προγραμματισμό έργο, υπάρχουν καθορισμένα επίπεδα ταμειακών ροών, υπό τη μορφή δαπανών για την έναρξη των δραστηριοτήτων και την εξέλιξη των πληρωμών σε διάφορες φάσεις του έργου, το κριτήριο της Καθαρής Παρούσας Αξίας αποτελεί το πιο κατάλληλο μέτρο μέτρησης της απόδοσής του (Bey et al., 1981). Το κριτήριο αυτό δημιουργεί ένα κρίσιμο δρόμο κόστους μαζί με το χρονοδιάγραμμα των δραστηριοτήτων, εν αντιθέσει με το κριτήριο ελαχιστοποίησης της διάρκειας που δημιουργεί ένα κρίσιμο δρόμο χρόνου και το χρονοπρόγραμμα(Kolisch and Padman, 1997).
3. *Ελαχιστοποίηση των κοστών (Minimize Costs).* Λόγω της πρακτικής εφαρμογής του, το συγκεκριμένο κριτήριο απόδοσης αποσπά ιδιαίτερο ενδιαφέρον, και παρουσιάζει δύο προσεγγίσεις: των κοστών των δραστηριοτήτων (*activity cost objectives*) και των αντίστοιχων των πόρων (*resource cost objectives*). Όσον αφορά τα κόστη των δραστηριοτήτων, ο τρόπος που οι δραστηριότητες εκτελούνται (π.χ. οι χρόνοι έναρξης ή ο επιλεγμένος τρόπος εκτέλεσης σε περιπτώσεις πολλαπλών τρόπων εκτέλεσης δραστηριοτήτων), οδηγεί σε άμεσα κόστη, τα οποία και ελαχιστοποιούνται. Αντίστοιχα, στην προσέγγιση των κοστών των πόρων, ο χρονοπρογραμματισμός των δραστηριοτήτων επηρεάζει έμμεσα τα κόστη μέσω των χρησιμοποιούμενων πόρων για την υλοποίηση των δραστηριοτήτων.
4. *Μεγιστοποίηση της ποιότητας (Maximize Quality).* Το συγκεκριμένο κριτήριο, στη βάση του οποίου έγκειται η ικανοποίηση του πιο κρίσιμου στόχου για τους διευθυντές

των έργων, αφορά την ποσοτικοποίηση και μέτρηση της ποιότητας του έργου, μέσω είτε της συμφωνίας του έργου με τις αρχικές απαιτήσεις, τις προθεσμίες και τον προϋπολογισμό, είτε του χρόνου που έχει αφιερωθεί για την επανεκτέλεση και διόρθωση λαθών σε δραστηριότητες, κ.ά (Icmeli-Tukel and Rom, 1997).

Τρόποι Αναπαράστασης Έργων

Ένα έργο έχει τη δυνατότητα αναπαράστασης ως ένα δίκτυο έργου (*project network*), ως ένα διάγραμμα Gannt (Clark et al., 1922), ως μια γραμμή προγραμματισμού (*track planning*) (Herroelen et al., 1998), ή ως μια γραμμή ευστάθειας (*line of balance*) (Lumsden, 1968). Ένα δίκτυο έργου απεικονίζεται ως μια γραφική αναπαράσταση γεγονότων, δραστηριοτήτων και σχέσεων προτεραιότητας, σε συνδυασμό με όλες τις παραμετρικές πληροφορίες που αφορούν τις δραστηριότητες του έργου, δημιουργώντας ένα γράφο $G = (N, A)$ συνιστώμενος από N κόμβους και A τόξα (Demeulemeester and Herroelen, 2002).

Γενικά, δύο τρόποι αναπαράστασης αποτελούν τις πιο διαδεδομένες τεχνικές απεικόνισης των δικτύων έργων: η δραστηριότητα-στο-τόξο (*Activity-on-Arc -AoA*) και η δραστηριότητα-στον-κόμβο (*Activity-on-Node -AoN*). Στην AoA αναπαράσταση, οι κόμβοι αποτελούν γεγονότα και τα τόξα αποτελούν δραστηριότητες. Βοηθητικές δραστηριότητες χρησιμοποιούνται για την εξασφάλιση των σχέσεων προτεραιότητας και βοηθητικοί κόμβοι για την απεικόνιση της αρχής και λήξης του έργου. Στην AoN αναπαράσταση, οι δραστηριότητες και οι συσχετιζόμενες με αυτές παραμετρικές πληροφορίες, αναπαρίστανται μέσα στους κόμβους και οι σχέσεις προτεραιότητας δηλώνονται από προσανατολισμένα τόξα. Η AoN αναπαράσταση δίδει καλύτερη εποπτεία των σχέσεων μεταξύ των δραστηριοτήτων του έργου, εν αντιθέσει με την AoA που δίδει καλύτερη εποπτεία των χρόνων. Χαρακτηριστικά αναφέρεται ότι προβλήματα προγραμματισμού έργων με κριτήριο απόδοσης την ελαχιστοποίηση της συνολικής διάρκειας ή έργα με πολύπλοκες σχέσεις προτεραιότητας, συνηθίζεται να αναπαρίστανται από AoN δίκτυα, ενώ προβλήματα με κριτήριο τη μεγιστοποίηση της καθαρής παρούσας αξίας από AoA δίκτυα καθώς παρέχουν μια διαισθητική απλότητα και ευκολία στη σύλληψη των λογικών ροών (Kolisch and Padman, 2001).

2.2.2 Κατάταξη Προβλημάτων Προγραμματισμού Έργων

Ένα κοινό σύστημα ταξινόμησης (*classification*) των προβλημάτων ενός συγκεκριμένου γνωστικού πεδίου, συντελεί στην οργανωμένη κατηγοριοποίηση μιας αχανούς ποικιλίας προβλημάτων, στην άμεση αναγνώριση των ιδιαίτερων χαρακτηριστικών και των υποθέσεων του κάθε προβλήματος, στη λακωνική παρουσίασή του μέσα από τη συμβολοσειρά που το εκπροσωπεί, στη σιωπηλή υπόδειξη της κατηγορίας προβλημάτων που ανήκει, της πολυπλοκότητάς του και των προτεινόμενων μεθόδων επίλυσής του, καθώς και στη σκιαγράφηση προβλημάτων που έχουν μείνει ανεξερεύνητα και χρήζουν έρευνας για την επίλυσή τους.

Η κατάταξη των προβλημάτων του Προγραμματισμού Έργων, στηρίζεται στο σύστημα ταξινόμησης των προβλημάτων προγραμματισμού μηχανών (*machine scheduling*), όπου γίνεται χρήση της σημειογραφίας τριών, πεδίων $\alpha|\beta|\gamma$, που εισήχθη από τους (Graham et al., 1979, Blazewicz et al., 1983). Στα προβλήματα μηχανών, το πεδίο α προσδιορίζει το περιβάλλον των μηχανών (προβλήματα μονών ή παράλληλων μηχανών, συστήματα συνεχούς ροής, συστήματα παραγωγής κατά παραγγελία, γενικά εργοστάσια ή συνδυασμός αυτών), το

πεδίο β προσδιορίζει τα χαρακτηριστικά των δραστηριοτήτων και των πόρων (περιλαμβάνει παραμέτρους χαρακτηρισμού για δυνατότητα προεκχώρησης, σχέσεις προτεραιότητας, χρονικές προθεσμίες, διάρκειες δραστηριοτήτων, παρτίδες και επιπλέον πόρους), ενώ το πεδίο γ σκιαγραφεί το κριτήριο βελτιστοποίησης (κριτήριο απόδοσης) (Herroelen et al., 1999).

Το σύστημα κατάταξης των προβλημάτων του Προγραμματισμού Έργων, ακολουθεί τη σημειογραφία του προαναφερθέντος συστήματος, με τα πεδία α, β, γ να προσδιορίζονται ως εξής (Herroelen et al., 1999):

Το <u>*πεδίο α*</u> αφορά τα χαρακτηριστικά των πόρων, τα οποία προσδιορίζονται από τρία, κατά μέγιστο, στοιχεία $\alpha_1, \alpha_2, \alpha_3$. Η παράμετρος $\alpha_1 \in \{°, 1, m\}$ υποδεικνύει τον αριθμό των διαφορετικών τύπων πόρων, με το ° να δηλώνει το κενό, η $\alpha_2 \in \{°, 1, T, 1T, v\} = \{$*χωρίς συγκεκριμένο τύπο, ανανεώσιμος, μη ανανεώσιμος, διπλά περιορισμένος, μερικώς ανανεώσιμος*$\}$ υποδεικνύει τους συγκεκριμένους τύπους χρησιμοποιούμενων πόρων, η $\alpha_3 \in \{°, va\} = \{$*η διαθεσιμότητα των ανανεώσιμων πόρων σταθερή σε κάθε χρονική στιγμή, η διαθεσιμότητα των ανανεώσιμων πόρων διαφέρει ανά χρονική στιγμή*$\}$ υποδεικνύει τα χαρακτηριστικά διαθεσιμότητας των πόρων.

Το <u>*πεδίο β*</u> αφορά τα χαρακτηριστικά των δραστηριοτήτων, τα οποία προσδιορίζονται από οχτώ, κατά μέγιστο, στοιχεία $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8$. Η παράμετρος $\beta_1 \in \{°, pmtn, pmtn - rep\} = \{$*δεν επιτρέπεται προεκχώρηση, προεκχώρηση και συνέχεια από το σημείο σταματήματος, προεκχώρηση και εκτέλεση από την αρχή* $\}$ υποδεικνύει την δυνατότητα προεκχώρησης (*preemption*), η $\beta_2 \in \{°, cpm, min, gpr, prob\} = \{$*χωρίς σχέσεις προτεραιότητας, σχέσεις τέλους-αρχής με μηδενική καθυστέρηση-όπως στο pert/cpm μοντέλο-, όλοι οι τύποι συσχετίσεων με ελάχιστους χρόνους καθυστέρησης, γενικευμένες σχέσεις προτεραιότητας με ελάχιστους και μέγιστους χρόνους καθυστέρησης, όπου το διάγραμμα δρατηριοτήτων βασίζεται σε πιθανότητες και δεν είναι σαφώς καθορισμένη η εξέλιξή του*$\}$ υποδεικνύει τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, η $\beta_3 \in \{°, \rho_j\} = \{$*όλοι οι χρόνοι προετοιμασίας μηδενικοί, οι χρόνοι διαφέρουν για κάθε δραστηριότητα j*$\}$ υποδεικνύει τα χαρακτηριστικά των χρόνων προετοιμασίας (*ready times*) έναρξης των δραστηριοτήτων, η $\beta_4 \in \{°, cont, d_j = d\} = \{$*αυθαίρετες διάρκειες ακέραιων τιμών, αυθαίρετες συνεχείς διάρκειες, όλες οι διάρκειες ίσες με d*$\}$ υποδεικνύει τα χαρακτηριστικά των διαρκειών των δραστηριοτήτων του έργου, η $\beta_5 \in \{°, \delta_j, \delta_n\} = \{$*χωρίς χρονικές διορίες, με χρονικές διορίες σε δραστηριότητες-γεγονότα, με χρονική διορία του έργου*$\}$ υποδεικνύει τις χρονικές διορίες (*deadlines*) που έχουν καθοριστεί, η $\beta_6 \in \{°, vr, disc, cont, \} = \{$*απαίτηση σταθερής ποσότητας πόρων σε κάθε χρονική περίοδο, απαίτηση μεταβλητής ποσότητας πόρων σε κάθε χρονική περίοδο, η απαίτηση πόρων αποτελεί διακριτή συνάρτηση της διάρκειας της δραστηριότητας, η απαίτηση πόρων αποτελεί συνεχή συνάρτηση της διάρκειας της δραστηριότητας*$\}$ υποδεικνύει τα χαρακτηριστικά των απαιτήσεων σε πόρους από τις δραστηριότητες του έργου, η $\beta_7 \in \{°, mu, id\} = \{$*οι δραστηριότητες είναι μονού τρόπου εκτέλεσης, οι δραστηριότητες είναι πολλαπλών προκαθορισμένων τρόπων εκτέλεσης, οι δραστηριότητες υπόκεινται σε περιορισμούς τρόπων εκτέλεσης ανάλογα με την ταυτότητά τους (id)*$\}$ υποδεικνύει τον αριθμό και τους τύπους των πιθανών τρόπων εκτέλεσης των δραστηριοτήτων του έργου, η $\beta_8 \in \{°, c_j, per, sched\} = \{$*χωρίς καθορισμένες ταμειακές ροές, με καθορισμένες ταμειακές ροές, με περιοδικές ταμειακές ροές, με ταμειακές ροές που τόσο το μέγεθος όσο και η χρονική περίοδος εφαρμογής τους πρέπει να καθοριστούν* $\}$ υποδεικνύει τις οικονομικές επιπτώσεις των δραστηριοτήτων του έργου.

Το <u>*πεδίο γ*</u> αφορά τα χαρακτηριστικά των κριτηρίων βελτιστοποίησης (κριτήρια απόδοσης), τα οποία μπορεί να είναι είτε σύντομης ολοκλήρωσης (κανονικά) $\gamma = reg$, είτε

ελεύθερης ολοκλήρωσης (μη κανονικά) $\gamma = nonreg$. Η λίστα με τα κριτήρια απόδοσης είναι ατελέσφορη, οπότε παρουσιάζονται ενδεικτικά τα πιο συνήθη κριτήρια με την αντίστοιχη σημειογραφία τους: (1) Ελαχιστοποίηση της διάρκειας του έργου: $\gamma = C_{max}$, (2) Ελαχιστοποίηση της καθυστέρησης του έργου (*lateness*): $\gamma = L_{max}$, (3) Ελαχιστοποίηση της βραδύτητας του έργου (*tardiness*): $\gamma = T_{max}$, (4) Ελαχιστοποίηση των καθυστερημένων δραστηριοτήτων: $\gamma = n_T$, (5) Μεγιστοποίηση της καθαράς παρούσας αξίας του έργου: $\gamma = npv$, (6) Ελαχιστοποίηση των κοστών των απαιτήσεων σε πόρους: $\gamma = rac$, (7) Ελαχιστοποίηση των απαιτήσεων σε πόρους για την επίτευξη της χρονικής διορίας: $\gamma = av$, (8) Βελτιστοποίηση της αναμενόμενης τιμής του κριτηρίου απόδοσης: $\gamma = E[.]$.

2.3 Πρόβλημα Προγραμματισμού Έργων υπό Περιορισμένους Πόρους

Στα τέλη της δεκαετίας του 1950, η ανάπτυξη των τεχνικών Αξιολόγησης και Αναθεώρησης Έργου (*Project Evaluation and Review Technique - PERT*) και Μεθόδου Κρίσιμης Διαδρομής (*Critical Path Method- CPM*), έδιναν τη δυνατότητα στα έργα να αναπαρίστανται με διαγράμματα δικτύων (*network diagrams*). Ωστόσο, ο Προγραμματισμός Έργων με τέτοιες τεχνικές, εστιάζει την προσοχή του στον παράγοντα του χρόνου μη λαμβάνοντας υπόψιν περιορισμούς πόρων ή/και ταμειακών ροών. Πάραυτα, σε πολλές περιπτώσεις στην πραγματική ζωή, η μη διάθεση των απαιραίτητων πόρων μιας δραστηριότητας στο προγραμματισμένο χρονικό διάστημα υλοποίησής της, αποτελεί τη γενεσιουργό αιτία χρονικών καθυστερήσεων στην εκτέλεσή της (Montoya-Torres et al., 2010). Το συγκεκριμένο πρόβλημα, αποτελεί και τον προθάλαμο όπου εκκολάφθηκε το πρόβλημα του Προγραμματισμού Έργων υπό Περιορισμένους Πόρους, γνωστό στη διεθνή βιβλιογραφία με το ακρωνύμιο RCPSP (*Resource Constrained Project Scheduling Problem*).

Το RCPSP, με την άνωθι υπογραμμισμένη γενεσιουργό αιτία, εμφανίστηκε στο επιστημονικό προσκήνιο από τους πρωτοπόρους (Kelley Jr, 1963, Wiest, 1963), και από τότε απασχολεί πλήθος ερευνητών. Στην κλασική του μορφή, έγκειται στον προγραμματισμό των δραστηριοτήτων ενός έργου με περιορισμούς σχέσεων προτεραιότητας τέλους-αρχής με μηδενικές καθυστερήσεις, καθώς και περιορισμούς σταθερής διαθεσιμότητας ανανεώσιμων πόρων, με κύριο στόχο την ελαχιστοποίηση της συνολικής διάρκειας του έργου (*project makespan*) (Herroelen et al., 1998). Η έκβαση ενός επιλύσιμου προβλήματος, παρέχει ένα χρονοπρόγραμμα που δηλώνει τον ακριβή χρόνο που θα πραγματοποιηθεί η έναρξη της κάθε δραστηριότητας του έργου.

Κάνοντας χρήση της σημειογραφίας που παρουσιάστηκε (Κεφάλαιο 2.2.2), το RCPSP συμβολίζεται ως: $m, 1|cpm|C_{max}$ (Herroelen et al., 1999). Αποτελεί ένα γενικό πρόβλημα προγραμματισμού με πολλές πρακτικές εφαρμογές σε διαφορετικά πεδία˙ από παραγωγικές διαδικασίες, σιδηροδρομικές και αεροπορικές εταιρείες και κατασκευαστικά έργα, μέχρι καθημερινές εφαρμογές όπως η δημιουργία σχολικών ή άλλων χρονοπρογραμμάτων. Χαρακτηριστικά αναφέρεται ότι και το πρόβλημα προγραμματισμού μηχανών (*machine scheduling*), αποτελεί συνιστώσα του RCPSP (Brucker, 1999). Οι διαφορετικές εκφάνσεις εφαρμογής, υπογραμμίζουν ότι το RCPSP και όλες οι επεκτάσεις του αποτελούν μια πολύτιμη συλλογή ισχυρών εργαλείων που δίνουν τη δυνατότητα προσδιορισμού προβλημάτων βελτιστοποίησης από άλλα γνωστικά αντικείμενα και επίλυσής αυτών, μέσω των μεθόδων επίλυσης που έχουν εφαρμοστεί στο RCPSP (Hartmann and Briskorn, 2010). Το γεγονός αυτό, σε συνδυασμό με την πολυπλοκότητα επίλυσής του, αφού αποτελεί ένα πρόβλημα συνδυαστικής βελτιστοποίησης NP-hard (Blazewicz et al., 1983), σκιαγραφεί τη

σημαντικότητα του RCPSP και αιτιολογεί τόσο την εκτεταμένη υφιστάμενη βιβλιογραφία και συνεχή προσέλκυση ερευνητών, όσο και την καθιέρωσή του ως ένα κλασικό πρόβλημα προγραμματισμού.

2.3.1 Ορισμός του προβλήματος

Το πρόβλημα του Προγραμματισμού Έργων με Περιορισμένους Πόρους (RCPSP) ορίζεται ως εξής (Brucker, 1999, Christofides et al., 1987, Demeulemeester and Herroelen, 2002):

- Το υπό προγραμματισμό έργο, αποτελείται από n δραστηριότητες $i = 1, \dots, n$ και από r ανανεώσιμους πόρους $k = 1, \dots, r, k \in K$.
- Ορίζονται δύο *βοηθητικές δραστηριότητες* (*dummy activities*), η βοηθητική δραστηριότητα αρχής $i = 0$ (*dummy source activity*) και η βοηθητική δραστηριότητα τέλους $i = n + 1$ (*dummy sink activity*), οι απαιτήσεις των οποίων σε πόρους, καθώς και οι διάρκειές τους λαμβάνουν μηδενική τιμή, ενώ αναπαριστούν την έναρξη και τη λήξη του έργου αντίστοιχα.
- Κάθε δραστηριότητα $i = 1, \dots, n$ έχει σταθερή δεδομένη διάρκεια d_i, μετρούμενη σε ακέραιο αριθμό χρονικών περιόδων, υποδεικνύοντας τον απαιτούμενο χρόνο ως την ολοκλήρωσή της. Η διάρκεια κάθε δραστηριότητας συμπεριλαμβάνει και ενδεχόμενους χρόνους προετοιμασίας (*ready times*).
- Δεν επιτρέπεται η προεκχώρηση (*preemption*), γεγονός που υποδηλώνει ότι από τη στιγμή της έναρξης μιας δραστηριότητας δεν επιτρέπεται η διακοπή της μέχρι και την ολοκλήρωσή της (χρόνος λήξης = χρόνος έναρξης + διάρκεια).
- Η διαθεσιμότητα του ανανεώσιμου πόρου k κάθε χρονική στιγμή, είναι σταθερή καθ'όλη τη διάρκεια του έργου και ορίζεται ως R_k, ενώ η ανά περίοδο απαιτούμενη δεδομένη ποσότητα σε μονάδες πόρου k από την δραστηριότητα i, είναι σταθερή καθ'όλη τη διάρκεια εκτέλεσής της και ορίζεται ως r_{ik}. Σημειώνεται, ότι ο εκθέτης ρ, που δηλώνει τον τύπο του ανανεώσιμου πόρου, παραλείπεται χάριν συντομίας, δεδομένης της ύπαρξης μόνο ανανεώσιμων πόρων.
- Οι δεδομένες σχέσεις προτεραιότητας που αναπτύσσονται μεταξύ των δραστηριοτήτων είναι σχέσεις τέλους-αρχής (*FS*) και αναπαρίστανται ως $i \rightarrow j$, υποδηλώνοντας ότι για την έναρξη της δραστηριότητας j πρέπει να έχει προηγηθεί η λήξη της δραστηριότητας i. Επομένως, για την έναρξη κάθε δραστηριότητας πρέπει όλες οι προαπαιτούμενες δραστηριότητές της να έχουν ολοκληρωθεί. Για κάθε δραστηριότητα i, λοιπόν, ορίζεται το σύνολο των προαπαιτούμενων δραστηριοτήτων της (*predecessors*) $Pred(i)$ και των διαδόχων (*successors*) της $Succ(i)$. Σε περίπτωση που το έργο έχει αναπαρασταθεί ως δίκτυο δραστηριοτήτας-σε-κόμβο (*AoN*) ορίζεται το διάνυσμα $G = (V, E)$, όπου το V αποτελεί το σύνολο όλων των δραστηριοτήτων και το $E = \{(i, j) \mid i, j \in V; i \rightarrow j\}$ αναπαριστά τους περιορισμούς προτεραιότητας. Για την δραστηριότητα i ορίζονται, λοιπόν, τα σύνολα των προαπαιτούμενων και των διαδόχων της, αντίστοιχα ως εξής: $Pred(i) := \{j \mid (j, i) \in E\}$ και $Succ(i) := \{j \mid (i, j) \in E\}$.
- Στόχος είναι ο υπολογισμός των χρόνων έναρξης ST_i των δραστηριοτήτων $i = 1, \dots, n$, ώστε:

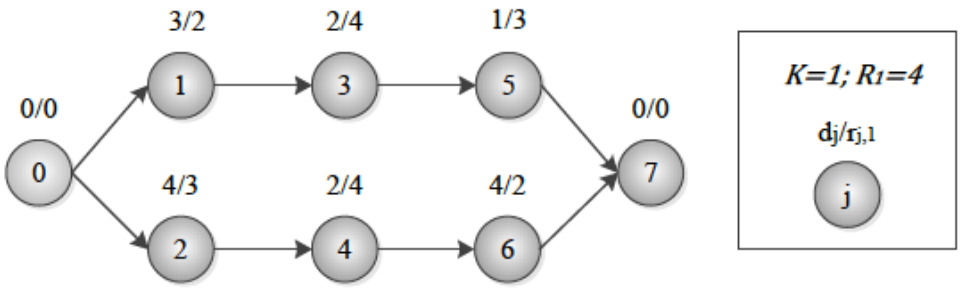
1. Σε κάθε χρονική περίοδο t η συνολική απαίτηση (των δραστηριοτήτων που τη χρονική στιγμή t είναι σε εξέλιξη ή ξεκινούν την υλοποίηση τους) σε κάθε πόρο k, να είναι μικρότερη ή ίση της διαθεσιμότητας R_k του πόρου.
2. Οι δεδομένες σχέσεις προτεραιότητας των δραστηριοτήτων να ικανοποιούνται πλήρως.
3. Η συνολική διάρκεια του έργου $C_{max} = {}_{i=1}^{n}maxFT_i$ να ελαχιστοποιείται, όπου $FT_i = ST_i + d_i$ αποτελεί τη χρονική στιγμή ολοκλήρωσης της δραστηριότητας i. Σημειώνεται, ότι $C_{max} = ST_{n+1} = FT_{n+1}$, δεδομένης της μηδενικής διάρκειας της βοηθητικής δραστηριότητας τέλους, $n + 1$.

- Το διάνυσμα $ST = \overrightarrow{ST_i}$ αποτελεί το παραχθέν χρονοπρόγραμμα (*schedule*), το οποίο θεωρείτω εφικτό (*feasible schedule*), αν ικανοποιούνται οι περιορισμοί των σχέσεων προτεραιότητας και της διαθεσιμότητας κάθε ανανεώσιμου πόρου. Τέλος, σημειώνεται ότι όλα τα δεδομένα (δραστηριότητες, διάρκειες, απαιτήσεις σε πόρους και σχέσεις προτεραιότητας) είναι διαθέσιμα, ντετερμινιστικά και ακέραιης τιμής.

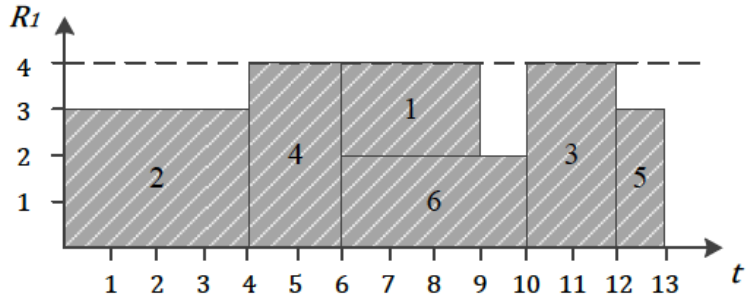
Στη συνέχεια παρουσιάζεται ένα παράδειγμα RCPSP προβλήματος με έξι δραστηριότητες και έναν ανανεώσιμο πόρο (Kolisch and Hartmann, 1999), τα δεδομένα του οποίου παρουσιάζονται στον Πίνακα 1. Το δίκτυο του έργου απεικονίζεται στο Σχήμα 3 σε AoN αναπαράσταση, όπου δίδεται η δυνατότητα εποπτείας των δραστηριοτήτων του έργου, των συσχετιζόμενων με κάθε μία εξ αυτών παραμετρικών πληροφοριών, καθώς και των σχέσεων προτεραιότητας. Τέλος, στο Σχήμα 4 σκιαγραφείται ένα εφικτό χρονοπρόγραμμα, υποδηλώνοντας τις χρονικές στιγμές έναρξης των δραστηριοτήτων του έργου και την ποσότητα χρησιμοποίησης του ανενώσιμου πόρου ανά χρονική περίοδο.

Πίνακας 1. Παράδειγμα (1) RCPSP: Δεδομένα προβλήματος

Δραστηριότητες	**Προαπαιτούμενες Δραστηριότητες**	**Διάρκεια**	**Απαίτηση σε Πόρο 1 με $R_1 = 4$**
0	-	0	0
1	0	3	2
2	0	4	3
3	1	2	4
4	2	2	4
5	3	1	3
6	4	4	2
7	5,6	0	0



Σχήμα 3. Δίκτυο έργου του Παραδείγματος (1) RCPSP, σε AoN αναπαράσταση



Σχήμα 4. Εφικτό χρονοπρόγραμμα Παραδείγματος (1) RCPSP

Εννοιολογικό Μοντέλο

Ένα εννοιολογικό μοντέλο για το RCPSP, διατυπώνεται ως ένα μοντέλο γραμμικού προγραμματισμού, ως εξής (Talbot and Patterson, 1978, Demeulemeester and Herroelen, 1992, Demeulemeester and Herroelen, 2002) :

$$minFT_{n+1} \quad [3.1]$$

υπό τους περιορισμούς:

$$FT_i \leq FT_j - d_j, \qquad \forall i \in Pred(j) \quad [3.2]$$

$$\sum_{i \in S_t} r_{ik} \leq R_k, \qquad \forall\, k = 1, \dots, r \;\&\; t = 1, \dots, FT_{n+1} \quad [3.3]$$

$$ST_0 = 0, \quad [3.4]$$

$$ST_i \geq 0, \qquad i = 0, \dots, n+1 \quad [3.5]$$

Η αντικειμενική συνάρτηση [3.1] ελαχιστοποιεί τη χρονική περίοδο λήξης της βοηθητικής δραστηριότητας τέλους, η οποία ταυτίζεται και με τη χρονική περίοδο ολοκλήρωσης του έργου. Η εξίσωση [3.2] εκφράζει τις σχέσεις προτεραιότητας τέλους-αρχής μεταξύ των δραστηριοτήτων, όπου κάθε δραστηριότητα j για να ξεκινήσει πρέπει όλες οι προαπαιτούμενες δραστηριότητές της $i \in Pred(j)$ να έχουν ολοκληρωθεί. Η εξίσωση [3.3] εκφράζει την απαίτηση συμμόρφωσης των χρησιμοποιούμενων πόρων κάθε χρονική στιγμή, με τη μέγιστη διαθεσιμότητά τους. Έτσι, για κάθε πόρο $k = 1, \dots, r$, και για κάθε χρονική στιγμή του χρονικού ορίζοντα του έργου $t = 1, \dots, FT_{n+1}$, απαιτείται η συνολική κατανάλωση κάθε πόρου k από τις δραστηριότητες που ανήκουν στο σύνολο S_t, να είναι μικρότερη ή ίση από τη συνολική διαθεσιμότητα R_k. Σημειώνεται ότι το σύνολο S_t αποτελείται από τις δραστηριότητες που τη χρονική στιγμή t βρίσκονται υπό εκτέλεση, επομένως $S_t = \{j \,|j = 1, \dots, n,\ ST_j + 1 \leq t \leq FT_j\}$. Τέλος, οι εξισώσεις [3.4] και [3.5] εκφράζουν την έναρξη της βοηθητικής δραστηριότητας αρχής τη χρονική στιγμή μηδέν (άρα και τη λήξη της μιας και $d_0 = 0$) και την μη αρνητικότητα των χρόνων έναρξης των δραστηριοτήτων, αντίστοιχα.

Το προαναφερθέν γραμμικό μοντέλο δεν υπόκειται σε μια εφικτή επίλυση, καθώς οι εξισώσεις [3.1] ως [3.5] δεν παρουσιάζουν κανένα μηχανισμό για τον υπολογισμό του συνόλου S_t για κάθε χρονική στιγμή (Kolisch, 1995). Για την αντιμετώπιση αυτής της δυσκολίας το RCPSP μοντελοποιείται μαθηματικά με χρηση 0-1 μεταβλητών (Pritsker et al., 1969).

Μαθηματικό Μοντέλο

Μια μαθηματική μοντελοποίηση που προσδιορίζει τον περιορισμό των πόρων με ένα σωστό και επιλύσιμο τρόπο, παρουσιάστηκε από τους Pritsker και λοιπούς (Pritsker et al., 1969). Στη συγκεκριμένη μοντελοποίηση γίνεται χρήση δυαδικών 0-1 μεταβλητών x_{it}, όπου προσδιορίζεται αν η δραστηριότητα i τη χρονική περίοδο t ολοκληρώνεται ή όχι. Ειδικότερα, για κάθε δραστηριότητα i και για κάθε δυνατή χρονική περίοδο ολοκλήρωσης $t \in [EF_i, LF_i]$ η μεταβλητή λαμβάνει την τιμή $x_{it} = 1$ αν ολοκληρώνεται τη χρονική περίοδο t, ενώ σε αντίθετη περίπτωση $x_{it} = 0$.

Η μεταβλητή x_{it} μπορεί να προσδιοριστεί μόνο στο χρονικό διάστημα (*time window*) μεταξύ νωρίτερης δυνατής (*early finish - EF*) και βραδύτερης δυνατής (*late finish - LF*) ολοκλήρωσης. Σημειώνεται ότι τα EF και LF, υπολογίζονται με τις κλασικές προς τα εμπρός και πίσω (*forward and backward*) διαδικασίες υπολογισμού στο γράφο G, όπως υλοποιείται και στην τεχνική CPM, θέτοντας $EF_0 = LF_0 = 0$ και $EF_{n+1} = LF_{n+1} = T_{max}$, όπου T_{max} είναι ο χρονικός ορίζοντας ολοκλήρωσης του έργου (Demeulemeester and Herroelen, 2002). Η χρονική περίοδος ολοκλήρωσης, λοιπόν, της δραστηριότητας i υπολογίζεται ως $\sum_{t=EF_i}^{LF_i} t \cdot x_{it}$. Η προτεινόμενη μαθηματική μοντελοποίηση (Pritsker et al., 1969) έχει ως εξής:

$$min \sum_{t=EF_{n+1}}^{LF_{n+1}} t \cdot x_{n+1,t} \qquad [3.6]$$

υπό τους περιορισμούς:

$$\sum_{t=EF_i}^{LF_i} x_{it} = 1, \ \forall\, i = 0, \dots, n+1 \qquad [3.7]$$

$$\sum_{t=EF_i}^{LF_i} t \cdot x_{it} \le \sum_{t=EF_j}^{LF_j} t \cdot x_{jt} - d_j, \qquad \forall i \in Pred(j) \qquad [3.8]$$

$$\sum_{i=0}^{n+1} \sum_{q=\max\{t, EF_i\}}^{\min\{t+d_i-1, LF_i\}} r_{ik} \cdot x_{iq} \le R_k, \quad \forall\, k = 1, \dots, r \ \&\ t = 1, \dots, T \qquad [3.9]$$

$$x_{it} \in \{0,1\}, \qquad \forall\, i = 0, \dots, n+1 \ \&\ t = EF_i, \dots, LF_i \qquad [3.10]$$

Η αντικειμενική συνάρτηση [3.6] ελαχιστοποιεί τη χρονική περίοδο λήξης της βοηθητικής δραστηριότητας τέλους, η οποία ταυτίζεται και με τη χρονική περίοδο ολοκλήρωσης του έργου. Η παράσταση [3.7] εκφράζει την απαίτηση ύπαρξης μόνο μίας χρονικής περιόδου ολοκλήρωσης κάθε δραστηριότητας, η οποία πρέπει να βρίσκεται εντός του διαστήματος [EF,LF] της δραστηριότητας. Η παράσταση [3.8] εκφράζει τις σχέσεις προτεραιότητας τέλους-αρχής μεταξύ των δραστηριοτήτων, όπου κάθε δραστηριότητα j για να ξεκινήσει πρέπει όλες οι προαπαιτούμενες δραστηριότητές της $i \in Pred(j)$ να έχουν ολοκληρωθεί (η χρονική περίοδος ολοκλήρωσης μιας δραστηριότητας δεν πρέπει να υπερβαίνει την χρονική περίοδο έναρξης των διαδόχων της). Η παράσταση [3.9] εκφράζει την απαίτηση συμμόρφωσης των χρησιμοποιούμενων πόρων κάθε χρονική στιγμή, με τη μέγιστη διαθεσιμότητά τους. Έτσι, υπολογίζονται για κάθε χρονική περίοδο t και για κάθε πόρο $k = 1, \dots, r$, όλες οι δραστηριότητες που βρίσκονται υπό εκτέλεση και οι δυνατές χρονικές περίοδοι ολοκλήρωσης τους. Το σταθμισμένο άθροισμα των αντίστοιχων μεταβλητών απόφασης των υπό εκτέλεση δραστηριοτήτων, δεν πρέπει να υπερβαίνει τη διαθεσιμότητα του αντίστοιχου πόρου κάθε χρονική περίοδο t. Τέλος, η παράσταση [3.10] δηλώνει τις μεταβλητές x_{it} ως δυαδικές.

Άλλες μαθηματικές, γραμμικού προγραμματισμού, μοντελοποιήσεις που έχουν παρουσιαστεί στη βιβλιογραφία (Demeulemeester and Herroelen, 2002) , αποτελούν των Mingozzi και λοιποί (Mingozzi et al., 1998), των Alvarez-Valdés και Tamarit (Alvarez-Valdes and Tamarit, 1993) και του Klein (Klein, 2000).

2.3.2 Παραλλαγές και Επεκτάσεις του RCPSP

Μολονότι, το κλασικό Πρόβλημα Προγραμματισμού Έργων υπό Περιορισμένους Πόρους (RCPSP) αποτελεί ένα ισχυρό μοντέλο, δεν έχει τη δυνατότητα να καλύψει όλες τις υφιστάμενες καταστάσεις που λαμβάνουν χώρα σε πρακτικές εφαρμογές. Ως εκ τούτου, πολλοί ερευνητές ανέπτυξαν πιο σύνθετα προβλήματα προγραμματισμού έργων που καλύπτουν διαφορετικές οπτικές προσέγγισης, έχοντας ως βάση το RCPSP, διαφοροποιώντας ένα ή περισσότερα από τα συστατικά του στοιχεία (Hartmann and Briskorn, 2010). Αβίαστα, λοιπόν, δημιουργήθηκε ένα πλήθος παραλλαγών και επεκτάσεων του RCPSP, όπου προτείνονται γενικεύσεις αναφορικά με (Hartmann and Briskorn, 2010):

1. Τη φύση των δραστηριοτήτων του έργου: Ενδέχεται να επιτρέπεται η προεκχώρηση (preemption) όπου οδηγούμαστε στον Προεκχωρητικό Προγραμματισμό (Preemptive Scheduling), να διαφοροποιούνται οι απαιτήσεις σε πόρους ανά χρονική περίοδο (*Resource requests varying with time*), να υφίστανται χρόνοι προετοιμασίας πριν την υλοποίηση μιας δραστηριότητας (*Set-up times*), να υπάρχει η δυνατότητα πολλαπλών τρόπων εκτέλεσης των δραστηριοτήτων (*Multiple Modes*), να προσδιορίζονται εναλλακτικές διάρκειες και αντίστοιχες απαιτήσεις σε πόρους (*Tradeoff problems*), να υφίστανται απαγορευμένες περίοδοι εκτέλεσης των δραστηριοτήτων (*Forbidden Periods*), να λαμβάνουν οι δραστηριότητες συνεχείς τιμές (*Continuous Durations*).

2. Τους χρονικούς περιορισμούς: Ενδέχεται να υπάρχουν γενικευμένες σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων (*Generalized Precedence Constraints*) με ελάχιστα ή/και μέγιστα χρονικά περιθώρια καθυστέρησης (*Minimal/Maximal Time Lags*), να λαμβάνουν χώρα χρονικές στιγμές νωρίτερης δυνατής έναρξης και προθεσμίες ολοκλήρωσης (*Release dates and Deadlines*), να προσδιορίζονται συγκεκριμένα χρονικά διαστήματα εκτέλεσης δραστηριοτήτων και άλλα όπου δεν επιτρέπεται καμία εκτέλεση (*Time-switch constraints*), κ.ά.

3. Τους περιορισμούς σε πόρους: Είναι πιθανό να υπάρχουν ανανεώσιμοι πόροι (*Renewable resources*), μη ανανεώσιμοι πόροι (*Non Renewable resources*), διπλά περιορισμένοι (*Doubly constrained resources*), μερικώς ανανεώσιμοι (*Partially Renewable resources*), σωρρευτικοί (*Cumulative resources*), αφοσιωμένοι σε συγκεκριμένη δραστηριότητα (*Dedicated resources*), συνεχείς (*Continuous resources*), ή πόροι με μεταβλητή χρονικά διαθεσιμότητα (*Resource capacities varying with time*), κ.ά.

4. Τα κριτήρια απόδοσης: Ενδέχεται να υπάρχουν κριτήρια απόδοσης βάσει του χρόνου (*Time-based objectives*), της ευρωστίας (*Robustness-based objectives*), του επαναπρογραμματισμού (*Objectives for rescheduling*), των ανανεώσιμων ή των μη ανανεώσιμων πόρων (*Objectives based on reneable or nonrenewable resources*), των κοστών (*Objectives based on costs*), της καθαρής παρούσας αξίας (*Objectives based on the npv*), ή και συνδυασμός των προαναφερθέντων (*Multiple objectives*).

5. Τον αριθμό των, προς προγραμματισμό, έργων: Ενδέχεται να επιδιώκεται ο προγραμματισμός πολλαπλών έργων ταυτόχρονα (*Multi-Project Scheduling*).

Εν συνεχεία, παρουσιάζονται εκτενώς οι επεκτάσεις του Προβλήματος Προγραμματισμού Έργων υπό Περιορισμένους Πόρους που αποτελούν κομβικά σημεία και αφορούν άμεσα το προς επίλυση πρόβλημα της παρούσας εργασίας.

2.3.2.1 Προγραμματισμός Έργων με Γενικευμένες Σχέσεις Προτεραιότητας

Στην επίλυση του κλασικού RCPSP προβλήματος, για να λάβει χώρα η έναρξη μιας δραστηριότητας πρέπει να έχουν ολοκληρωθεί όλες οι προαπαιτούμενες δραστηριότητές της, λόγω των σχέσεων τέλους-αρχής που τις συνδέουν. Γενικεύοντας τις σχέσεις προτεραιότητας (*Generalized Precedence Relationships - GPRs*) μεταξύ των δραστηριοτήτων του έργου, επιτρέποντας όλους τους τύπους συσχετίσεων πέραν της κλασικής τέλους-αρχής, και εισάγοντας ελάχιστες και μέγιστες χρονικές καθυστερήσεις (*Minimum and Maximum Time Lags*) μεταξύ των χρόνων έναρξης ή λήξης των δραστηριοτήτων, οδηγούμαστε στο Πρόβλημα Προγραμματισμού Έργων υπό Περιορισμένους Πόρους με Γενικευμένες Σχέσεις Προτεραιότητας (*RCPSP-GPRs*), γνωστό στη διεθνή βιβλιογραφία με το ακρωνύμιο $RCPSP/max$ και με τη σημειογραφία $m,1|gpr|C_{max}$ (Herroelen et al., 1999, Hartmann and Briskorn, 2010).

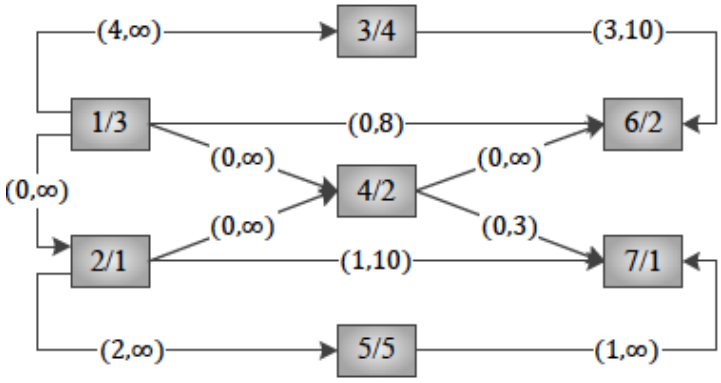
Οι γενικευμένες σχέσεις προτεραιότητας αποτελούν σύχνα ένα πολύτιμο εργαλείο σε πρακτικό επίπεδο, όπως σε περιπτώσεις όπου οι δραστηριότητες απαιτούν καθορισμένους ή αλληλοσυσχετιζόμενους χρόνους έναρξης ή/και λήξης, μη καθυστερημένες εκτελέσεις, υποχρεωτικές χρονικές υπερκαλύψεις μεταξύ τους, χρονικά μεταβαλλόμενη απαίτηση σε πόρους, χρόνους προετοιμασίας, προθεσμίες κ.ά. (De Reyck and Herroelen, 1999). Επιπρόσθετα, η χρήση ελάχιστων χρονικών προθεσμιών λαμβάνουν χώρα π.χ. σε περιπτώσεις συναρμολόγησης κύριων προϊόντων σε χαμηλότερα και ανώτερα επίπεδα της δομής τους, ενώ παράλληλα η χρήση μέγιστων χρονικών προθεσμιών μπορεί να χρησιμοποιηθεί για τη διαμόρφωση των άνω ορίων του συνόλου της χρονικής ροής των προϊόντων (*flow times of products*), εξυπηρετώντας τη μείωση των εργασιών που βρίσκονται υπό εξέλιξη (Neumann and Schwindt, 1997). Γενικότερα, η χρήση ελάχιστων και μέγιστων χρονικών καθυστερήσεων επιτρέπει τη μοντελοποίηση των προθεσμιών των δραστηριοτήτων, καθώς και της μεταβαλλόμενης απαίτησης και διαθεσιμότητας των πόρων, οδηγώντας σε ένα πιο γενικό μοντέλο RCPSP τύπου $m,1,v\alpha|gpr,\rho_j,\delta_j,vr|C_{max}$ (Herroelen et al., 1999).

Οι γενικευμένες σχέσεις προτεραιότητας (όπως παρουσιάστηκαν στο Κεφάλαιο 2.2.1) διακρίνουν τύπους σχέσεων προτεραιότητας τέλους-αρχής (*Finish-Start, FS*), τέλους-τέλους (*Finish-Finish, FF*), αρχής-αρχής (*Start-Start, SS*) και αρχής-τέλους (*Start-Finish, SF*). Οι χρονικοί περιορισμοί των ελάχιστων και μέγιστων χρονικών καθυστερήσεων υφίστανται μεταξύ των χρονικών περιόδων έναρξης ή/και λήξης μεταξύ των δραστηριοτήτων. Επί παραδείγματι, οι χρονικές καθυστερήσεις μεταξύ των χρονικών περιόδων έναρξης (*start times -ST*) δύο δραστηριοτήτων i και j, ορίζονται ως (Bartusch et al., 1988):

$$ST_i + l_{ij}^{min} \leq ST_j \leq ST_i + l_{ij}^{max} \qquad [3.11]$$

όπου l_{ij}^{min} και l_{ij}^{max} αποτελούν την ελάχιστη και τη μέγιστη χρονική καθυστέρηση, αντίστοιχα, αρχής-αρχής (*start-to-start lag*) της δραστηριότητας j σε σχέση με τη δραστηριότητα i. Η υφιστάμενη ελάχιστη χρονική καθυστέρηση αρχής-αρχής l_{ij}^{min}, υποδηλώνει ότι η δραστηριότητα j δεν μπορεί να ξεκινήσει νωρίτερα από l_{ij}^{min} χρονικές μονάδες μετά από την έναρξη της δραστηριότητας i, ενώ η μέγιστη χρονική καθυστέρηση αρχής-αρχής l_{ij}^{max}, υποδηλώνει ότι η δραστηριότητα j δεν μπορεί να ξεκινήσει αργότερα από l_{ij}^{max} χρονικές μονάδες μετά από την έναρξη της δραστηριότητας i (Ballestín et al., 2011). Το *χρονικό διάστημα «παραθύρου»* (*time window*) του χρόνου έναρξης της δραστηριότητας j αναφορικά με το χρόνο έναρξης της δραστηριότητας i ορίζεται $W_{ij} := [ST_i + l_{ij}^{min}, ST_i + l_{ij}^{max}]$. Ομοίως ορίζονται οι χρονικές καθυστερήσεις τέλους-αρχής, τέλους-τέλους και αρχής-τέλους.

Οι γραφικές αναπαραστάσεις δικτύων με χρονικές καθυστερήσεις συνήθως απαρτίζονται από ένα τετράγωνο για κάθε δραστηριότητα, του οποίου η αριστερή (δεξιά) πλευρά καθορίζει την έναρξη (λήξη) της δραστηριότητας, ενώ οι χρονικές καθυστερήσεις αναπαρίστανται στα τόξα μεταξύ των συσχετιζόμενων πλευρών των τετραγώνων ως $[l_{ij}^{min}, l_{ij}^{max}]$. Στο Σχήμα 5 παρουσιάζεται ένα τυπικό παράδειγμα δικτύου με χρονικές καθυστερήσεις.



Σχήμα 5. Παράδειγμα δικτύου με χρονικές καθυστερήσεις (Bartusch et al., 1988)

Όλες οι γενικευμένες σχέσεις προτεραιότητας με ελάχιστες και μέγιστες χρονικές καθυστερήσεις, δύνανται να μετασχηματιστούν σε μία κοινή σχέση προτεραιότητας αρχής-αρχής, δεδομένου του ότι οι διάρκειες και οι χρόνοι καθυστέρησης λαμβάνουν σταθερές και ντετερμινιστικές τιμές. Ο μετασχηματισμός αυτός γίνεται αντικαθιστώντας τους χρόνους λήξης των δραστηριοτήτων με το άθροισμα των αντίστοιχων διαρκειών και χρόνων έναρξης, ως $FT_i = ST_i + d_i$. Έπειτα, κάθε ανίσωση [3.3.2.1] μετατρέπεται στη μορφή $ST_i + l_{ij} \leq ST_j$, επιτρέποντας την ύπαρξη αρνητικών χρονικών καθυστερήσεων που δηλώνουν τη χρονική επικάλυψη μεταξύ δραστηριοτήτων, σύμφωνα με τους ακόλουθους *κανόνες μετασχηματισμού* (Bartusch et al., 1988):

1. Χρονικές καθυστερήσεις αρχής-αρχής:

$$ST_i + l_{ij}^{min} \leq ST_j \;\rightarrow\; ST_i + \delta_{ij} \leq ST_j \qquad \text{όπου } \delta_{ij} = l_{ij}^{min} \qquad [3.12]$$

$$ST_i + l_{ij}^{max} \geq ST_j \;\rightarrow\; ST_j + \delta_{ji} \leq ST_i \qquad \text{όπου } \delta_{ji} = -l_{ij}^{max} \qquad [3.13]$$

2. Χρονικές καθυστερήσεις αρχής-τέλους:

$$ST_i + l_{ij}^{min} \leq FT_j \;\rightarrow\; ST_i + \delta_{ij} \leq ST_j \qquad \text{όπου } \delta_{ij} = l_{ij}^{min} - d_j \qquad [3.14]$$

$$ST_i + l_{ij}^{max} \geq FT_j \;\rightarrow\; ST_j + \delta_{ji} \leq ST_i \qquad \text{όπου } \delta_{ji} = d_J - l_{ij}^{max} \qquad [3.15]$$

3. Χρονικές καθυστερήσεις τέλους-αρχής:

$$FT_i + l_{ij}^{min} \leq ST_j \;\rightarrow\; ST_i + \delta_{ij} \leq ST_j \qquad \text{όπου } \delta_{ij} = d_i + l_{ij}^{min} \qquad [3.16]$$

$$FT_i + l_{ij}^{max} \geq ST_j \;\rightarrow\; ST_j + \delta_{ji} \leq ST_i \qquad \text{όπου } \delta_{ji} = -d_i - l_{ij}^{max} \qquad [3.17]$$

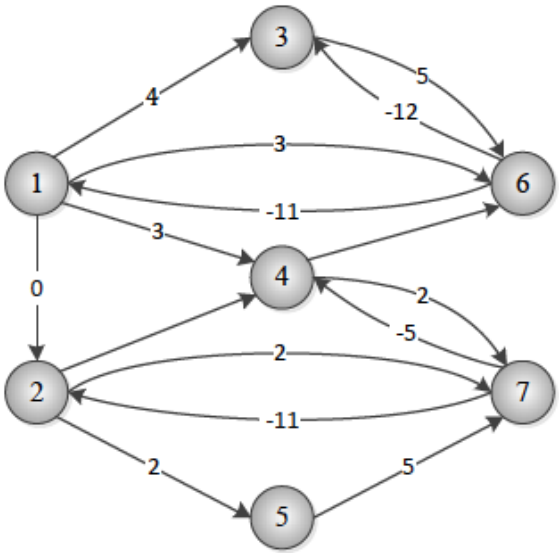
4. Χρονικές καθυστερήσεις τέλους-τέλους:

$$FT_i + l_{ij}^{min} \leq FT_j \;\rightarrow\; ST_i + \delta_{ij} \leq ST_j \qquad \text{όπου } \delta_{ij} = d_i - d_j + l_{ij}^{min} \qquad [3.18]$$

$$FT_i + l_{ij}^{max} \geq FT_j \;\rightarrow\; ST_j + \delta_{ji} \leq ST_i \qquad \text{όπου } \delta_{ij} = d_j - d_i - l_{ij}^{max} \qquad [3.19]$$

Οι άνωθι μετασχηματισμοί, δίνουν τη δυνατότητα αναπαράστασης των χρονικών περιορισμών του έργου ως δίκτυο δραστηριοτήτας-σε-κόμβο (*AoN*) με το διάνυσμα

$G = (V, E)$, γνωστό και ως δίγραφος (*digraph*), να ορίζει ως V το σύνολο όλων των δραστηριοτήτων και ως $E = \{(i,j) \mid i,j \in V; i \to j\}$ το σύνολο των σχέσεων προτεραιότητας αρχής-αρχής με $\delta_{ij} > -\infty$. Σε περίπτωση ύπαρξης σχέσης προτεραιότητας μεταξύ δύο δραστηριοτήτων i και j, το βέλος ένωσης των αντίστοιχων κόμβων φέρει ως τιμή (*weight/length*) τη μέγιστη τιμή δ_{ij}. Ένα χαρακτηριστικό παράδειγμα δίγραφου απεικονίζεται στο Σχήμα 6, το οποίο παρήχθη βάση του δικτύου του Σχήματος 5.



Σχήμα 6. Δίγραφος βάσει του Σχήματος 5 (Bartusch et al., 1988)

Σημειώνεται ότι ένας δίγραφος μπορεί να περιλαμβάνει κυκλικές διαδρομές ($i \to j \to \cdots \to i$), το μήκος των οποίων υπολογίζεται ως το άθροισμα όλων των χρονικών καθυστερήσεων που περιλαμβάνονται στην κυκλική διαδρομή. Αν ένα δίκτυο έργου δεν περιλαμβάνει καμία κυκλική διαδρομή θετικού μήκους, τότε υπάρχουν εφικτά χρονοπρόγραμματα (*time-feasible schedules*). Στο συγκεκριμένο πρόβλημα, εφικτό, από άποψη χρόνου, θεωρείται ένα χρονοπρόγραμμα που ικανοποιεί όλες τις δεδομένες χρονικέςκαθυστερήσεις (Bartusch et al., 1988). Επιπρόσθετα, για κάθε δραστηριότητα i ενός δίγραφου υπάρχει μία διαδρομή από τον κόμβο αρχής (της βοηθητικής δραστηριότητας αρχής, 0) στον κόμβο i μη αρνητικού μήκους, και μία διαδρομή από τον κόμβο i στον κόμβο τέλους (της βοηθητικής δραστηριότητας τέλους, $n+1$) με μήκος κατ' ελάχιστο ίσο με d_i, δηλαδή με τη διάρκεια της δραστηριότητας i (Neumann-Braun et al., 2003).

Το εννοιολογικό μοντέλο του RCPSP/max, κάνοντας χρήση και της σημειογραφίας και του ορισμού που παρουσιάστηκε στο RCPSP (Κεφάλαιο 3.3.1), ορίζεται ως εξής (Ballestín et al., 2011):

$$minFT_{n+1} \qquad [3.20]$$

υπό τους περιορισμούς:

$$ST_j - ST_i \geq \delta_{ij}, \qquad \forall (i,j) \in E \qquad [3.21]$$

$$\sum_{i \in V: ST_i \leq t < ST_i + d_i} r_{ik} \leq R_k, \qquad \forall\, k = 1, \ldots, r \;\&\; 0 \leq t \leq FT_{n+1} \qquad [3.22]$$

$$ST_0 = 0, \qquad [3.23]$$

$$ST_i \geq 0, \qquad \forall i \in V \qquad [3.24]$$

Η αντικειμενική συνάρτηση [3.20] ελαχιστοποιεί τη χρονική περίοδο λήξης της βοηθητικής δραστηριότητας τέλους, η οποία ταυτίζεται και με τη χρονική περίοδο ολοκλήρωσης του έργου. Η εξίσωση [3.21] εκφράζει τις σχέσεις προτεραιότητας αρχής-αρχής μεταξύ των δραστηριοτήτων με τις χρονικές καθυστερήσεις. Η εξίσωση [3.22] εκφράζει την απαίτηση συμμόρφωσης των χρησιμοποιούμενων πόρων κάθε χρονική στιγμή, με τη μέγιστη διαθεσιμότητά τους. Έτσι, για κάθε χρονική στιγμή του χρονικού ορίζοντα του έργου $t = 1, \dots, FT_{n+1}$, οι απαιτήσεις σε κάθε ανανεώσιμο πόρο $k = 1, \dots, r$ των δραστηριοτήτων που είναι υπό εκτέλεση πρέπει να είναι μικρότερες ή ίσες από τη συνολική διαθεσιμότητα R_k^{ρ} του πόρου k. Τέλος, οι εξισώσεις [3.23] και [3.24] εκφράζουν την έναρξη της βοηθητικής δραστηριότητας αρχής τη χρονική στιγμή μηδέν (άρα και τη λήξη της μιας και $d_0 = 0$) και την μη αρνητικότητα των χρόνων έναρξης των δραστηριοτήτων, αντίστοιχα.

Ολοκληρώνοντας, αναφέρεται ότι το $RCPSP/max$, αποτελεί ένα πρόβλημα βελτιστοποίησης πολυπλοκότητας NP-complete, ακόμα και για την εύρεση μιας εφικτής, μη βέλτιστης, λύσης (Bartusch et al., 1988) (Nübel and Schwindt, 1997).

2.3.2.2 Προγραμματισμός Έργων υπό Πολλαπλούς Τρόπους Εκτέλεσης Δραστηριοτήτων

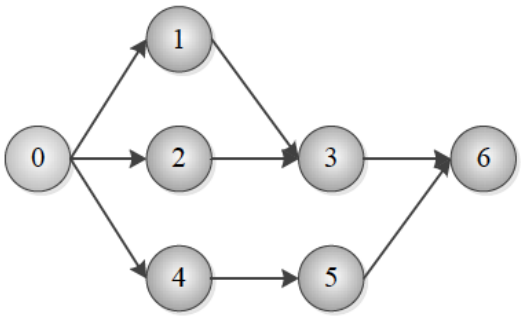
Στην επίλυση του κλασικού RCPSP προβλήματος, κάθε δραστηριότητα μπορεί να εκτελεστεί με ένα μοναδικό τρόπο (*single mode*), ο οποίος ορίζει συγκεκριμένη και σταθερή διάρκεια και απαιτήση σε πόρους. Επιτρέποντας πολλαπλούς ενναλακτικούς τρόπους (*multiple modes*) εκτέλεσης των δραστηριοτήτων, οδηγούμαστε στο Πρόβλημα Προγραμματισμού Έργων υπό Περιορσμένους Πόρους με Πολλαπλούς Τρόπους Εκτέλεσης Δραστηριοτήτων (*Multi-modes-RCPSP*), γνωστό στη διεθνή βιβλιογραφία με το ακρωνύμιο $MRCPSP$ και με τη σημειογραφία $m, 1T|cpm, disc, mu|C_{max}$, για κριτήριο απόδοσης ελαχιστοποίησης της συνολικής διάρκειας του έργου (Herroelen et al., 1999).Το υφιστάμενο πρόβλημα προσδιορίζεται ως εξής: «*δοσμένου ενός συνόλου αλληλοσυσχετιζόμενων δραστηριοτήτων (σχέσεις προτεραιότητας), όπου κάθε δραστηριότητα έχει τη δυνατότητα να υλοποιηθεί με έναν από πολλούς τρόπους εκτέλεσης και κάθε τρόπος χαρακτηρίζεται από μια γνωστή διάρκεια δραστηριότητας και από δεδομένες απαιτήσεις σε πόρους, αναζητείται η χρονική στιγμή έναρξης και ο τρόπος εκτέλεσης κάθε δραστηριότητας, ώστε να βελτιστοποιηθεί ένα συγκεκριμένο κριτηρίο απόδοσης;*» (Boctor, 1990).

Στο MRCPSP κάθε δραστηριότητα μπορεί να υλοποιηθεί με έναν από πολλούς τρόπους εκτέλεσης (Elmaghraby, 1964). Κάθε τρόπος εκτέλεσης, προσδιορίζει τη διάρκεια της δραστηριότητας που αφορά και τις συγκεκριμένες απαιτήσεις σε πόρους για την πραγματοποίησή της υπό το δεδομένο τρόπο. Η ιδέα των πολλαπλών τρόπων εκτέλεσης, έγκειται στην υπόθεση ότι χρησιμοποιώντας περισσότερες μονάδες πόρων, πιο αποδοτικούς τύπους πόρων, ή διαφορετικό συνδυασμό πόρων, είναι πιθανό να μειωθεί η διάρκεια εκτέλεσης μιας δραστηριότητας. Το σύνολο των τρόπων εκτέλεσης μιας δραστηριότητας i, ορίζεται ως M_i, ο κάθε ενναλακτικός τρόπος ως m_i, όπου $1 < m_i < M_i$, και η διάρκεια της δραστηριότητας i υπό τον τρόπο m_i, ως d_{im_i}. Σημειώνεται, ότι σε περίπτωση που μια δραστηριότητα έχει ξεκινήσει να εκτελείται με έναν τρόπο, δεν μπορεί να διακοπεί ή να αλλάξει τρόπο εκτέλεσης.

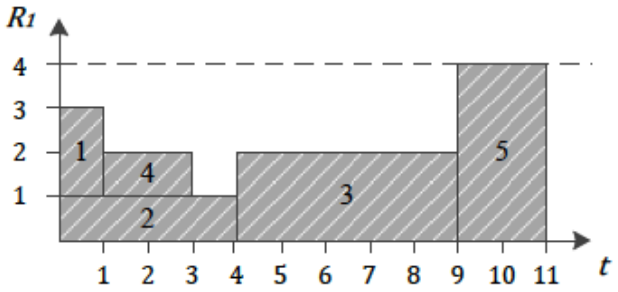
Εν συνεχεία, παρουσιάζεται ένα παράδειγμα MRCPSP προβλήματος, με τον πίνακα των δεδομένων, το δίκτυο του έργου και δύο εφικτά χρονοπρογράμματα ανάλογα με τους επιλεγμένους τρόπους εκτέλεσης κάθε δραστηριότητας:

Πίνακας 2. Παράδειγμα (2) MRCPSP: Δεδομένα προβλήματος

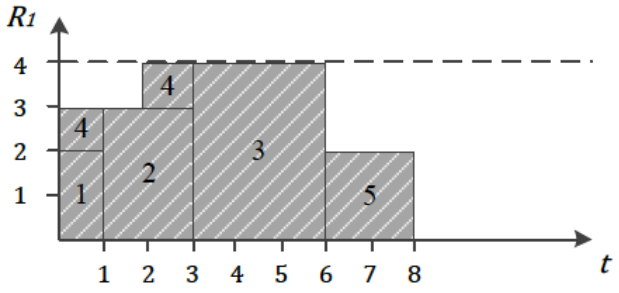
Δραστηριότητες	Προαπαιτούμενες Δραστηριότητες	Τρόποι Εκτέλεσης	Διάρκεια	Απαίτηση σε Πόρο 1 με
0	-	1	0	0
1	0	1	1	2
		2	3	1
2	0	1	4	1
		2	2	3
3	1,2	1	5	2
		2	3	4
4	0	1	2	1
		2	1	4
5	3,4	1	4	1
		2	2	2
6	5	1	0	0



Σχήμα 7. Δίκτυο έργου Παραδείγματος (2) MRCPSP, σε AoN αναπαράσταση



Σχήμα 8 Εφικτό χρονοπρόγραμμα Παραδείγματος (2) MRCPSP με τρόπους εκτέλεσης [1,1,1,1,2,1]



Σχήμα 9. Εφικτό χρονοπρόγραμμα Παραδείγματος (2) MRCPSP με τρόπους εκτέλεσης [1,1,2,2,1,2,1]

Τα είδη των πόρων που χρησιμοποιούνται στο MRCPSP είναι *ανανεώσιμοι, μη ανανεώσιμοι και διπλά περιορισμένοι* (Κεφάλαιο 2.2.1), με τους τελευταίους εξ'ορισμού να μοντελοποιούνται ως ένας ανανεώσιμος και ένας μη ανανεώσιμος πόρος (Patterson et al.,

1989). Το σύνολο των ανανεώσιμων πόρων έχει οριστεί ως R^{ρ}, ενώ η διαθεσιμότητα κάθε ανανεώσιμου πόρου $k \in R^{\rho}$ είναι σταθερή κάθε χρονική περίοδο για όλη τη διάρκεια του έργου, και έχει οριστεί ως R_k^{ρ}. Αντίστοιχα, το σύνολο των μη ανανεώσιμων πόρων συμβολίζεται ως R^{ν}, και κάθε μη ανανεώσιμος πόρος $l \in R^{\nu}$ έχει συνολική διαθεσιμότητα R_l^{ν} για όλο το έργο. Κάθε δραστηριότητα i υπό τον τρόπο m_i, έχει μια σταθερή ανά περίοδο απαιτούμενη ποσότητα σε μονάδες του ανανεώσιμου πόρου k, $r_{im_ik}^{\rho}$, και $r_{im_il}^{\nu}$, σε μονάδες του μη ανανεώσιμου πόρου l, αντίστοιχα.

Στόχος του MRCPSP είναι η εύρεση ενός χρονοπρογράμματος που ελαχιστοποιεί τη διάρκεια του έργου και καθορίζει τόσο τους χρόνους έναρξης των δραστηριοτήτων όσο και την επιλογή των τρόπων εκτέλεσης, ώστε το χρονοπρόγραμμα να ικανοποιεί τους περιορισμούς διαθεσιμότητας των πόρων και των σχέσεων προτεραιότητας των δραστηριοτήτων. Ένα εννοιολογικό μοντέλο για το MRCPSP, κάνοντας χρήση της σημειογραφίας και του ορισμού που παρουσιάστηκε στο RCPSP, διατυπωνεται ως εξής (Hartmann, 2001):

$$minFT_{n+1} \qquad [3.25]$$

υπό τους περιορισμούς:

$$FT_i \leq FT_j - d_{jm_j}, \qquad \forall i \in Pred(j) \qquad [3.26]$$

$$\sum_{i \in S_t} r_{im_ik}^{\rho} \leq R_k^{\rho}, \qquad \forall\, k \in R^{\rho}, \forall\, m_i \in M_i\ , t = 1, \dots, FT_{n+1} \qquad [3.27]$$

$$\sum_{i=0}^{n} r_{im_il}^{\nu} \leq R_k^{\nu}, \qquad \forall\, l \in R^{\nu}, \forall\, m_i \in M_i \qquad [3.28]$$

$$ST_0 = 0, \qquad [3.29]$$

$$ST_i \in int^{+}, \qquad \forall i \in V \qquad [3.30]$$

Η αντικειμενική συνάρτηση [3.25] ελαχιστοποιεί τη χρονική περίοδο λήξης της βοηθητικής δραστηριότητας τέλους, η οποία ταυτίζεται και με τη χρονική περίοδο ολοκλήρωσης του έργου. Η εξίσωση [3.26] εκφράζει τις σχέσεις προτεραιότητας τέλους-αρχής μεταξύ των δραστηριοτήτων με μηδενική χρονική καθυστέρηση. Η εξίσωση [3.27] εκφράζει την απαίτηση συμμόρφωσης των χρησιμοποιούμενων ανανεώσιμων πόρων κάθε χρονική στιγμή, με τη μέγιστη διαθεσιμότητά τους. Έτσι, για κάθε χρονική στιγμή του χρονικού ορίζοντα του έργου $t = 1, \dots, FT_{n+1}$, οι απαιτήσεις σε κάθε ανανεώσιμο πόρο $k = 1, \dots, r$ των δραστηριοτήτων που είναι υπό εκτέλεση ($\in S_t = \{j \,|j = 1, \dots, n,\ ST_j + 1 \leq t \leq FT_j\}$), πρέπει να είναι μικρότερες ή ίσες από τη συνολική διαθεσιμότητα R_k^{ρ} του πόρου k. Η εξίσωση [3.28] εκφράζει την απαίτηση συμμόρφωσης των συνολικών χρησιμοποιούμενων μη ανανεώσιμων πόρων από όλες τις δραστηριότητες, με τη μέγιστη διαθεσιμότητά τους. Έτσι, για κάθε μη ανανεώσιμο πόρο $l \in R^{\nu}$, οι απαιτήσεις σε πόρους όλων των δραστηριοτήτων, για το σύνολο του έργου, πρέπει να είναι μικρότερες ή ίσες από τη συνολική διαθεσιμότητα R_k^{ρ} του πόρου k. Τέλος, οι εξισώσεις [3.23] και [3.24] εκφράζουν την έναρξη της βοηθητικής δραστηριότητας αρχής τη χρονική στιγμή μηδέν (άρα και τη λήξη της αφού $d_0 = 0$) και την μη αρνητικότητα των χρόνων έναρξης των δραστηριοτήτων, αντίστοιχα.

Πριν την έναρξη της διαδικασίας επίλυσης του MRCPSP, πραγματοποιείται μια προεπεξεργαστική διαδικασία (*preprocessing*) απλοποίησης των δεδομένων εισόδου, με στόχο τη μείωση του όγκου των δεδομένων και την επιτάχυνση της αποτελεσμάτικης επίλυσης (Sprecher et al., 1997). Ειδικότερα, η προεπεξεργαστική διαδικασία αποκλείει τρόπους εκτέλεσης δραστηριοτήτων που θεωρούνται μη αποδοτικοί ή μη εκτελέσιμοι, καθώς

και πλεονάζοντες πόρους, χωρίς να επηρεάζει το σύνολο των εφικτών και βέλτιστων λύσεων. Ένας τρόπος εκτέλεσης m_i θεωρείται *μη αποδοτικός* όταν υπάρχει ένας άλλος τρόπος m_i' της ίδιας δραστηριότητας με ίδια ή μικρότερη διάρκεια $d_{im_i'} \leq d_{im_i}$ και περισσότερες απαιτήσεις τόσο σε ανανεώσιμους $r^{\rho}_{im_i'k} \leq r^{\rho}_{im_ik}$, όσο και σε μη ανανεώσιμους πόρους $r^{\nu}_{im_i'l} \leq r^{\nu}_{im_il}$. Ένας τρόπος εκτέλεσης m_i θεωρείται *μη εκτελέσιμος* όταν παραβιάζει τους περιορισμούς διαθεσιμότητας είτε ενός ανανεώσιμου πόρου $r^{\rho}_{im_ik} > R^{\rho}_{k}$ είτε ενός μη ανανεώσιμου πόρου $\sum_{j\neq i} \min\left(r^{\nu}_{jm_jl}\right) + r^{\nu}_{im_il} > R^{\nu}_{l}$. Ένας μη ανανεώσιμος πόρος r^{ν}_{l} θεωρείται *πλεονάζων*, αν το άθροισμα των μέγιστων δυνατών απαιτήσεων για τον συγκεκριμένο πόρο, δεν υπερβαίνει τη συνολική διαθεσιμότητά του, $\sum_j \max\left(r^{\nu}_{jm_jl}\right) \leq R^{\nu}_{l}$ (Sprecher et al., 1997).

Τέλος, το MRCPSP αποτελεί ένα πρόβλημα πολυπλοκότητας NP-hard και σε περιπτώσεις όπου υπάρχουν κατ'ελάχιστο δύο μη ανανεώσιμοι πόροι, η αποτελεσματική εύρεση ενός εφικτού χρονοπρογράμματος με συνέπεια στους περιορισμούς, αποτελεί ένα NP-complete πρόβλημα (Kolisch and Drexl, 1997).

2.3.2.3 Προγραμματισμός Έργων υπό Περιορισμένους Πόρους με Ευέλικτα Προφίλ Πόρων

Στην επίλυση του κλασικού RCPSP προβλήματος, η κατανομή πόρων καθ'όλη τη διάρκεια εκτέλεσης κάθε δραστηριότητας είναι δεδομένη και σταθερή. Επιτρέποντας, η κατανάλωση πόρων να μεταβάλλεται κατά τη διάρκεια εκτέλεσης μιας δραστηριότητας, οδηγούμαστε σε ένα μη σταθερό προφιλ κατανανομής πόρων. Υπό αυτές τις συνθήκες, η απαιτούμενη κατανάλωση πόρων ανά χρονική περίοδο και αντίστοιχα η διάρκεια μιας δραστηριότητας είναι άγνωστες και πρέπει να υπολογιστούν κατά τον προγραμματισμό των δραστηριοτήτων από τους χρόνους έναρξης και λήξης των τελευταίων.

Οι Kolisch και λοιποί (Kolisch et al., 2003), πρότειναν ένα μοντέλο στο οποίο η κατανομή πόρων πρέπει να υπολογιστεί. Ως εκ τούτου, το προφίλ πόρων (*work profile/ resource profile*), δεν αποτελεί πλέον ένα τετράγωνο σχήμα, στο διάγραμμα κατανομής πόρων-χρόνου του χρονοπρογράμματος (ως σταθερή ποσότητα πόρου * διάρκεια), όπως στην κλασική περίπτωση. Εν αντιστοιχία με το προφίλ πόρων, ορίζεται και η «*ποσότητα εργασίας*» (*work content/resource requirement*) ως το συνολικό χρονοποσοτικό μέγεθος του κάθε πόρου που χρειάζεται μια δραστηριότητα για να ολοκληρωθεί (Fündeling and Trautmann, 2010). Επι παραδείγματι, μια ποσότητα εργασίας 10 ανθρωποημερών που απαιτεί μια δραστηριότητα, μπορεί να κατανεμηθεί σε ένα σταθερό προφίλ 2 ανθρώπων για 5 ημέρες, ή σε ένα ευέλικτο προφίλ 3 ανθρώπων για 2 ημέρες και 2 ανθρώπων για 2 ημέρες, ώστε η συνολική ποσότητα εργασίας να έγκειται σε 10 ανθρωποημέρες συνολικά (Naber and Kolisch, 2014b).

Επιτρέποντας την ευέλικτη κατανομή πόρων, οδηγούμαστε στο πρόβλημα Προγραμματισμού Έργων υπό Περιορισμένους Πόρους με Ευέλικτη Κατανομή Πόρων (*RCPSP with Flexible Resource Profiles*), γνωστό με το ακρωνύμιο *FRCPSP*. Το συγκεκριμένο πρόβλημα μελετήθηκε πρώτη φορά από τους Kolisch και λοιποί (Kolisch et al., 2003) για μια εφαρμογή σε ερευνητικά φαρμακευτικά έργα, των οποίων οι δραστηριότητες απαιτούσαν διαφορετικές ομάδες πόρων. Λόγω, της πρόσφατης εισόδου του προβλήματος στο επιστημονικό προσκήνιο, η βιβλιογραφική και ερευνητική επισκόπησή του είναι σε πρώιμο στάδιο. Πάραυτα, κάποιες εφαρμογές του FRCPSP που έχουν λάβει χώρα σε πρακτικό επίπεδο, αποτελούν προβλήματα καθολικού σχεδιασμού και προγραμματισμού έργων, όπως σχεδιασμός δυναμικότητας (rough-cut capacity planning-RCCP, (Hans et al., 2007)), προβλήματα συντριβής έργων (*project crashing*), όπου αυξάνοντας την ποσότητα των

χρησιμοποιούμενων πόρων σε συγκεκριμένες περιόδους μειώνεται η καθυστέρηση του έργου, ή σε προβλήματα προγραμματισμού στον τομέα της υγείας. Επιπρόσθετα, το FRCPSP αποτελεί μια γενική περίπτωση τόσο του προβλήματος MRCPSP και δύναται να μετασχηματιστεί σε αυτό μέσω τη διακριτοποίησης της κατανομής συνεχών πόρων (Kis, 2005), όσο και του προβλήματος διακριτής εναλλαγής χρόνου/πόρου (discrete *time/resource trade-off problem -DTRTP*) -όπου οι διάρκειες εκτέλεσης των δραστηριοτήτων αποτελούν συνάρτηση της ποσότητας των χρησιμοποιούμενων πόρων και είναι σταθερές για όλη την εκτέλεσή τους. Λόγω του ότι το DTRTP αποτελεί ένα ισχυρό NP-hard πρόβλημα (Demeulemeester and Herroelen, 2000), το FRCPSP ως επέκτασή του, αποτελεί επίσης ένα ισχυρό NP-hard πρόβλημα βελτιστοποίησης (Ranjbar and Kianfar).

Το FRCPSP έγκειται στον ταυτόχρονο προσδιορισμό ενός βέλτιστου χρονοπρογράμματος δραστηριοτήτων και της κατανομή των πόρων στις δραστηριότητες του έργου ώστε να ελαχιστοποιηθεί η συνολική διάρκεια του έργου (Naber and Kolisch, 2014a). Κάθε δραστηριότητα απαιτεί, αφότου ξεκινήσει η εκτέλεσή της, ένα μη διακοπτόμενο προφίλ ανά πόρο για τη διάρκεια εκτέλεσής της. Το παραχθέν χρονοπρόγραμμα πρέπει να ικανοποιεί όλες τις απαιτήσεις σχέσεων προτεραιότητας, όπως αναπαρίστανται σε ένα δίκτυο έργου G, τέλους-αρχής με μηδενικές χρονικές καθυστερήσεις μεταξύ των δραστηριοτήτων του έργου. Επιπρόσθετα, τα ευέλικτα προφίλ πόρων πρέπει να ικανοποιούν τους ακόλουθους πρακτικούς κανόνες :

- Η συνολική ποσότητα από καθε απαιτούμενο πόρο που έχει αποδοθεί για κατανάλωση για όλες τις χρονικές περιόδους που αθροίζουν τη διάρκεια κάθε δραστηριότητας, πρέπει να είναι κατ'ελάχιστο ίση με την απαιτούμενη ποσότητα εργασίας της δραστηριότητας από τον αντίστοιχο πόρο. Εμμέσως, γίνεται η υπόθεση ότι η διάρκεια μιας δραστηριότητας είναι το αποτέλεσμα μιας συνάρτησης του μεγέθους χρήσης των πόρων ανά χρονική περίοδο. Όσο περισσότεροι πόροι κατανεμηθούν, τόσο μικρότερη θα είναι και η επικείμενη διάρκεια της δραστηριότητας που αφορούν (Naber and Kolisch, 2014b).
- Πρέπει να υπάρχει ένας ελάχιστος αριθμός χρονικών περιόδων με σταθερή χρησιμοποιούμενη ποσότητα του κάθε πόρου. Αυτός ο περιορισμός, ορίζεται ως *«ελάχιστο μήκος διαστήματος»* (*minimum time block*) (Fündeling and Trautmann, 2010).
- Η χρησιμοποιούμενη ποσότητα πόρων κάθε χρονική περίοδο, πρέπει να βρίσκεται μεταξύ ενός προκαθορισμένου εύρους -μη αρνητικών τιμών- για κάθε πόρο, που ορίζει το ελάχιστο και το μέγιστο όριο κατανάλωσής του ανά χρονική περίοδο.

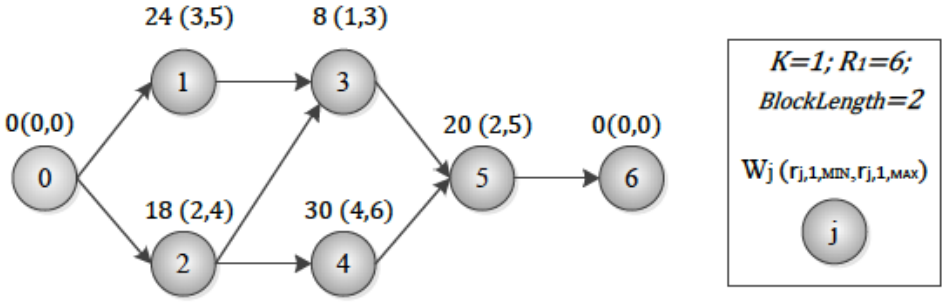
Οι χρησιμοποιούμενοι πόροι είναι συνεχείς, ανανεώσιμοι και ταξινομούνται σε τρεις κατηγορίες, ήτοι κύριοι (*principal*), εξαρτημένοι (*dependent*) και ανεξάρτητοι (*independent*) πόροι (Naber and Kolisch, 2014b). Ένας *κύριος πόρος* μιας δραστηριότητας αποτελεί το βασικό πόρο, του οποίου η χρησιμοποιούμενη ποσότητα μπορεί να εξαρτάται και να αποτελεί υπολογιστική βάση για εξαρτώμενους πόρους. Παρότι διαφορετικοί κύριοι πόροι ενδέχεται να απαιτούνται από διαφορετικές δραστηριότητες, μόνο ένας κύριος πόρος ορίζεται ανά δραστηριότητα. Ένας *εξαρτημένος πόρος*, βασίζεται στον κύριο πόρο της κάθε δραστηριότητας. Η χρησιμοποιούμενη ποσότητα εξαρτάται από την αντίστοιχη του κύριου για την εκτέλεση της κάθε δραστηριότητας. Σημειώνεται, ότι η εξάρτηση μεταξύ των πόρων εκφράζεται μεταξύ μιας γραμμικής συνάρτησης. Επί παραδείγματι, ο εξαρτημένος πόρος k που χρησιμοποιείται από τη δραστηριότητα j τη χρονική στιγμή t, r_{kjt}, ακολουθεί μια γραμμική συνάρτηση της χρησιμοποίησης του κύριου πόρου ρ από τη δραστηριότητα j τη χρονική περίοδο $t, r_{\rho jt}$, όπως: $r_{kjt} = \alpha_{\rho kj} \cdot r_{r\rho t} + \beta_{\rho kj}$. Ένας *ανεξάρτητος πόρος*, δεν

παρουσιάζει καμία εξάρτηση σε σχέση με τη χρησιμοποιούμενη ποσότητά του, όμως η έναρξη και η λήξη κατανάλωσής του πρέπει να ταυτίζεται με τις αντίστοιχες των υπολοίπων πόρων της κάθε δραστηριότητας. Ανεξάρτητα από την κατηγορία, η διαθεσιμότητα όλων των πόρων ανά χρονική περίοδο, είναι σταθερή και ανεξάρτητη (Naber and Kolisch, 2014a).

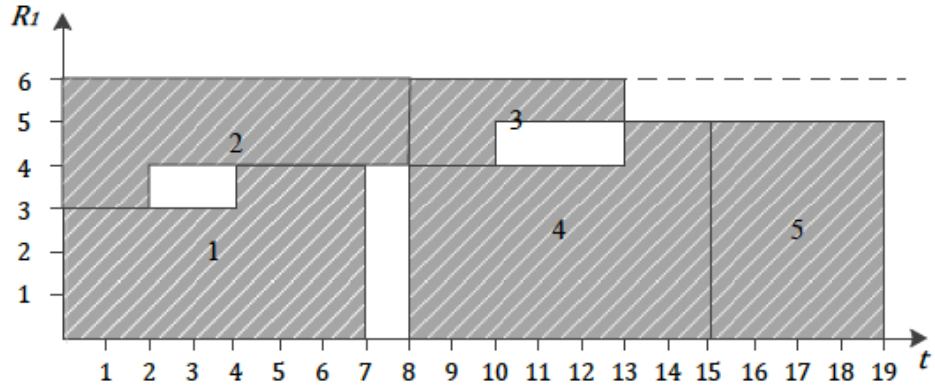
Ένα παράδειγμα FRCPSP παρατίθεται στη συνέχεια, με τον πίνακα των δεδομένων, το δίκτυο του έργου και ένα εφικτό χρονοπρογράμμα (Ranjbar and Kianfar):

Πίνακας 3. Παράδειγμα (3) FRCPSP: Δεδομένα προβλήματος

Δραστηριότητες	Προαπαιτούμενες Δραστηριότητες	Απαιτούμενη Ποσότητα Εργασίας	(min,max) Κατανάλωση Πόρου 1 ανά t, με $R_1 = 6$
0	-	0	(0,0)
1	0	24	(3,5)
2	0	18	(2,4)
3	1,2	8	(1,3)
4	2	30	(4,6)
5	3,4	20	(2,5)
6	5	0	(0,0)



Σχήμα 10. Δίκτυο έργου του Παραδείγματατος (3) FRCPSP, σε AoN αναπαράσταση



Σχήμα 11. Εφικτό χρονοπρόγραμμα Παραδείγματος (3) FRCPSP

Εν συνεχεία παρουσιάζεται η σημειογραφία (Πίνακας 4) και η μαθηματική μοντελοποίηση του FRCPSP, από τους Naber και Kolisch, η οποία έχει εμπνευστεί από το RCPSP μοντέλο του Klein (Klein, 2000), κάνοντας χρήση και της σημειογραφίας που έχει παρουσιαστεί κατά το RCPSP (Κεφάλαιο 2.3.1) (Naber and Kolisch, 2014b):

Πίνακας 4. Σημειογραφία μαθηματικού μοντέλου FRCPSP

E	Σύνολο σχέσεων προτεραιότητας
V	$= \{0, \ldots, n+1\}$, σύνολο δραστηριοτήτων συμπεριλαμβανομένων και

	των βοηθητικών
N	={1, ..., n}, σύνολο δραστηριοτήτων εκτός των βοηθητικών
R	Σύνολο όλων των πόρων
R_j	Σύνολο απαιτούμενων πόρων από τη δραστηριότητα j
T	= {1, ..., T}, σύνολο περιόδων προγραμματισμού
T_j	{ESj, …, LFj}, σύνολο εφικτών περιόδων εκτέλεσης δραστηριότητας j
ST_j	={ESj, …, LSj}, σύνολο εφικτών περιόδων έναρξης δραστηριότητας j
FT_j	= {EFj, …, LFj}, σύνολο εφικτών περιόδων λήξης δραστηριότητας j
$\boldsymbol{\Omega}$	Σύνολο πλειάδων (ρ, k, j), με j∈N; ρ και k οι κύριοι και εξαρτημένοι πόροι της δραστηριότητας j αντίστοιχα
T	=Tmax +1, χρονικός ορίζοντας
T_{max}	Άνω όριο της συνολικής διάρκειας του έργου
T_{min}	Κάτω όριο της συνολικής διάρκειας του έργου
ES_j, LS_j	Νωρίτερη και αργότερη δυνατή έναρξη δραστηριότητας j
EF_j, LF_j	Νωρίτερη και αργότερη δυνατή λήξη δραστηριότητας j
ρ_j	Κύριος πόρος δραστηριότητας j
r_{kjt}	Η ποσότητα του πόρου $r \in R_j$ που έχει κατανεμηθεί για την εκτέλεση της δραστηριότητας j τη χρονική περίοδο $t \in T_j \cup \{ES_j - 1, LF_j - 1\}$
$\alpha_{\rho kj}, \beta_{\rho kj}$	Μεταβλητός και σταθερός συντελεστής γραμμικής συνάρτησης (ρ, k, j) ∈ Ω
R_{kt}	Διαθεσιμότητα του πόρου k τη χρονική περίοδο t
w_{kj}	Απαιτούμενη ελάχιστη ποσότητα εργασίας για την εκτέλεση της δραστηριότητας j
l_{kj}	Ελάχιστο μήκος διαστήματος δραστηριότητας j
$\overline{u_{kj}}, \underline{u_{kj}}$	Άνω και κάτω όριο της απαιτούμενης ποσότητας του πόρου k από τη δραστηριότητα j ανά χρονική περίοδο
$\overline{d_j}, \underline{d_j}$	Άνω και κάτω όριο της διάρκειας της δραστηριότητας j
z_{jt}	$= \begin{cases} 1, \text{αν η δραστηριότητα j είναι υπό εκτέλεση τη χρονική στιγμή t} \\ 0, \text{σε άλλη περίπτωση} \end{cases}$
δ_{kjt}	$= \begin{cases} 1, \text{αν } r_{kjt,-1} \neq r_{kjt} \text{ (αλλάζει από περίοδο t} - 1 \text{ σε t)} \\ 0, \text{σε άλλη περίπτωση} \end{cases}$

Μαθηματικό Μοντέλο:

$$\min C_{max} \quad [3.31]$$

υπό τους περιορισμούς:

$$r_{kjt} - \overline{u_{kj}} \cdot z_{jt} \leq 0, \qquad \forall\, k \in R_j, j \in N, t \in T_j \quad [3.32]$$

$$r_{kjt} - \underline{u_{kj}} \cdot z_{jt} \geq 0, \qquad \forall k \in R_j, j \in N, t \in T_j \quad [3.33]$$

$$r_{kjt} \geq \alpha_{\rho kj} \cdot r_{\rho jt} + \beta_{\rho kj} \cdot z_{jt} \qquad \forall (\rho, k, j) \in \Omega, t \in T_j \quad [3.34]$$

$$\sum_{t \in T_j} r_{kjt} \geq w_{kj}, \qquad \forall\, k \in R_j, j \in N \quad [3.35]$$

$$\sum_{j \in V} r_{kjt} \leq R_{kt}, \qquad \forall\, k \in R_j, t \in T \quad [3.36]$$

$$\sum_{\tau=t}^{t+l_{kj}-1} \delta_{kj\tau} \leq 1, \qquad \forall\, k \in R_j, j \in N, t \in T_j, l_{kj} \geq 2 \quad [3.37]$$

$$r_{kjt} - r_{kj,t-1} - \overline{u_{kj}} \cdot \delta_{kjt} \leq 0 \qquad \forall\, k \in R_j, j \in N, t \in T_j \cup \{LF_j + 1\} \quad [3.38]$$

$$r_{kj,t-1} - r_{kjt} - \overline{u_{kj}} \cdot \delta_{kjt} \leq 0 \qquad \forall\, k \in R_j, j \in N, t \in T_j \cup \{LF_j + 1\} \quad [3.39]$$

$$r_{kj,ES_j-1} = r_{kj,LF_j+1} = 0 \qquad \forall\, k \in R_j, j \in N \quad [3.40]$$

$$r_{kjt} \geq 0 \qquad [3.41]$$

$$\delta_{kjt} \in \{0,1\}, \qquad \forall\, k \in R_j, j \in N, t \in T_j \cup \{LF_j + 1\} \qquad [3.42]$$

$$C_{max} \geq T_{min} \qquad [3.43]$$

$$C_{max} \geq tz_{jt} \qquad \forall j \in N, t \in T_j \qquad [3.44]$$

$$\sum_{t \leq \tau} z_{j\tau} \leq \overline{d_j} \cdot (1 - z_{it}), \qquad \forall\, (i,j) \in E, t \in T_i \cap T_j \qquad [3.45]$$

$$\sum_{\tau \geq t+1} z_{j\tau} + \overline{d_j} \cdot (z_{jy} - z_{j,t+1}) \leq \overline{d_j}, \qquad \forall j \in N, t \in T_j \qquad [3.46]$$

$$z_{jt} \in \{0,1\}, \qquad \forall j \in N, t \in T_j \qquad [3.47]$$

Η αντικειμενική συνάρτηση [3.31] ελαχιστοποιεί τη συνολική διάρκεια του έργου. Οι περιορισμοί [3.32] ως και [3.42] αφορούν τους περιορισμούς των πόρων. Πιο συγκεκριμένα, οι περιορισμοί [3.32] και [3.33] καθορίζουν το άνω και κάτω όριο της χρησιμοποιούμενης ποσότητας των πόρων ανά χρονική περίοδο. Ο περιορισμός [3.34] εκφράζει τη γραμμική σχέση μεταξύ κύριου και εξαρτημένων πόρων της κάθε δραστηριότητας. Ο περιορισμός [3.35] εξασφαλίζει ότι η κατανεμημένη ποσότητα του κάθε πόρου για μία δραστηριότητα, επαρκεί για την ολοκλήρωσή της, δηλαδή είναι μεγαλύτερη ή ίση από την απαιτούμενη ποσότητα εργασίας του κάθε πόρου για τη συγκεκριμένη δραστηριότητα. Ο περιορισμός [3.36] εξασφαλίζει ότι η συνολική χρησιμοποίηση του κάθε πόρου ανά χρονική περίοδο δεν υπερβαίνει τη συνολική διαθεσιμότητα του πόρου. Ο περιορισμός [3.37] δηλώνει το ελάχιστο μήκος διαστήματος όπου δεν δύναται να αλλάξει η χρησιμοποιούμενη ποσότητα του κάθε πόρου, ενώ ο ορισμός της δυαδικής μεταβλητής δ_{kjt} για αλλαγή της χρησιμοποιούμενης ποσότητας, είτε προς τα πάνω είτε προς τα κάτω, από μια χρονική περίοδο στην επόμενη εκφράζεται από τους περιορισμούς [3.38] και [3.39]. Ο περιορισμός [3.40] αρχικοποιεί μηδενική τιμή στην απαιτούμενη ποσότητα πόρων πριν την νωρίτερη δυνατή έναρξη και ύστερα από την αργότερη δυνατή λήξη κάθε δραστηριότητας. Οι περιορισμοί [3.41] και [3.42] ορίζουν τους τύπους των μεταβλητών r_{kjt}, δ_{kjt} αντίστοιχα. Ο περιορισμός [3.43] ορίζει τη θετική ακέραια μεταβλητή T_{max}, ως το κάτω όριο της διάρκεια του έργου, ενώ ο περιορισμός [3.44] εκφράζει την τιμή της διάρκειας του έργου σε σχέση με την τελευταία περίοδο όπου μία μη βοηθητική μεταβλητή είναι υπό εκτέλεση. Ο περιορισμός [3.45] εκφράζει τις σχέσεις προτεραιότητας τέλους-αρχής μεταξύ των δραστηριοτήτων, ενώ ο μη-προεκχωρητικός περιορισμός [3.46] εξασφαλίζει ότι αν μια δραστηριότητα j ολοκληρωθεί τη χρονική περίοδο t, z_{jt} και $z_{jt+1} = 0$, δεν δύναται να ξεκινήσει ξανά ύστερα από τη χρονική περίοδο t, όπου $z_{j\tau} = 0, \forall \tau \geq t + 1$. Δεδομένης της άγνωστης τιμής της διάρκειας κάθε δραστηριότητας στο FRCPSP, η $\overline{d_j}$ ορίζει ένα άνω όριο της διάρκειας της δραστηριότητας. Τέλος ο περιορισμός [3.47] καθορίζει τη φύση της δυαδικής μεταβλητής $z_{j\tau}$.

Η προεπεξεργαστική διαδικασία υπολογισμού των νωρίτερων και αργότερων δυνατών χρόνων έναρξης και λήξης, καθώς και των άνω και κάτω ορίων της ενδεχόμενης διάρκειας κάθε δραστηριότητας, ποικίλει στη βιβλιογραφία. Χαρακτηριστικά παρατίθενται κάποιες τυπικές παρουσιάσεις της προαναφερθείσας διαδικασίας υπολογισμού στις ακόλουθες βιβλιογραφικές πηγές (Naber and Kolisch, 2014b, Ranjbar and Kianfar).

2.3.3 Πολυπλοκότητα

Η πολυπλοκότητα του εκάστοτε προβλήματος βελτιστοποίησης και κατά συνέπεια των προβλημάτων χρονοπρογραμματισμού έργων, όπως το RCPSP και οι επεκτάσεις του, αποτελεί συστατικό στοιχείο του υπό επίλυση προβλήματος. Η θεωρία πολυπλοκότητας (*complexity theory*) παρέχει το μαθηματικό υπόβαθρο, βάσει του οποίου μελετώνται τα υπολογιστικά προβλήματα και εν συνεχεία κατατάσσονται σε *«εύκολα»* ή *«δύσκολα»* (Karp, 1975) (Graham et al., 1979) (Shmoys and Tardos, 1993) (Garey and Johnson, 1979) (Du and Ko, 2011).

Ένα υπολογιστικό πρόβλημα μπορεί να θεωρηθεί ως μια συνάρτηση $f()$, όπου για κάθε είσοδο x εντός του πεδίου ορισμού, δίνει μια έξοδο $f(x)$ εντός του συνόλου τιμών. Η θεωρία πολυπλοκότητας έγκειται στην εκτίμηση του χρόνου που απαιτεί ένας αλγόριθμος για τον υπολογισμό της συνάρτησης $f(x)$, ως συνάρτηση του μήκους της κωδικοποίησης της εισόδου x, και το οποίο συμβολίζεται ως $|x|$. Η αποδοτικότητα ενός αλγορίθμου που υπολογίζει την $f(x)$ για κάθε είσοδο x, μετράται ως ένα άνω όριο $T(n)$ στον αριθμό των βημάτων που υλοποιεί ο αλγόριθμος για κάθε είσοδο x, με $|x|=n$. Σε αρκετές περιπτώσεις, η δυσκολία υπολογισμού με ακρίβεια της συνάρτησης T, οδηγεί στη χρήση της ασυμπτωτικής συμπεριφοράς, ως $T(n)=O(p(n))$ αν υπάρχουν σταθερές $c>0$ και ένας μη αρνητικός ακέραιος n_0, τέτοιοι ώστε $T(n)\leq c \cdot p(n)$, $\forall n \geq n_0$. Ένα πρόβλημα Π θεωρείται *«εύκολο»* αν υπάρχει ένας αλγόριθμος, όπου για την επίλυσή του Π απαιτεί χρόνο εκτέλεσης $T(n) = O(n^k)$, για κάποια σταθερά k. Ως εκ τούτου, η $T(n)$ οριοθετείται από μια πολυωνυμική συνάρτηση του n. Ένας *πολυωνυμικού-χρόνου αλγόριθμος* ή απλά *πολυωνυμικός αλγόριθμος* (*polynomial algorithm*), ορίζεται εκείνος του οποίου η συνάρτηση χρόνου πολυπλοκότητας είναι $O(p(n))$, όπου p κάποιο πολυώνυμο και n το μήκος της εισόδου. Κάθε αλγόριθμος του οποίου η συνάρτηση χρόνου πολυπλοκότητας δεν μπορεί να οριοθετηθεί με αυτό τον τρόπο ορίζεται *ως εκθετικού-χρόνου αλγόριθμος* (*exponential-time algorithm*) (Garey and Johnson, 1979).

Κάθε πρόβλημα προγραμματισμού μπορεί να μοντελοποιηθεί ως ένα πρόβλημα απόφασης, με την υπολογιστική δυσκολία των δύο προβλημάτων να μη φέρει διαφορές. Ως εκ τούτου, η αποδοτική λύση ενός προβλήματος βελτιστοποίησης, δηλαδή σε πολυωνυμικό χρόνο, συνεπάγεται και την αποδοτική επίλυση του αντίστοιχου προβλήματος απόφασης. Επιπρόσθετα, αν ένα πρόβλημα απόφασης είναι υπολογιστικά δύσκολο, τότε και το αντίστοιχο πρόβλημα βελτιστοποίησης θα είναι επίσης δύσκολο (Herroelen et al., 1997). Τα προβλήματα απόφασης κατηγοριοποιύνται ως εξής:

- Η *κατηγορία P (Polynomial time)* αφορά όλα τα προβλήματα απόφασης που μπορούν να επιλυθούν από μια μηχανή Turing (αφηρημένος υπολογιστής), σε χρόνο που οριοθετείται άνωθι από έναν πολυωνυμικού-χρόνου αλγόριθμο για το μήκος της εισόδου.
- Η *κατηγορία NP (Non-deterministic Polynomial time)* αφορά τα προβλήματα απόφασης για τα οποία δεν έχει βρεθεί πολυωνυμικού-χρόνου αλγόριθμος, αλλά υπάρχει η δυνατότητα απόδοσης μιας λύση σε πολυωνυμικό χρόνο. Συνεπάγεται ότι $P \subseteq NP$.

Ορίζεται ως *NP-πλήρες* (*NP-complete*) ένα πρόβλημα που ανήκει στην κατηγορία NP και αν κάθε άλλο πρόβλημα NP μπορεί να αναχθεί σε αυτό, σε πολυωνυμικό χρόνο. Για δύο δεδομένα προβλήματα A και B, θεωρείται οτι το A ανάγεται στο B αν υπάρχει μια πολυωνυμικού-χρόνου υπολογίσιμη συνάρτηση g που μετατρέπει τις εισόδους του A σε

εισόδους του B, ώστε το x να είναι είσοδος για το A αν και μόνο αν το $g(x)$ είναι είσοδος για το B. Ορίζεται ως *NP-δύσκολο* (*NP-hard*) ένα πρόβλημα βελτιστοποίησης, αν το αντίστοιχο πρόβλημα απόφασης είναι NP-πλήρες. Η απόδειξη του ότι ένα πρόβλημα βελτιστοποίησης είναι NP-δύσκολο, συνίσταται στην απόδειξη του ότι το αντίστοιχο πρόβλημα απόφασης είναι NP- πλήρες. Ένα πρόβλημα βελτιστοποίησης αποδεικνύεται ότι είναι εύκολο, αν βρεθεί ένας πολυωνυμικού-χρόνου αλγόριθμος βελτιστοποίησης (Demeulemeester and Herroelen, 2002).

Έχει αποδειχθεί ότι το RCPSP ανήκει στην κατηγορία των NP-δύσκολων προβλημάτων (Blazewicz et al., 1983). Ειδικότερα, το πρόβλημα απόφασης που αντιστοιχεί στο RCPSP έχει οριστεί ως NP- πλήρες, 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 τελευταίο πρόβλημα αποδείχθηκε οτι είναι NP- πλήρες (Garey and Johnson, 1979), οπότε το αντίστοιχο πρόβλημα βελτιστοποίησης,δηλαδή το RCPSP, αποτελεί ένα NP-δύσκολο πρόβλημα.

2.4 Μέθοδοι Επίλυσης

Για την επίλυση δύσκολων προβλημάτων συνδυαστικής βελτιστοποίησης υψηλής πολυπλοκότητας, όπως το RCPSP και οι επεκτάσεις του, έχει αναπτυχθεί μια πληθώρα λύσεων και διαφορετικών προσεγγίσεων, οι οποίες δύνανται να κατηγοριοποιηθούν σε δύο γενικές οικογένειες μεθόδων επίλυσης, ανάλογα με τον τύπο λύσης που παράγουν. Συνοψίζονται σε αναλυτικούς αλγόριθμους (exact algorithms) ή αναλυτικές διαδικασίες (*exact procedures*), που βρίσκουν πάντοτε βέλτιστη λύση και σε προσεγγιστικούς αλγόριθμους (*approximation algorithms*) ή ευρετικές μεθόδους (*heuristic methods*), που παρέχουν προσεγγιστικές λύσεις.

Οι *αναλυτικοί αλγόριθμοι*, για την επίλυση προβλημάτων χρονικού προγραμματισμού, που εφαρμόζονται, έγκεινται σε αλγόριθμους δυναμικού προγραμματισμού (*dynamic programming*), σε αλγορίθμους γραμμικού προγραμματισμού (*linear programming*) και σε τεχνικές περιορισμού και διακλάδωσης (*branch and bound techniques*) (Kolisch and Padman, 2001). Οι μέθοδοι αυτοί απαιτούν υψηλή υπολογιστική προσπάθεια, χρόνο και μνήμη, ειδικά σε μεγάλου μεγέθους προβήματα, καθώς ερευνούν όλο το χώρο λύσεων προκειμένου να καταλήξουν στην καθολική βέλτιστη λύση. Πάραυτα οδηγούν πάντοτε στη βέλτιστη λύση, γεγονός που σκιαγραφεί τη σημαντικότητά τους και την αναμφισβήτητη παρουσία τους στο πεδίο της βελτιστοποίησης προβλημάτων. Χαρακτηριστικά αναφέρεται ότι με άγνωστη την ολική βέλτιστη λύση που παρέχει μια αναλυτική μέθοδος, δεν μπορούν να αξιολογηθούν οι προσεγγιστικές λύσεις που παρέχουν όλες οι ευρετικές μέθοδοι, και κατά συνέπεια αυτές καθ' αυτές οι μέθοδοι.

Οι *ευρετικές μέθοδοι* που εφαρμόζονται για την επίλυση χρονοπρογραμματιστικών προβλημάτων, οι οποίες στηρίζονται στο συσσωρευμένο εμπειρικό επιστημονικό απόφευγμα γνώσης, διακρίνονται σε δομικές μεθόδους (*constructive methods*) ή απλά *ευρετικές* και σε *μεθόδους βελτιστοποίησης τοπικής αναζήτησης* (*local search methods*) ή *μετα-ευρετικές μεθόδοι* (*meta-heuristics*). Μια ευρετική μέθοδος αποτελεί μια λογική ακολουθία βημάτων όπου σταδιακά προγραμματίζονται δραστηριότητες, η σειρά προγραμματισμού των οποίων στηρίζεται σε δεδομένους κανόνες προτεραιότητας, συνυφασμένες με τους υφιστάμενους περιορισμούς, μέχρις ότου δημιουργηθεί ένα εφικτό χρονοπρόγραμμα (Demeulemeester and Herroelen, 2002). Αντίθετα, μια μετα-ευρετική μέθοδος αρχίζει την αναζήτησή της από ένα εφικτό χρονοπρόγραμμα –όχι απαραίτητα το βέλτιστο, που έχει παραχθεί από κάποια δομική

ευρετική μέθοδο, και με κατάλληλες επαναληπτικές διαδικασίες μετασχηματίζει το αρχικό πρόγραμμα σε ένα βελτιωμένο, μέχρις ότου προκύψει ένα τοπικά βέλτιστο χρονοπρόγραμμα. Στην κατηγορία των μετα-ευρετικών μεθόδων ανήκουν αλγόριθμοι όπως οι γενετικοί αλγόριθμοι (*genetic algorithms*), η μέθοδος προσομοιωμένης ανόπτησης (*simulated annealing*), η αναζήτηση ταμπού (*tabu search*) και η βελτιστοποίηση με αποικία μυρμηγκιών (*ant colony optimization*).

Οι ευρετικές μέθοδοι δεν παράγουν απαραίτητα μια βέλτιστη λύση, αλλά μια προσεγγιστική και καλή λύση, ικανή να εφαρμοστεί στην πράξη. Οι μετα-ευρετικές μέθοδοι, ως πολύπλοκες υπολογιστικές διαδικασίες, επιδιώκουν την αποφυγή ενός τοπικού βελτίστου μέσα από την εφαρμογή ευρετικών κανόνων. Επομένως, ενώ οι ευρετικές μέθοδοι συνήθως έχουν μεγαλύτερη πιθανότητα να επικεντρωθούν και να παραμείνουν καθηλωμένες σε ένα τοπικό βέλτιστο, οι μετα-ευρετικές είναι πιο πιθανό να φθάσουν σε μία καλύτερη ή και στη βέλτιστη λύση του προβληματος (Demeulemeester and Herroelen, 2002). Καμία από τις μεθόδους δεν μπορεί να εγγυηθεί την εύρεση της βέλτιστης λύσης, καθώς και ποια ή ποιός συνδυασμός ευρετικών μεθόδων δίνει τις καλύτερες λύσεις σε ένα δεδομένο πρόβλημα. Παρόλα ταύτα, η σημαντικότητα και η πρακτική εφαρμογή τους, έγκεινται στο μειωμένο απαιτούμενο υπολογιστικό χρόνο. Πολλές φορές σε πρακτικές εφαρμογές, η εύρεση μιας καλής -κοντά στη βέλτιστη- λύσης σε σύντομο χρονικό διάστημα είναι προτιμότερη από την εύρεση της βέλτιστης λύσης με μεγάλο απαιτούμενο υπολογιστικό χρόνο, γεγονός που σκιαγραφεί την λογική αιτίαση εστίασης του επιστημονικού ενδιαφέροντος (Herroelen et al., 1998).

2.4.1 Αναλυτικοί Αλγόριθμοι

2.4.1.1 Αλγόριθμος Διακλάδωσης και Περιορισμού (Branch and Bound)

Ο αλγόριθμος Διακλάδωσης και Περιορισμού (*Branch and Bound - B&B*) εμφανίστηκε στο επιστημονικό προσκήνιο από τους Land και Doig (Land and Doig, 1960) και αποτελεί μια μέθοδο διακριτής και συνδυαστικής βελτιστοποίησης, όπου ένα πρόβλημα διασπάται διαδοχικά και με συνεχείς επαναλήψεις, σε μικρότερα υποπροβλήματα -κλαδιά- (*διακλάδωση*), για κάθε ένα από τα οποία εκτιμώνται οι πιθανές λύσεις και αν δεν είναι ικανοποιητικές το «κλαδί» απομονώνεται και αγνοείται για την υπόλοιπη διαδικασία επίλυσης (*περιορισμός*).

Ο B&B αποτελεί ένα ισχυρό εργαλείο, ικανό να επιλύει υπολογιστικά προβλήματα με διαφόρων τύπων περιορισμούς. Κατά την υλοποίησή του, παράγεται μια δενδρική αποτύπωση αποτελούμενη από συνδεδεμένους κόμβους. Οι κόμβοι αποτελούν υποσύνολα του συνόλου των λύσεων του υπό επίλυση προβλήματος. Κάθε *κόμβος* αποτελεί μία πλήρη ή μερική λύση, με τη φύση της τελευταίας να τους διακρίνει σε *κόμβους-φύλλα* (*leaf node),* όπου θεωρείται κάθε ολοκληρωμένη τελική λύση και στη δενδρική απεικόνιση βρίσκονται στο πιο χαμηλό επίπεδο, καθώς και σε *κόμβους-οφθαλμούς* (*bud nodes*), όπου θεωρείται κάθε μερική λύση, ανεξάρτητα από την τελική εφικτότητά της. Επιπρόσθετα, ως *κόμβος-ρίζα* (*root node*) ορίζεται ο αρχικός κόμβος, στην κορυφή του δένδρου, όπου ξεκινά η διαδικασία του B&B και αναπαριστά το σύνολο όλων των λύσεων. Η διαδικασία με την οποία αρχικά ο κόμβος-ρίζα και εν συνεχεία ο κάθε κόμβος-οφθαλμός, διασπάται και παράγει νέα *κλαδιά* ή αλλιώς *κόμβους-παιδιά*, ονομάζεται *διακλάδωση* (*branching*). Για την επιλογή του επόμενου κόμβου από τον οποίο θα ξεκινήσει η διακλάδωση, γίνεται χρήση είτε της στρατηγικής του *πρώτου-καλύτερου κόμβου* (*best-first node*), όπου επιλέγεται ο κόμβος με την καλύτερη τιμή

αντικειμενικής συνάρτησης σε όλη την έκταση του «δένδρου» είτε του σε *βάθος-καλύτερου κόμβου* (*depth-first*), όπου επιλέγεται το παιδί με την κάλυτερη τιμή αντικειμενικής συνάρτησης του τρέχοντα κόμβου (Demeulemeester and Herroelen, 2002).

Για κάθε κόμβο-παιδί, υπολογίζεται το *κάτω όριο* (*lower bound*), ως η τιμή της αντικειμενικής συνάρτησης που το προσδιορίζει. Το κάτω όριο αποτελεί την τιμή της αντικειμενικής συνάρτησης όλων των λύσεων που συμπεριλαμβάνονται σε ένα συγκεκριμένο κόμβο, ενώ παράλληλα όλοι οι κόμβοι-παιδιά που μπορούν να διακλαδωθούν από το συγκεκριμένο κόμβο, θα έχουν ένα καλύτερο κάτω όριο από αυτόν. Αβίαστα εξάγεται το συμπερασμα ότι κανένας κόμβος δεν διακλαδώνεται περαιτέρω, αν το κάτω όριό του είναι χειρότερο από την τιμή της αντικειμενικής συνάρτησης μιας πλήρους λύσης που έχει παραχθεί. Η καλύτερη τιμή αντικειμενικής συνάρτησης του κόμβου-φύλλου που έχει βρεθεί σε κάθε χρονική στιγμή αποτελεί το τρέχον βέλτιστο (*current optimum - CO*) ή το *άνω όριο*, βάσει του οποίου αποφασίζεται αν ένας κόμβος είναι υποσχόμενος για παραγωγή βέλτιστης ή εφικτής λύσης (Brucker and Knust, 2012). Κάθε κόμβος, λοιπόν, με κάτω όριο μεγαλύτερο ή ίσο από το τρέχον άνω όριο, υπόκειται στη διαδικασία του *κλαδέματος* (*pruning*) ή αλλιώς *εμβάθυνσης* (*fathoming*), όπου αποκόπτεται και δεν εξελίσσεται στη συνέχεια.

Τα κύρια χαρακτηριστικά του B&B, που αποτελούν υψίστης σημασίας για την υλοποίησή του και την τελική εύρεση της βέλτιστης λύσης του προβλήματος, είναι τα χαρακτηριστικά διακλάδωσης (*branching characteristics*) και τα χαρακτηριστικά περιορισμού (*bounding characteristics*) (Agin, 1966). Τα χαρακτηριστικά διακλάδωσης εγγυώνται την εύρεση της βέλτιστης λύσης στο τέλος της διαδικασίας B&B, εξασφαλίζοντας το συνυπολογισμό όλων των πιθανών εφικτών συνδυασμών που θα οδηγήσουν σε τελικές λύσεις, κάποια από τις οποίες θα αποτελεί τη βέλτιστη. Τα χαρακτηριστικά περιορισμού προσκομίζουν τη δυνατότητα εντοπισμού μιας βέλτιστης λύσης, χωρίς να απαιτείται η επέκταση κάθε κόμβου και ο υπολογισμός όλων των δυνατών τελικών λύσεων, μέσα από διαδικασίες *«κλαδέματος»* ή απομόνωσης κάποιων μη υποσχόμενων κόμβων-κλαδιών. Επί παραδείγματι, αν εντοπιστεί μία πλήρης εφικτή λύση με τιμή αντικειμενικής συνάρτησης μικρότερη ή ίση από όλα τα κατώ όρια όλων των υφιστάμενων μερικών λύσεων, τότε η συγκεκριμένη πλήρης λύση αποτελεί τη βέλτιστη λύση και δεν συνεχίζεται η αναζήτηση μέσω διακλάδωσης νέων κόμβων (*διαδικασία αξιωματοποίησης*) (Demeulemeester and Herroelen, 2002).

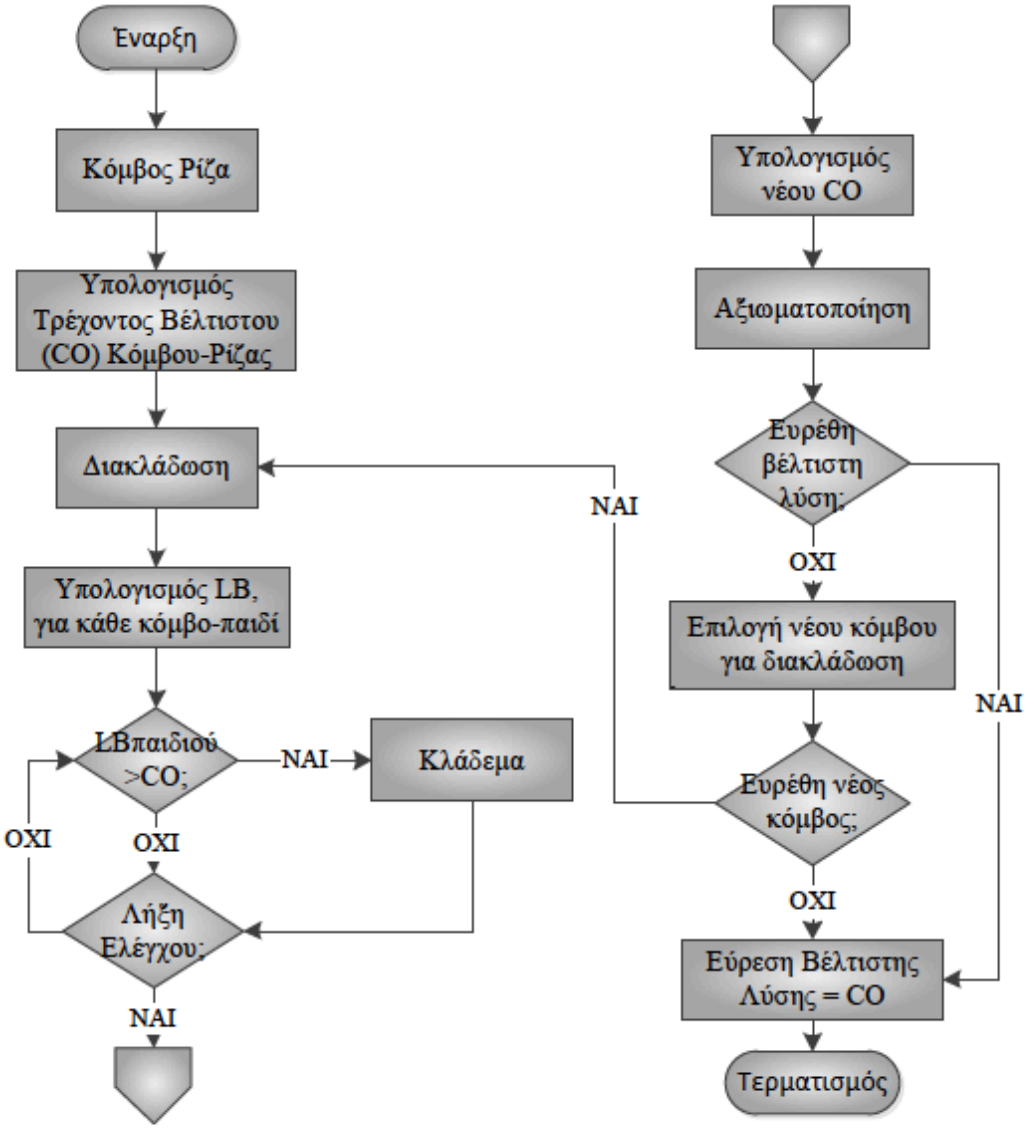
Καθ'όλη τη διάρκεια εκτέλεσης της διαδικασίας του B&B, επιβάλλονται περιορισμοί, οι οποίοι διακρίνονται σε έμμεσους (*implicit*) και άμεσους (*explicit*) (Agin, 1966). Οι έμμεσοι περιορισμοί ικανοποιούνται από τον τρόπο με τον οποίο δομείται και υλοποιείται ο αλγόριθμος, ενώ οι άμεσοι περιορισμοί απαιτούν διαδικασίες για την αναγνώριση και την ικανοποίησή τους ως αναπόσπαστο στοιχείο του αλγορίθμου (Agin, 1966, Demeulemeester and Herroelen, 2002). Χαρακτηριστικό παράδειγμα έμμεσων περιορισμών, σύμφωνα με την προαναφερθείσα διάκριση, αποτελούν οι σχέσεις προτεραιότητας στο RCPSP, ενώ άμεσων περιορισμών αποτελούν οι περιορισμοί διαθεσιμότητας των πόρων (Demeulemeester and Herroelen, 2002).

Συνοψίζοντας, είναι δυνατό να προσδιοριστεί ένας B&B αλγόριθμος μέσα από ένα σύνολο κανόνων που αφορούν (Demeulemeester and Herroelen, 2002):

- Τη διακλάδωση από κόμβους σε νέους κόμβους: δεδομένου του κόμβου-οφθαλμού που θα διακλαδωθεί κάθε φορά, πόσοι και τί νέοι κόμβοι θα παραχθούν.
- Τον υπολογισμό των κάτω ορίων όλων των νέων κόμβων.

- Την επιλογή του επόμενου κόμβου-οφθαλμού που θα υποστεί διακλάδωση.
- Την αναγνώριση των κόμβων που περιλαμβάνουν μόνο μη εφικτές ή μη βέλτιστες λύσεις.
- Την αναγώριση της βέλτιστης λύσης που περιλαμβάνει ένας κόμβος-φύλλο

Εν συνεχεία, παρουσιάζεται ένα τυπικό διάγραμμα ροής (Σχήμα 12), της διαδικασίας που ακολουθείται από ένα τυπικό B&B αλγόριθμο.



Σχήμα 12. Διάγραμμα ροής ενός τυπικού αλγορίθμου B&B

Στη βιβλιογραφία παρουσιάζεται ένας μεγάλος αριθμός B&B αλγορίθμων για τη βελτιστοποίηση του προβλήματος του RCPSP και των επεκτάσεών του, όπως (Pritsker et al., 1969, Davis, 1973, Stinson et al., 1978, Christofides et al., 1987, Bell and Park, 1990). Χαρακτηριστικά αναφέρεται ο B&B αλγόριθμος των Demeulemeester και Herroelen, γνωστός στη βιβλιογραφία με την ονομασία *DH-procedure*, ο οποίος θεωρείται ο πιο αποδοτικός από άποψη υπολογιστικής ταχύτητας (Demeulemeester and Herroelen, 1992).

2.4.2 Ευρετικοί Αλγόριθμοι

2.4.2.1 Μέθοδοι Παραγωγής Χρονοπρογραμμάτων με Κανόνες Προτεραιότητας

Οι μέθοδοι παραγωγής προγραμμάτων (*Schedule Generation Schemes- SGS*) με κανόνες προτεραιότητας (*Priority Rules*), παρότι ανήκουν στις παλαιότερες μεθόδους επίλυσης του RCPSP (Kelley Jr and Walker, 1959), αποτελούν τον πυρήνα της πλειοψηφίας των ευρετικών μεθόδων επίλυσής του. Το γεγονός αυτό έγκειται στο ότι οι χρησιμοποιούμενες μέθοδοι καθίστανται διασθητικές, εύκολες στη χρήση, και γρήγορες από τη σκοπιά της υπολογιστικής προσπάθειας (Kolisch, 1996b).

Ο προγραμματισμός με κανόνες προτεραιότητας, αποτελείται από δύο συστατικά στοιχεία: έναν αλγόριθμο παραγωγής χρονοπρογραμμάτων και έναν κανόνα προτεραιότητας. Δυο διαφορετικοί αλγόριθμοι SGS είναι διαθέσιμοι, οι οποίοι διαχωρίζονται σε σχέση με την υλοποιημένη προσαύξηση που λαμβάνει χώρα. Ο σειριακός αλγόριθμος παραγωγής προγραμμάτων (*serial Schedule Generation Schemes- sSGS*), ο οποίος παρουσιάζει προσαύξηση δραστηριοτήτων (*activity-incrementation*), καθώς και ο παράλληλος αλγόριθμος παραγωγής προγραμμάτων (*parallel Schedule Generation Schemes- pSGS*), ο οποίος παρουσιάζει προσαύξηση χρόνου (*time-incrementation*). Και οι δύο αλγόριθμοι παράγουν ένα εφικτό χρονοπρόγραμμα μέσα από την επέκταση ενός *μερικού προγράμματος* (*partial schedule*) σε διαδοχικά στάδια. Ως μερικό πρόγραμμα ορίζεται αυτό που μόνο ένα υποσύνολο των δραστηριοτήτων έχει ορισμένο χρόνο ολοκλήρωσης. Σε κάθε στάδιο, ο SGS παράγει το σύνολο των δραστηριοτήτων που μπορούν να εκτελεστούν, δεδομένων των περιορισμών πόρων και των σχέσεων προτεραιότητας. Εν συνεχεία, ένας συγκεκριμένος κανόνας προτεραιότητας εφαρμόζεται με στόχο την επιλογή της δραστηριότητας ή των δραστηριοτήτων που θα προγραμματιστούν για εκτέλεση.

Σε περιπτώσεις όπου ο ευρετικός αλγόριθμος παράγει ένα συγκεκριμένο χρονοπρόγραμμα, ορίζεται ως *μέθοδος μονής-διέλευσης* (*single pass method*), ενώ σε αντίθετη περίπτωση ως *μέθοδος πολλαπλών-διελεύσεων* (*multi pass method*). Στην πρώτη περίπτωση, εφαρμόζεται ένας τύπος αλγορίθμου παραγωγής χρονοπρογραμμάτων (sSGS ή pSGS) και ένας κανόνας προτεραιότητας. Στη δεύτερη περίπτωση, η ευρετική διαδικασία επαναλαμβάνεται με ενδεχόμενη χρησιμοποίηση διαφορετικών κανόνων προτεραιότητας ή/και αλγορίθμων προγραμματισμού SGS κάθε φορά (Kolisch and Hartmann, 1999).

Σειριακή Μέθοδος Παραγωγής Χρονοπρογραμμάτων

Ο σειριακός αλγόριθμος παραγωγής χρονοπρογραμμάτων (*serial Schedule Generation Schemes- sSGS*), αποτελείται από $g = 1, \dots, n$ στάδια, σε κάθε ένα από τα οποία μία δραστηριότητα επιλέγεται και προγραμματίζεται για εκτέλεση στη νωρίτερη δυνατή εφικτή χρονική περίοδο έναρξης ικανοποιώντας τους υφιστάμενους περιορισμούς των σχέσεων προτεραιότητας και της διαθεσιμότητας των απαιτούμενων πόρων. Ορίζονται δύο ανεξάρτητα σύνολα δραστηριοτήτων, άμεσα συσχετιζόμενα με το κάθε στάδιο προγραμματισμού: το *σύνολο προγραμματισμού* S_g, όπου εμπεριέχει τις δραστηριότητες που έχουν ήδη προγραμματιστεί και το *σύνολο απόφασης* D_g, όπου εμπεριέχει όλες τις επιλέξιμες δραστηριότητες (*eligible activities*) που έχουν τη δυνατότητα να εκτελεστούν και είναι υποψήφιες για προγραμματισμό στο υφιστάμενο στάδιο. Επιλέξιμη θεωρείται μια απρογραμμάτιστη δραστηριότητα, όπου όλες οι προαπαιτούμενες δραστηριότητές της έχουν προγραμματιστεί και κατά συνέπεια βρίσκονται στο S_g. Σημειώνεται ότι η ένωση των δύο συνόλων δεν παράγει το σύνολο όλων των δραστηριοτήτων, διότι υπάρχουν και μη-

εκτελέσιμες δραστηριότητες, όπως π.χ. δραστηριότητες που δεν έχουν προγραμματιστεί και δεν δύνανται να προγραμματιστούν στο στάδιο g, καθώς δεν έχουν ακόμη ολοκληρωθεί όλες οι προαπαιτούμενες δραστηριότητές τους. Επιπρόσθετα, ορίζεται ως $\overline{R}_k(t) = R_k - \sum_{j \in S_t} r_{jk}$ η παραμένουσα διαθέσιμη ποσότητα του πόρου k τη χρονική περίοδο t, ως $FT_g = \{FT_j | j \in S_g\}$ το σύνολο όλων των χρόνων λήξης των δραστηριοτήτων, και ως $D_g = \{j \in V \setminus S_g | Pred(j) \subseteq S_g\}$ το σύνολο των εκτελέσιμων δραστηριοτήτων (Kolisch and Hartmann, 1999). Ακολούθως παρουσιάζεται ο αλγόριθμος sSGS υπό μορφή ψευδοκώδικα.

Αλγόριθμος 1. *σειριακή* Μέθοδος Παραγωγής Προγραμμάτων (sSGS) (Kolisch and Hartmann, 1999):

$Initialization$: $FT_0 = 0, S_0 = \{0\}$;
for $g = 1$ to n **do**
  Calculate $D_g, FT_g, \overline{R}_k(t) (k = 1, \dots, r, t \in FT_g)$;
  Select one $j \in D_g$;
  $EF_j = max_{h \in Pred(j)}\{FT_h\} + d_j$;
  $FT_j = \min\{t \in [EF_j - d_j, LF_j - d_j] \cap FT_g | r_{jk} \leq \overline{R}_k(\tau), k \in K, \tau \in [t, t + d_j] \cap FT_g\} + d_j$;
  $S_g = S_{g-1} \cup \{j\}$;
end
$FT_{n+1} = max_{h \in Pred(n+1)}\{FT_h\}$;

Σε κάθε στάδιο g, επιλέγεται μια δραστηριότητα από το σύνολο απόφασης D_g σύμφωνα με ένα κανόνα προτεραίοτητας (*priority rule*), που υποδεικνύει ποια από τις επιλέξιμες δραστηριότητες θα προγραμματιστεί. Η τελική επιλεγμένη δραστηριότητα (*selected activity*) προγραμματίζεται στη νωρίτερη δυνατή -υπό τους περιορισμούς των πόρων και των σχέσεων προτεραιότητας- χρονική περίοδο έναρξης. Στη συνέχεια, η επιλεγμένη δραστηριότητα μετακινείται από το σύνολο απόδασης D_g στο σύνολο προγραμματισμού S_g. Αυτή η μετακίνηση, ενδέχεται να προκαλέσει την τοποθέτηση κάποιων δραστηριότητων στο σύνολο απόφασης, δεδομένου του ότι όλες οι προαπαιτούμενες δραστηριότητές τους έχουν πλέον προγραμματιστεί. Η συνολική διαδικασία προγραμματισμού ολοκληρώνεται στο στάδιο $g = n$, όπου όλες οι μη-βοηθητικές δραστηριότητες έχουν προγραμματιστεί.

Πίνακας 5. Επίλυση Παραδείγματος (1) RCPSP, με sSGS

g	*1*	*2*	*3*	*4*	*5*	*6*
D_g	*{1,2}*	*{1,4}*	*{1,6}*	*{3,6}*	*{3}*	*{5}*
j	*2*	*4*	*1*	*6*	*3*	*5*

Ο σειριακός αλγόριθμος παραγωγής χρονοπρογραμμάτων παράγει ενεργά χρονοπρογράμματα (*active schedules*), ανεξαρτήτου επιλεγμένου κανόνα προτεραιότητας (Kolisch, 1996b). Ως ενεργό, ορίζεται ένα εφικτό πρόγραμμα όπου καμία δραστηριότητα δεν μπορεί να υποστεί *τοπική ή καθολική αριστερή μετατόπιση* (*local and global left shift*), δηλαδή να εκτελεστεί νωρίτερα από την προκαθορισμένη χρονική περίοδο έναρξης, χωρίς να καθυστερήσει κάποια άλλη δραστηριότητα ή να παραβιαστεί κάποια σχέση προτεραιότητας (Sprecher et al., 1995). Για προβλήματα προγραμματισμού με χρήση κανονικών κριτηρίων απόδοσης (*regular performance measures*) (Sprecher et al., 1995), όπως η ελαχιστοποίηση της διάρκειας του έργου, η βέλτιστη λύση αποτελεί πάντοτε μέρος του συνόλου των ενεργών προγραμμάτων (Kolisch and Hartmann, 1999). Τέλος, η χρονική πολυπλοκότητα του sSGS είναι $\mathcal{O}(n^2 \cdot K)$ (Pinson et al., 1994).

Παράλληλη Μέθοδος Παραγωγής Χρονοπρογραμμάτων

Ο παράλληλος αλγόριθμος παραγωγής χρονοπρογραμμάτων (*paralell Schedule Generation Schemes- pSGS*), υλοποιεί προσαύξηση χρόνου. Σε κάθε επανάληψη g, υπάρχει ένας *χρόνος προγραμματισμού* t_g. Οι δραστηριότητες που έχουν προγραμματιστεί ως το g είναι είτε στοιχεία του τελικού συνόλου C_g είτε του ενεργού συνόλου A_g. Το *τελικό σύνολο* αποτελείται από όλες τις δραστηριοτητες που έχουν ολοκληρωθεί ως τη χρονική περίοδο t_g και ορίζεται ως $C_g = \{j \in V | FT_j \leq t_g\}$. Το *ενεργό σύνολο* αποτελείται από όλες τις δραστηριότητες που είναι ενεργές -βρίσκονται υπό εξέλιξη- τη χρονική περίοδο t_g και ορίζεται ως $A_g = A(t_g) = \{j \in V | FT_j - d_j \leq t < FT_j\}$. Το *σύνολο απόφασης* D_g αποτελείται από όλες τις δραστηριότητες που μπορούν, από άποψη περιορισμών πόρων και σχέσεων προτεραιότητας, να ξεκινήσουν να εκτελούνται τη χρονική περίοδο t_g και ορίζεται ως $D_g = \{j \in V(C_g \cup A_g) | Pred(j) \subseteq C_g \wedge r_{jk} \leq \overline{R}_k(t_g), (k = 1, \dots, r)\}$. Η παραμένουσα διαθέσιμη ποσότητα του πόρου k τη χρονική περίοδο t_g ορίζεται ως $\overline{R}_k(t) = R_k - \sum_{j \in S_t} r_{jk}$. Στη συνέχεια παρουσιάζεται ο αλγόριθμος pSGS υπό μορφή ψευδοκώδικα:

Αλγόριθμος 2. *παράλληλη* Μέθοδος Παραγωγής Προγραμμάτων (pSGS) (Kolisch and Hartmann, 1999)

$Initialization$: $g = 0, t_g = 0, A_0 = \{0\}, C_0 = \{0\}, \overline{R}_k(0) = R_k$;
while $|A_g \cup C_g| \leq n$ **do**
    g=g+1;
    $t_g = min_{j \in A_g}\{FT_j\}$;
    `Calculate` $C_g, A_g, \overline{R}_k(t_g), D_g$;
    while $D_g \neq 0$ **do**
        Select $j \in D_g$;
        $FT_j = t_g + d_j$;
        `Calculate` $A_g, \overline{R}_k(t_g), D_g$;
    end
end
$FT_{n+1} = max_{h \in Pred(n+1)}\{FT_h\}$;

Κάθε επανάληψη χαρακτηρίζεται από δύο στάδια: (1) Υπολογισμός του νέου χρόνου προγραμματισμού t_g και μετακίνηση των δραστηριοτήτων που έχουν χρόνους ολοκλήρωσης ίσους με το νέο χρόνο προγραμματισμού, από το ενεργό στο τελικό σύνολο. Η μετακίνηση αυτή, ενδέχεται να προκαλέσει την τοποθέτηση κάποιων δραστηριοτήτων στο σύνολο απόφασης, δεδομένου του ότι όλες οι προαπαιτούμενες δραστηριότητές τους έχουν πλέον προγραμματιστεί. (2) Επιλογή μιας δραστηριότητας από το σύνολο απόφασης, μέσω ενός κανόνα προτεραιότητας, και προγραμματισμός της έναρξής της, τη χρονική περίοδο του τρέχοντος χρόνου προγραμματισμού. Κατά συνέπεια, η προγραμματισμένη, πλέον, δραστηριότητα μετακινείται από το σύνολο απόφασης στο ενεργό σύνολο. Το στάδιο (2) επαναλαμβάνεται μέχρις ότου το σύνολο απόφασης καταστεί κενό, γεγονός που σκιαγραφεί ότι οι δραστηριότητες έχουν προγραμματιστεί ή δεν είναι πλέον διαθέσιμες για προγραμματισμό λόγω περιορισμών διαθεσιμότητας πόρων. Ο pSGS ολοκληρώνεται όταν όλες οι δραστηριότητες είναι στο τελικό ή στο ενεργό σύνολο.

Πίνακας 6. Επίλυση Παραδείγματος (1) RCPSP, με pSGS

g	*1*	*2*	*3*	*3*	*4*	*5*	*6*

t_g	0	4	6	6	9	10	12
D_g	{1,2}	{1,4}	{1,6}	{6}	{}	{3}	{5}
j	2	4	1	6		3	5

Ο παράλληλος αλγόριθμος παραγωγής χρονοπρογραμμάτων, με αντίστοιχη του sSGS χρονική πολυπλοκότητα $\mathcal{O}(n^2 \cdot K)$ (Pinson et al., 1994), δημιουργεί χρονοπρογράμματα μη-καθυστέρησης (*non-delay schedules*), ανεξαρτήτου επιλεγμένου κανόνα προτεραιότητας (Kolisch, 1996b). Ως μη-καθυστέρησης, ορίζεται ένα εφικτό χρονοπρόγραμμα όπου, ακόμα και αν η προεκχώρηση επιτρέπεται, καμία δραστηριότητα δεν ενδέχεται να ξεκινήσει νωρίτερα χωρίς να καθυστερήσει κάποια άλλη δραστηριότητα (Sprecher et al., 1995). Σημειώνεται ότι το σύνολο των χρονοπρογραμμάτων μη-καθυστέρησης αποτελεί υποσύνολο του συνόλου των ενεργών χρονοπρογραμμάτων που παράγονται από τον sSGS. Το γεγονός αυτό, σκιαγραφεί ότι τα παραγόμενα, από τον pSGS, χρονοπρογράμματα μη-καθυστέρησης έχουν μικρότερη πληθικότητα (*cardinality*) από τα αντίστοιχα ενεργά του sSGS. Επομένως, ο pSGS αναζητά λύση σε ένα μικρότερο πεδίο λύσεων, με το μειονέκτημα ότι εφαρμόζοντας ένα κανονικό μέτρο απόδοσης ενδέχεται το πεδίο λύσεων να μην περιλαμβάνει τη βέλτιστη λύση, εν αντιθέσει με το πεδίο λύσεων του sSGS που πάντα την συμπεριλαμβάνει (Kolisch, 1996b). Τέλος, υπογραμμίζεται ότι πειραματικές μελέτες, όπως του Kolisch (Kolisch, 1996b), έχουν αντικρούσει την πεποίθηση ότι ο παράλληλος αλγόριθμος είναι γενικά πιο αποδοτικός από τον σειριακό (Alvarez-Valdes and Tamarit, 1993).

Κανόνες Προτεραιότητας

«*Ένας κανόνας προτεραιότητας αποτελεί ένα πλάνο, που εκχωρεί σε κάθε δραστηριότητα j που ανήκει στο σύνολο απόφασης D_g μια συγκεκριμένη τιμή $v(j)$, και προσδιορίζει το κριτήριο επιλογής -μέγιστη ή ελάχιστη τιμή- της δραστηριότητας που θα επιλεχθεί*» (Kolisch and Hartmann, 1999). Σε περιπτώσεις ισοπαλίας, όπου δύο ή περισσότερες δραστηριότητες έχουν την ίδια τιμή, εφαρμόζονται κανόνες διάσπασης της ισοπαλίας. Ο πιο απλός κανόνας διάσπασης ισοπαλίας αποτελεί η επιλογή της δραστηριότητας με τη μικρότερη τιμή ταυτότητας. Στη βιβλιογραφία λαμβάνουν χώρα δεκάδες μελέτες για τους πιο αποδοτικούς κανόνες προτεραιότητας ανά τύπο προβλήματος. Χαρακτηριστικά στον Πίνακα 7 αναφέρονται οι πιο συνήθεις κανόνες προτεραιότητας και οι αντίστοιχες μαθηματικές διατυπώσεις τους.

Πίνακας 7. Κανόνες προτεραιότητας (Kolisch and Hartmann, 1999)

Κανόνας	**Αναφορά**	$\boldsymbol{v(j)}$	**Κριτήριο**
GPRW	*(Alvarez-Valdes and Tamarit, 1989)*	$d_j + \sum_{i \in Succ(j)} d_i$	Max
LFT	*(Davis and Patterson, 1975)*	LF_j	Min
LST	*(Kolisch, 1995)*	$LF_j - d_j$	Min
MSLK	*(Davis and Patterson, 1975)*	$LF_j - EF_j$	Min
MTS	*(Alvarez-Valdes and Tamarit, 1989)*	$\lvert\bar{S}_j\rvert$	Max
RSM	*(Shaffer et al., 1965)*	$max_{(i,j)\in AD}\{0, t_g + d_j - (LF_j - d_j)\}$	Min
SPT	*(Alvarez-Valdes and Tamarit, 1989)*	d_j	Min
WCS	*(Kolisch, 1996a)*	$LF_j - d_j - max_{(i,j)\in AD}\{E(i,j)\}$	Min

Ειδικότερα παρουσιάζονται: η μικρότερη διάρκεια εκτέλεσης (shortest processing time-SPT), ο μέγιστος αριθμός διαδόχων της δραστηριότητας j (most total successors-MTS), η μέγιστη διάρκεια δραστηριότητας συμπεριλαμβανομένων των διαρκειών των άμεσων

διαδόχων της (greatest rank positional weight-GRPW), ο αργότερος χρόνος λήξης (latest finish time-LFT), ο αργότερος χρόνος έναρξης (latest start time-LST), το μικρότερο περιθώριο (minimum slack-MSLK), το χειρότερο πιθανό περιθώριο όπου υπολογίζεται η νωρίτερη δυνατή, υπό τους περιορισμούς πόρων και σχέσεων προτεραιότητας, έναρξη $E(i,j)$ της δραστηριότητας j αν η i ξεκινήσει τη χρονική περίοδο t_g (worst case slack-WCS) και η μέθοδος προγραμματισμού πόρου (resource scheduling method-RSM). Οι δύο τελευταίοι κανόνες κάνουν χρήση του συνόλου των ζευγαριών των δραστηριοτήτων που βρίσκονται στο σύνολο απόφασης $AD = \{(i,j) \in D_g \times D_g | i \neq j\}$.

2.4.3 Μετα-Ευρετικοί Αλγόριθμοι

Οι μετα-ευρετικοί αλγόριθμοι, όπως ορίστηκαν στην εισαγωγή του κεφαλαίου, που λαμβάνουν χώρα για την επίλυση του RCPSP και των επεκτάσεών του, δεν υλοποιούνται πάνω σε ενεργά χρονοπρογραμμάτα (*active schedules*), δηλαδή στα υπαρκτά αυτά καθ' αυτά χρονοπρογράμματα, αλλά σε αναπαραστάσεις αυτών. Κάθε αναπαράσταση μιας λύσης ενός μετα-ευρετικού αλγορίθμου, αποτελεί μια κωδικοποίηση (encoding) και μετατρέπεται σε ένα εφικτό χρονοπρόγραμμα μέσω διαδικασιών αποκωδικοποίησης (*decoding procedures*). Οι πιο συχνά χρησιμοποιούμενες αναπαραστάσεις λύσεων στο RCPSP, συνοψίζονται ως εξής (Kolisch and Hartmann, 1999):

- Λίστα Δραστηριοτήτων (*Activity List*): στην αναπαράσταση λίστας δραστηριοτήτων, μια εφικτή -υπό τους περιορισμούς των σχέσεων προτεραίοτητας- λίστα $\lambda = \{j_1, j_2, \dots, j_n\}$ δίδεται, στην οποία κάθε δραστηριότητα j_g πρέπει να έχει μεγαλύτερης τιμής δείκτη g από κάθε προκάτοχό της που ανήκει στο σύνολο $Pred(j_g)$.
- Τυχαίο Κλειδί Προτεραιότητας (*Random Key of Priority Representation*): στην αναπαράσταση με τυχαία κλειδιά προτεραιότητας, γίνεται χρήση μιας λίστας $\rho = \{r_1, r_2, \dots, r_n\}$ για την απόδοση μιας τυχαίας πραγματικής τιμής σε κάθε δραστηριότητα j. Οι τιμές r_j χρησιμοποιούνται ως προτεραιότητες, καθώς οι δραστηριότητες ταξινομούνται και προγραμματίζονται με τη φθίνουσα σειρά της λίστας ρ.
- Κανόνες Προτεραιότητας (*Priority Rule Representation*): η αναπαράσταση με κανόνες προτεραιότητας, στηρίζεται σε μία λίστα $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$, όπου κάθε π_i αντιπροσωπεύει έναν κανόνα προτεραιότηατας (π.χ. LFT, LST, κ.ό.κ) και κάθε δραστηριότητα i θα προγραμματιστεί σύμφωνα με τον αντίστοιχο κανόνα προτεραιότητας π_i.
- Διάνυσμα Μετατόπισης (*Shift Vector Representation*): στην αναπαράσταση με διάνυσμα μετατόπισης, μία λύση απεικονίζεται από ένα διάνυσμα $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n),\ \sigma_i \in int^+$. Η διαδικασία της αποκωδικοποίησης έγκειται στον υπολογισμό των χρόνων έναρξης ST_j κάθε δραστηριότητας j ως τη μέγιστη τιμή των χρόνων λήξης όλων των προαπαιτούμενων δραστηριοτήτων της συν τη μετατόπισή της, σ_j (Sampson and Weiss, 1993).

2.4.3.1 Γενετικοί Αλγόριθμοι

Οι γενετικοί αλγόριθμοι (*Genetic Algorithms - GA*) αναπτύχθηκαν ως μια μετα-ευρετική στρατηγική επίλυσης δύσκολων προβλημάτων συνδυαστικής βελτιστοποίησης,

όπως το RCPSP και οι επεκτάσεις του, από τον Holland (Holland, 1975). Ανήκουν στην εύρυτερη οικογένεια των εξελικτικών αλγορίθμων (*evolutionary algorithms*) και είναι εμπνευσμένοι από τη φυσική διαδικασία της βιολογικής εξέλιξης και της γενετικής, και ειδικότερα της κληρονομικότητας των γενετικών στοιχείων και της εξέλιξης αυτών από γενιά σε γενιά.

Η διαδικασία επίλυσης ενός GA, ξεκινά από τον υπολογισμό ενός *αρχικού πληθυσμού* (*initial population*) ή αλλιώς πρώτη γενιά. Ο αρχικός πληθυσμός αποτελείται από εφικτές λύσεις παραγόμενες συνήθως από μια ευρετική διαδικασία ή μια τυχαία διαδικασία επιλογής, κάθε μία από τις οποίες είναι κωδικοποιημένη, με έναν από τους προαναφερθέντες τρόπους αναπαράστασης, και αποτελεί ένα *χρωμόσωμα* (*chromosome*) ή απλά ένα *άτομο* (*individual*). Ο αρχικός πληθυσμός, επομένως, περιλαμβάνει POP χρωμοσώματα, όπου POP είναι ο ζυγός ακέραιος αριθμός χρωμοσωμάτων του αρχικού πληθυσμού. Στη συνέχεια, ο αρχικός πληθυσμός χωρίζεται σε τυχαία ζεύγη *ατόμων-γονέων*, σε κάθε ένα από τα οποία εφαρμόζεται η διαδικασία της *διασταύρωσης* (*crossover*), όπου παράγονται δύο νέα *άτομα-απόγονοι* μέσα από κανόνες συνδυασμού των γενετικών στοιχείων των γονέων. Έπειτα, εφαρμόζεται η διαδικασία της *μετάλλαξης* (*mutation*) σε κάθε απόγονο όπου η τυχαία δοσμένη πιθανότητά του είναι μικρότερη από την καθορισμένη πιθανότητα μετάλλαξης $p_{mutation}$. Κατά τη μετάλλαξη λαμβάνουν χώρα τυχαίες αλλαγές στη δομή του χρωμοσώματος του απογόνου. Ύστερα από τις διαδικασίες της εξέλιξης δημιουργείται ένας νέος πληθυσμός μεγέθους $2 \cdot POP$ αποτελούμενος από γονείς και απόγονους. Για κάθε χρωμόσωμα υπολογίζεται η *τιμή καταλληλότητας* (*fitness value*), όπου υποδηλώνει την αξία της λύσης που αφορά, σύμφωνα με την αντικειμενική συνάρτηση του υπό μελέτη προβλήματος. Από το νέο πληθυσμό επιλέγονται POP χρωμοσώματα με την καλύτερη τιμή καταλληλότητας (*selection method*), τα οποία θα αποτελέσουν τον αρχικό πληθυσμό της επόμενης γενιάς λύσεων. Ο αλγόριθμος ολοκληρώνεται όταν επέλθει ο μέγιστος αριθμός επιτρεπόμενων γενεών λύσεων ή όταν ο καθορισμένος αποδεκτός υπολογιστικός χρόνος υπερβεί το επιτρεπτό όριο (*stopping criteria*). Ένας τυπικός γενετικός αλγόριθμος υπό τη μορφή ψευφοκώδικα παρατίθεται στη συνέχεια:

Αλγόριθμος 3. Ψευδοκώδικας ενός τυπικού Γενετικού Αλγορίθμου

```
Set PopulationSize = POP
Set CrossoverType; Set p_mutation ;
Set GenerationCounter g = 0;
Generate InitialPopulation 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
```

Στη βιβλιογραφία λαμβάνει χώρα μια πληθώρα εφαρμογών των γενετικών αλγορίθμων για την επίλυση του RCPSP και των επεκτάσεών του. Χαρακτηριστικά αναφέρονται: (Hartmann, 1998, Hartmann, 2002, Kim et al., 2003, Gutiérrez et al., 2007, Mendes et al., 2009, Montoya-Torres et al., 2010, Wang et al., 2010, Proon and Jin, 2011, Tritschler et al., 2014). Οι γενετικοί αλγόριθμοι αποτελούν ιδιαίτερα αποδοτικές διαδικασίες εύρεσης βέλτιστων, ή κοντά στη βέλτιστη, λύσεων. Κρίσιμος παράγοντας της

αποδοτικότητάς και της αποτελεσματικότητάς τους, αποτελεί η επιλογή των κατάλληλων τρόπων αναπαράστασης των δραστηριοτήτων, της συνάρτησης καταλληλότητας, των διαδικασιών της διασταύρωσης και της μετάλλαξης, καθώς και των μεθόδων επιλογής των κατάλληλων χρωμοσωμάτων για κάθε νέα γενιά (Goldberg, 2002).

2.4.3.2 Αναζήτηση Ταμπού

Η Αναζήτηση Ταμπού (*Tabu Search - TS*) αναπτύχθηκε από τον Glover (Glover, 1989, Glover, 1990) και αποτελεί μία μέθοδο απότομης καθόδου και ομαλής ανόδου με απώτερο στόχο τον απεγκλωβισμό από ενδεχόμενα τοπικά ακρότατα της αντικειμενικής συνάρτησης του υπό επίλυση προβλήματος (Glover and Laguna, 1997).

Η διαδικασία επίλυσης με τη μέθοδο TS, ξεκινά από μία αρχική εφικτή λύση (*initial solution*) η οποία δημιουργεί μια *γειτονιά* (*neighbourhood*) και εν συνεχεία όλες οι παραγόμενες λύσεις της γειτονιάς αξιολογούνται και η καλύτερη επιλέγεται και χρησιμοποιείται στην επόμενη επανάληψη. Η συγκεκριμένη διαδικασία ενδέχεται να οδηγήσει σε κυκλικές κινήσεις γύρω από ένα τοπικό ακρότατο. Για την αποφυγή του προβλήματος αυτού, ένα σύνολο από υλοποιημένες *κινήσεις* (*moves*) αποθηκεύεται σε μία μνήμη της μορφής δομής δεδομένων (*data-structure*), η οποία ορίζεται ως *λίστα ταμπού* (*Tabu List*). Η λίστα ταμπού χρησιμοποιείται για την αποφυγή επανάληψης υλοποιημένων κινήσεων που οδηγούν σε γειτονιές, και κατά συνέπεια σε λύσεις, που έχουν ήδη ερευνηθεί και υπολογιστεί σε ένα συγκεκριμένο αριθμό προηγούμενων επαναλήψεων (τόσες επαναλήψεις όσο και το δεδομένο μέγεθος της λίστας).

Σε περιπτώσεις που η βέλτιστη λύση μιας γειτονιάς που έχει ήδη επισκεφθεί και βρίσκεται στη λίστα ταμπού, είναι καλύτερη από την τρέχουσα βέλτιστη λύση, εφαρμόζεται το *κριτήριο «φιλιδοξίας»* (*Aspiration Criterion or Non-Tabu Criterion*). Το συγκεκριμένο κριτήριο, έχει τη δυνατότητα να απορρίψει και να αναιρέσει τους περιορισμούς που ορίζει η λίστα ταμπού, και να συνεχίσει την αναζήτηση απο τη προαναφερθείσα καλύτερη βέλτιστη λύση της γειτονιάς που έχει επισκεφθεί. Η διαδικασία της αναζήτησης ταμπού ολοκληρώνεται είτε όταν έχει λάβει χώρα ο μέγιστος επιτρεπόμενος αριθμός επαναλήψεων, είτε όταν έχει επέλθει ο μέγιστος αριθμός επιτρεπόμενων επαναλήψεων από τη στιγμή που βρέθηκε καλύτερη λύση, είτε όταν το σύνολο των λύσεων μιας γειτονιάς μετά την αφαίρεση των λύσεων που παραβιάζουν τα κριτήρια της λίστας ταμπού αποτελεί ένα κενό σύνολο. Ακολούθως, παρουσιάζεται ένας τυπικός αλγόριθμος αναζήτησης ταμπού υπό μορφή ψευδοκώδικα:

Αλγόριθμος 4. Ψευδοκώδικας Αναζήτησης Ταμπού (Icmeli and Erenguc, 1994)

```
Generate initial solution x_0, Calculate f(x_0);
Initialise TabuList;
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') ∉ TabuList OR M(x') meets AspirationCriteria then
            Execute move M(x');
            x_current = x'; f_current = f(x');
            Update TabuList;
            Update AspirationCriteria;
```

```
                end
            end
end
```

Η Αναζήτηση Ταμπού, παρότι δεν έχει προσελκύσει μεγάλο αριθμό ερευνητικών προσεγγίσεων και εφαρμογών συγκρτικά με άλλους μετα-ευρετικούς αλγορίθμους, αποτελεί μια μέθοδο επίλυσης προβλημάτων βελτιστοποίησης που έχει να αναδείξει αρκετά ικανοποιητικά αποτελέσματα σε προβλήματα χρονικού προγραμματισμού, όπως το RCPSP και οι επεκτάσεις του, χαρακτηριστικά παραδείγματα των οποίων αποτελούν (Icmeli and Erenguc, 1994, Klein, 2000, Nonobe and Ibaraki, 2002, Mika et al., 2008, Nonobe and Ibaraki, 2003) κ.ά.

2.4.3.3 Προσομειωμένη Ανόπτηση

Η Προσομειωμένη Ανόπτηση (*Simulated Annealing - SA*), εισήχθη στο επιστημονικό προσκήνιο σαν μέθοδος επίλυσης προβλημάτων βελτιστοποίησης από τους Kirkpatrick και λοιπούς (Kirkpatrick et al., 1983) και αποτελεί μια προσομείωση της φυσικής διαδικασίας ανόπτησης των μετάλλων, όπου ένα μέταλλο θερμαίνεται πάνω από την κρίσιμη θερμοκρασία, διατηρείται σε αυτήν για ένα ορισμένο χρονικό διάστημα και εν συνεχεία ψύχεται με απώτερο στόχο την ελαχιστοποίηση των εσωτερικών ατελειών στη δομή των κόκκων του πλέγματος του μετάλλου. Η υλοποίηση της συγκεκριμένης μεθόδου, έλαβε χώρα σε μια προσπάθεια απεγκλωβισμού από ενδεχόμενα τοπικά ακρότατα κατά τη διαδικασία επίλυσης προβλημάτων βελτιστοποίησης.

Η διαδικασία επίλυσης με τη μέθοδο SA, ξεκινά από μία αρχική εφικτή λύση (*initial solution*), η οποία χρησιμοποιείται για την παραγωγή μιας *γειτονιάς* λύσεων (*neighbourhood*), μέσα από μικρές αλλαγές της μορφής της αρχικής λύσης. Κάθε νέα παραχθείσα λύση, γίνεται αποδεκτή και χρησιμοποιείται για τη συνέχιση της διαδικασίας, μόνο αν θεωρείτω καλύτερη σε σχέση με την τρέχουσα λύση. Ενδέχεται, πάραυτα, να γίνει αποδεκτή και μία χειρότερη λύση, βάσει της *πιθανότητας αποδοχής* (*acceptance probability*). Η πιθανότητα αποδοχής εξαρτάται από τη *θερμοκρασία ψύξης* (*cooling temperature*), η οποία αποτελεί μια παράμετρο που αρχικοποιείται με μία τιμή που επιτρέπει την αποδοχή ενός μεγάλου ποσοστού παραγόμενων λύσεων και εν συνεχεία η τιμή της μειώνεται διαδοχικά ώστε να ελαττωθεί ο βαθμός αποδοχής λιγότερο υποσχόμενων λύσεων. Με αυτό τον τρόπο, αποτρέπεται ο εγκλωβισμός του αλγορίθμου σε κάποιο τοπικό ακρότατο σε αρχικά στάδια της υλοποίησής του (Kolisch and Hartmann, 1999). Ο αλγόριθμος ολοκληρώνεται όταν ικανοποιηθεί κάποιο δεδομένο κριτήριο τερματισμού.

Με τον αλγόριθμο SA επιτυγχάνεται υψηλή εξερεύνηση της περιοχής των λύσεων στις αρχικές επαναλήψεις, όπου η πιθανότητα αποδοχής είναι υψηλή και ο αλγόριθμος απεγκλωβίζεται από τα ενδεχόμενα τοπικά ακρότατα, καθώς και υψηλή εκμετάλλευση των αξιόλογων λύσεων στα τελικά στάδια, δεδομένης της μείωσης της «θερμοκρασίας» και της αποδοχής, με απώτερο στόχο την εύρεση της καλύτερης δυνατής ή ακόμα και της βέλτιστης λύσης (Demeulemeester and Herroelen, 2002).

Στη συνέχεια, παρουσιάζεται ένας τυπικός αλγόριθμος Προσομειωμένης Ανόπτησης υπό μορφή ψευδοκώδικα:

Αλγόριθμος 5. Ψευδοκώδικας Προσομειωμένης Ανόπτησης (Eglese, 1990)

```
Calculate initial solution x_0, fitness value f(x_0);
x_current = x_0; f_current = f(x_0);
```

```
Select initial temperature T > 0;
Set temperature change counter t = 0;
while stopping criteria not met do
    Set repetition counter n = 0;
    while n ≠ N(t) do
        Generate neighbour x' of x_current;
        Calculate f'(x_0) and Δ = f(x') − f(x);
        if Δ < 0 then
            x_current = x', f(x_current) = f(x');
        elseif random(0,1) < exp(−Δ/T) then
            x_current = x', f(x_current) = f(x');
        end
    end
    t := t + 1;
    T := T(t);
end
Explore neighbourhood of f_best;
```

Η Προσομειωμένη Ανόπτηση, αποτελεί μια μέθοδο επίλυσης προβλημάτων βελτιστοποίησης με πλήθος εφαρμογών και με αρκετά ικανοποιητικά αποτελέσματα σε προβλήματα χρονικού προγραμματισμού, όπως το RCPSP και οι επεκτάσεις του, χαρακτηριστικά παραδείγματα των οποίων αποτελούν (Cho and Kim, 1997, Boctor, 1996, (Bouleimen and Lecocq, 2003, Bouffard and Ferland, 2007).

3

Ορισμός του υπό μελέτη Προβλήματος

3.1 Περιγραφή του υπό μελέτη Προβλήματος

Ο χρονικός Προγραμματισμός Έργων αποτελεί ένα πρόβλημα υψηλής πολυπλοκότητας, όπου κάθε διαχειριστής αντιμετωπίζει στο αρχικό στάδιο κάθε έργου, ανεξαρτήτου φύσης και αντικειμένου του τελευταίου. Παράλληλα, αποτελεί έναν από τους πιο κρίσιμους παράγοντες επιτυχίας, μιας και αποτελεί την πιο κοινή αιτία αποτυχίας των έργων (Goldratt, 1997, Lewis, 1995).

Ο χρονοπρογραμματισμός έργων στοχεύει στη δημιουργία ενός εφικτού -υπό την έννοια του σεβασμού στους περιορισμούς που αφορούν τη χρήση και διαθεσιμότητα των πόρων, καθώς και των σχέσεων προτεραιότητας μεταξύ των δραστηριοτήτων του έργου- χρονοπρογράμματος, που ελαχιστοποιεί το κριτήριο απόδοσης του προγραμματισμού. Το παραχθέν χρονοπρόγραμμα, αποτελεί το πρόγραμμα αναφοράς (*baseline schedule*), βάσει του οποίου θα εκτελεστεί το έργο, θα κατανεμηθούν οι πόροι, θα προγραμματιστούν εξωτερικές και εσωτερικές δραστηριότητες της εφοδιαστικής αλυσίδας του οργανισμού που αναλαμβάνει το έργο, όπως προμήθεια μηχανημάτων, προληπτική συντήρηση και παράδοση εσωτερικών και εξωτερικών παραγγελιών, θα γίνει η σύναψη των απαραίτητων συμβάσεων και θα καθοριστούν τα χρονικά όρια παράδοσης και προθεσμιών των αποτελεσμάτων του έργου. (Herroelen and Leus, 2005).

Ένα διαχειριστής έργου, κατά το χρονοπρογραμματισμό, πρέπει ταυτόχρονα να διαχειριστεί τους περιορισμούς σε πόρους, τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, αυτές καθ' αυτές τις δραστηριότητες του έργου με εκτίμηση της διάρκειας εκτέλεσης της κάθε μίας αναφορικά με την αντίστοιχη απαιτούμενη ποσότητα πόρων για την εκτέλεσή της, τις χρονικές προθεσμίες και τυχόν εσωτερικές απαραίτητες χρονικές καθυστερήσεις κατά τα στάδια εκτέλεσης των δραστηριοτήτων. Η διαχείριση των προαναφερθέντων χαρακτηριστικών θα πρέπει να γίνει με τέτοιο τρόπο ώστε ο χρονοπρογραμματισμός της εκτέλεσης κάθε δραστηριότητας με την αντίστοιχη κατανομή των απαιτούμενων πόρων να ικανοποιεί κάθε υφιστάμενο περιορισμό, ενώ παράλληλα να οδηγεί στη βελτιστοποίηση του εκάστοτε αντικειμενικού στόχου.

Ο προθάλαμος όπου είναι σε θέση να εκκολαφθεί η αποτελεσματική διαχείριση του έργου και κατά συνέπεια η παραγωγή ενός σωστού χρονοπρογράμματος, έγκειται στη ρεαλιστική απεικόνιση της πραγματικότητας και στην ύπαρξη υπολογιστικών διαδικασιών με

δυνατότητα συνδυαστικής βελτιστοποίησης με σεβασμό σε όλες τις παραμέτρους και τους περιορισμούς που έχουν τεθεί. Υπολογιστικές διαδικασίες, που με την ελάχιστη απαιτούμενη προσπάθεια από τη σκοπιά του διαχειριστή, θα επιφέρουν το επιθυμητό αποτέλεσμα. Το εκάστοτε παραχθέν χρονοπρόγραμμα οφείλει να οδηγεί στο προσδοκώμενο αποτέλεσμα μέσα από τη βέλτιστη χρησιμοποίηση των διαθέσιμων πόρων, ενώ παράλληλα να είναι εύρωστο ώστε πιθανές μικρές αλλαγές στο πλάνο του έργου ή διαταραχές στη διάρκεια εκτέλεσης ή στη διαθεσιμότητα των πόρων, να έχουν περιορισμένη επίδραση στο συνολικό χρονοπρόγραμμα.

Η κύρια ιδέα του υπό μελέτη προβλήματος, του Προγραμματισμού Έργων Προκαθορισμένης Προσπάθειας με Ευέλικτα Προφίλ Πόρων και Γενικευμένες Σχέσεις Προτεραιότητας (*Effort Driven Resource Constrained Project Scheduling with Flexible Resource Profiles and Generilized Precedecne Relations with minimal and maximal Time Lags*), αποτυπώνεται στη δυνατότητα χρονοπρογραμματισμού και υπολογισμού της κατανομής των πόρων, μέσω της παραγωγής ευέλικτων προφίλ εκτέλεσης που βελτιστοποιούν τον αντικειμενικό σκοπό του έργου, με ταυτόχρονο σεβασμό στις απαιτήσεις και στους ρεαλιστικά διατυπωμένους περιορισμούς. Με αυτό τον τρόπο ο διαχειριστής του έργου δεν χρειάζεται να εκτιμά το εκάστοτε προφίλ κατανομής με την απαιτούμενη ποσότητα πόρων και την αντίστοιχη διάρκεια εκτέλεσης κάθε δραστηριότητας. Αντίθετα, εκτιμά την απαιτούμενη προσπάθεια (*effort*) (επί παραδείγματι τις απαιτούμενες εργατοημέρες) κάθε δραστηριότητας και μέσω του παραγόμενου χρονοπρογραμμάτος της υπολογιστικής διαδικασίας λαμβάνει και το βέλτιστο προφίλ εκτέλεσης κάθε δραστηριότητας για κάθε απαιτούμενο πόρο, με την κατανεμημένη ποσότητα ανά χρονική περίοδο και τη συνολική διάρκεια εκτέλεσης της δραστηριότητας.

Για την πληρότητα του μοντέλου, λαμβάνεται υπόψιν και η χρήση ελάχιστων και μέγιστων χρονικών καθυστερήσεων. Συνίσταται η χρήση τους σε περιορισμένο αριθμό δραστηριοτήτων του έργου και όπου η φύση μιας δραστηριότητας απαιτεί την ύπαρξη καθυστερήσεων. Επιπρόσθετα, σημειώνεται ότι οι χρονικές καθυστερήσεις δίδονται σε ποσοστιαίες μονάδες της διάρκειας της δραστηριότητας που αφορούν, καθώς δεν έχει λογική αιτίαση ο ορισμός τους σε χρονικές μονάδες με άγνωστες τις τιμές των διαρκειών των δραστηριοτήτων που αφορούν. Ακολουθεί διακριτή ανάλυση των στόχων, των πόρων, των δεδομένων εισόδου και των εξόδων του υπό μελέτη προβλήματος.

3.2 Αντικειμενικός Στόχος

Το παραχθέν χρονοπρόγραμμα ενός έργου οφείλει να οδηγεί σε ένα αποτέλεσμα που βρίσκεται σε συμφωνία με τα συμφωνηθέντα χαρακτηριστικά ποιότητας, χρόνου και προϋπολογισμού. Οι στόχοι που επιδιώκονται κατά το χρονοπρογραμματισμό ενός έργου ορίζονται μέσω της μετάφρασης των απαιτούμενων χαρακτηριστικών του χρονοπρογράμματος στους πραγματικούς στόχους που επρόκειτω να βελτιστοποιηθούν κατά τον προγραμματισμό του έργου.

Στην πλειονότητα των έργων υπάρχει μια ημερομηνία ολοκλήρωσης που τίθεται είτε από τον πελάτη είτε εσωτερικά από την ομάδα υλοποίησης του έργου. Αβίαστα, η διάρκεια ενός έργου είναι ένα βασικό κριτήριο απόδοσης και αποτελεί έναν συνήθη στόχο που λαμβάνει χώρα στη βιβλιογραφία και σε πρακτικές εφαρμογές (Boctor, 1990, Hartmann, 1998, Kolisch, 1996a). Η ελαχιστοποίηση της διάρκειας του έργου αποτελεί ένα σημαντικό και αποτελεσματικό κριτήριο απόδοσης για τους εξής λόγους (Kolisch, 1996a):

1. Η ποιότητα και η φερεγγυότητα των προβλέψεων εξαρτώνται από το βάθος χρόνου που αφορά η χρονική περιόδος εφαρμογής τους και τείνουν να μειώνονται με την αύξηση του βάθους χρόνου. Η ελαχιστοποίηση της διάρκειας του έργου μειώνει τον χρονικό ορίζοντα και κατά συνέπεια περιορίζει την αβεβαιότητα των δεδομένων των προβλέψεων.
2. Η ελαχιστοποίηση της διάρκειας του έργου, μειώνει την πιθανότητα παραβίασης των προσυμφωνηθέντων προθεσμιών.
3. Γενικά, η χρησιμοποίηση των πόρων τη νωρίτερη δυνατή χρονική περίοδο ελευθερώνει την ποσότητά τους στο εγγύς μέλλον, γεγονός που δίδει τη δυνατότητα ευελιξίας για διαχείριση ενδεχόμενων αλλαγών στο οικονομικό περιβάλλον του οργανισμού.

Στόχος του υπό μελέτη προβλήματος, είναι ο υπολογισμός των ανατιθέμενων προφίλ εκτέλεσης κάθε δραστηριότητας με τους απαιτούμενους, για την εκτέλεσή της, τύπους πόρων, καθώς και ο υπολογισμός των αντίστοιχων χρόνων έναρξης των δραστηριοτήτων του έργου, ώστε να βελτιστοποιείται ο αντικειμενικός στόχος της ελαχιστοποίησης της συνολικής διάρκειας του έργου, με ταυτόχρονη ικανοποίηση όλων των υφιστάμενων περιορισμών.

3.3 Δεδομένα Εισόδου και Περιορισμοί

Ένα χρονοπρόγραμμα αποτελεί ένα πλάνο που καθορίζει τη σειρά εκτέλεσης των δραστηριοτήτων του έργου, τη χρονική στιγμή έναρξης της εκτέλεσης κάθε δραστηριότητας, καθώς και την απαιτούμενη ποσότητα και χρονικό διάστημα χρησιμοποίησης κάθε τύπου πόρων για την εκτέλεση κάθε δραστηριότητας. Ο προσδιορισμός και η αποσαφήνιση των δραστηριοτήτων και των πόρων ενός έργου αποτελούν κομβικής σημασίας για τον ορισμό του υπό μελέτη προβλήματος.

Κάθε δραστηριότητα απαιτεί ένα σύνολο τύπων πόρων για την υλοποίησή της, για κάθε έναν από τους οποίους απαιτείται μια συγκεκριμένη προσπάθεια (*Effort*) για τη συνολική εκτέλεση και ολοκλήρωση της δραστηριότητας. Η διάρκεια κάθε δραστηριότητας είναι άγνωστη εκ των προτέρων, δυναμική και εξαρτώμενη από τη χρησιμοποιούμενη ποσότητα του κάθε τύπου πόρων και τα αντίστοιχα χρονικά διαστήματα της κάθε ανάθεσης, ενώ παράλληλα υπολογίζεται από το παραχθέν προφίλ της. Ως προφίλ ορίζεται το σύνολο των αναθέσεων για κάθε απαιτούμενο πόρο μιας δραστηριότητας, με την ανάθεση να αποτελεί την κατανομή των πόρων στην κάθε διαμέριση της δραστηριότητας σε συνδυασμό με τα αντίστοιχα χρονικά διαστήματα χρησιμοποίησης των πόρων.

Για κάθε δραστηριότητα του έργου, προκύπτει από το υπολογιζόμενο προφίλ ένας συνολικός αριθμός διαμερίσεων, ο οποίος είναι μικρότερος από το δεδομένο μέγιστο επιτρεπόμενο αριθμό διαμερίσεων των δραστηριοτήτων του έργου. Κάθε διαμέριση σηματοδοτεί την αλλαγή της ποσότητας των χρησιμοποιούμενων πόρων από την κάθε δραστηριότητα. Ο συνολικός χρόνος που χρησιμοποιείται ένας πόρος με μία σταθερή ποσότητα από μία δραστηριότητα, και σε κάθε διαμέριση της τελευταίας, πρέπει να είναι μεγαλύτερος από το δεδομένο ελάχιστο χρονικό διάστημα σταθερής ποσότητας χρησιμοποιούμενων πόρων από τη δραστηριότητα αυτή. Το προαναφερθέν χρονικό διάστημα αποτελεί τη χρονική μεταβλητή περιορισμού αναφορικά με τη συχνότητα των αλλαγών στις χρησιμοποιούμενες ποσότητες των πόρων.

Για την έναρξη της εκτέλεσης μιας δραστηριότητας πρέπει όλες οι προαπαιτούμενες δραστηριότητές της να έχουν προγραμματιστεί και ολοκληρωθεί. Οι γενικευμένες σχέσεις

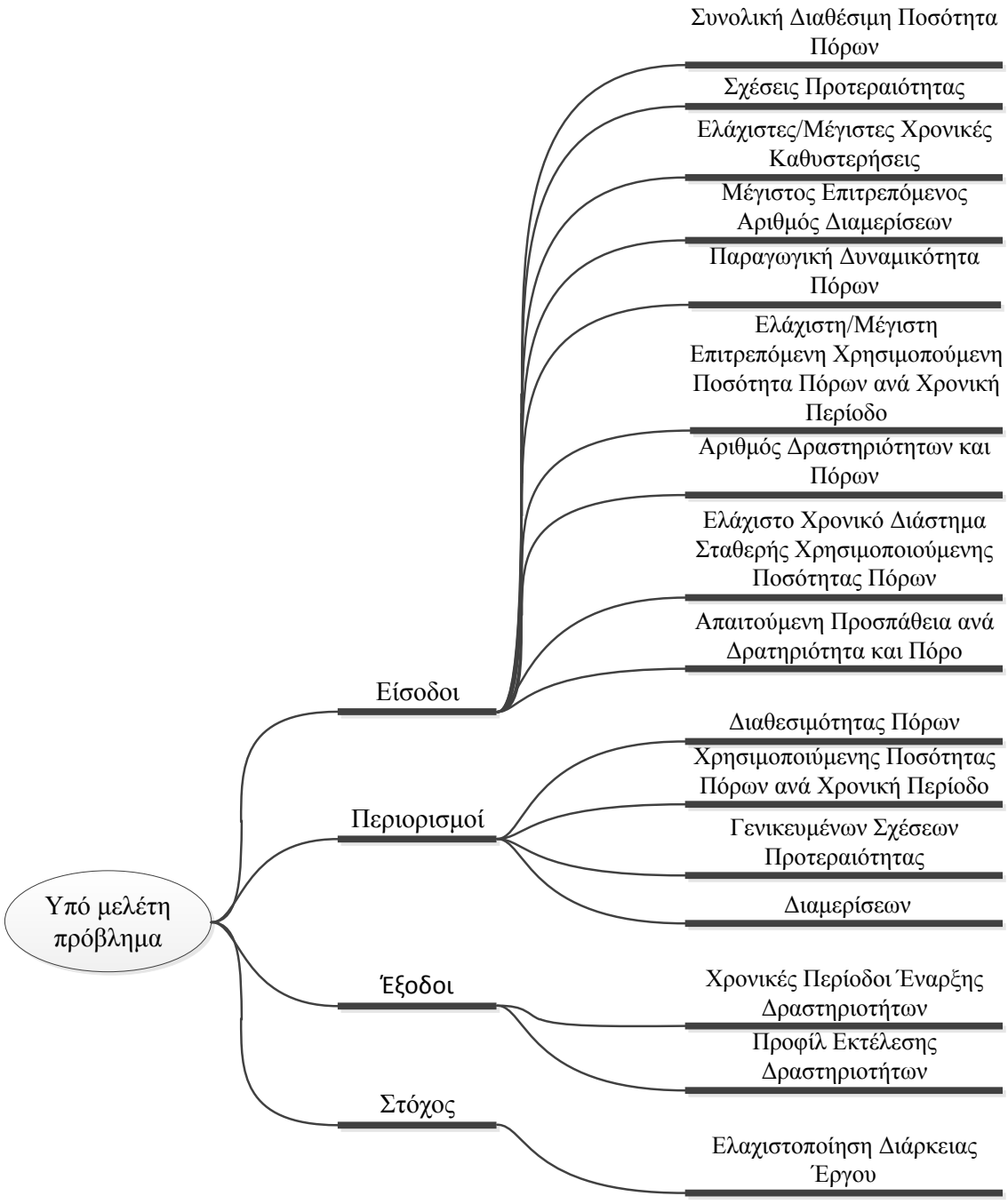
προτεραιότητας που λαμβάνουν χώρα μεταξύ των δραστηριοτήτων του έργου, εκφράζονται με σχέσεις αρχής-αρχής, με τις αντίστοιχες ελάχιστες και μέγιστες χρονικές καθυστερήσεις να είναι εκφρασμένες σε ποσοστιαίες μονάδες της διάρκειας της δραστηριότητας που αφορούν, καθώς δεν έχει λογική αιτίαση ο ορισμός τους σε χρονικές μονάδες με άγνωστες τις τιμές των διαρκειών των δραστηριοτήτων που σχετίζονται. Επιπρόσθετα, δεν επιτρέπεται η προεκχώρηση (*preemption*), γεγονός που υποδηλώνει ότι από τη στιγμή της έναρξης μιας δραστηρίοτητας δεν επιτρέπεται η διακοπή της μέχρι και την ολοκλήρωσή της.

Οι πόροι που καταναλίσκονται είναι ανανεώσιμοι πόροι. Κάθε ανανεώσιμος τύπος πόρων έχει σταθερή διαθέσιμη προς χρησιμοποίηση ποσότητα ανά χρονική περίοδο, ενώ παράλληλα η ανά περίοδος χρησιμοπούμενη ποσότητά του από οποιαδήποτε δραστηριότητα πρέπει να βρίσκεται εντός του διαστήματος που ορίζουν η ελάχιστη και η μέγιστη επιτρεπόμενη χρησιμοποιούμενη ποσότητά του ανά χρονική περίοδο. Επιπλέον, κάθε τύπος πόρων αντιστοιχίζεται με μια παραγωγική δυναμικότητα που υποδηλώνει την ποσότητα εργασίας που δύναται να πραγματοποιήσει στη μονάδα του χρόνου. Στο σημείο αυτό, σημειώνεται ότι ο διαχειριστής του έργου πρέπει πρακτικά να ορίσει την παραγωγική δυναμικότητα του κάθε πόρου ανά δραστηριότητα του έργου ή ανά ομάδες ομοειδών δραστηριοτήτων, καθώς η δυναμικότητα ενός πόρου και η φύση μιας δραστηριότητας αποτελούν αλληλεξαρτώμενες και ελληλεπιδρώμενες έννοιες. Πάραυτα, γίνεται η παραδοχή του ορισμού της παραγωγικής δυναμικότητα κάθε πόρου για όλες τις δραστηριότητες καθολικά.

Τα δεδομένα εισόδου του προβλήματος (όπως οι απαιτούμενοι πόροι των δραστηριοτήτων, η απαιτούμενη προσπάθειά τους, οι διαθεσιμότητες και οι παραγωγικές δυναμικότητες των πόρων, οι ελάχιστες και οι μέγιστες επιτρεπόμενες χρησιμοποιούμενες ποσότητες ανά χρονική περίοδο των πόρων, ο μέγιστος επιτρεπόμενος αριθμός διαμερίσεων, το ελάχιστο χρονικό διάστημα σταθερής ποσότητας των χρησιμοποιούμενων πόρων της κάθε δραστηριότητας, οι συσχετίσεις και οι χρονικές καθυστερήσεις μεταξύ των δραστηριοτήτων) μολονότι στην πραγματικότητα ενδέχεται να μην είναι ντετερμινιστικά δεδομένα, στο υπό μελέτη πρόβλημα θεωρούνται ντετερμινιστικά.

3.4 Έξοδοι

Στόχος του υπό μελέτη προβλήματος είναι ο υπολογισμός των ανατιθέμενων προφίλ - χρησιμοποιούμενη ποσότητα κάθε απαιτούμενου πόρου ανά χρονική περίοδο- όλων των δραστηριοτήτων του έργου και των αντίστοιχων χρόνων έναρξης, ώστε να βελτιστοποιηθεί ο αντικειμενικός στόχος της ελαχιστοποίησης της συνολικής διάρκειας του έργου με ικανοποίηση όλων των υφιστάμενων περιορισμών. Κατά το πέρας της επίλυσης του προβλήματος λαμβάνονται ως έξοδοι τόσο οι χρόνοι έναρξης όσο και τα προφίλ εκτέλεσης και οι διάρκειες όλων των δραστηριοτήτων του έργου. Το παραγόμενο χρονοπρόγραμμα παρουσιάζεται τελικά στο διαχειριστή έργου και μπορεί να χρησιμοποιηθεί ως εργαλείο παρακολούθησης και ελέγχου της εκτέλεσης του έργου.



Σχήμα 13. Δομή υπό μελέτη προβλήματος

4

Μαθηματική Μοντελοποίηση

4.1 Εννοιολογική Διατύπωση Προβλήματος

4.1.1 Ορισμοί

Το υπό μελέτη πρόβλημα, όπως ορίστηκε στο Κεφάλαιο 3, δύναται να διατυπωθεί εννοιολογικά, ως ακολούθως:

- Όλα τα δεδομένα θεωρούνται ντετερμινιστικά και γνωστά εκ των προτέρων.
- Το υπό προγραμματισμό έργο, αποτελείται από n δραστηριότητες, $i = 1, \dots, n$, και από δύο βοηθητικές δραστηριότητες (*dummy activities*), τη βοηθητική δραστηριότητα αρχής $i = 0$ (*dummy source activity*) και τη βοηθητική δραστηριότητα τέλους $i = n + 1$ (*dummy sink activity*), οι απαιτήσεις των οποίων σε πόρους, καθώς και οι διάρκειές τους λαμβάνουν μηδενική τιμή, ενώ αναπαριστούν την έναρξη και τη λήξη του έργου αντίστοιχα. Ορίζεται ως $V = \{0,1, \dots, n, n+1\}$, το σύνολο όλων των δραστηριοτήτων του έργου.
- Ορίζεται ως T, ο χρονικός ορίζοντας του έργου, ο οποίος προκύπτει ως το άθροισμα των μέγιστων δυνατών διαρκειών των δραστηριοτήτων του έργου. Η διάρκεια κάθε δραστηριότητας, για τον υπολογισμό του χρονικού ορίζοντα ως το άνω όριο του χρονοπρογραμματισμού, προκύπτει από την αντίστοιχη προσπάθεια αν θεωρηθεί μοναδιαία η χρήση πόρων.
- Οι πόροι που καταναλίσκονται είναι ανανεώσιμοι πόροι. Το σύνολο των ανανεώσιμων πόρων συμβολίζεται ως R^{ρ}. Για κάθε τύπο ανανεώσιμου πόρου $k \in R^{\rho}$, η συνολική διαθεσιμότητά του ανα χρονική περίοδο είναι σταθερή και συμβολίζεται ως α_k^{ρ}.
- Κάθε δραστηριότητα i απαιτεί ένα σύνολο K_i τύπους πόρων για την υλοποίησή της. Για κάθε τύπο ανανεώσιμων πόρων $k \in K_i$, ορίζεται ως E_{ki}, η δεδομένη απαιτούμενη προσπάθεια (*Effort*) που απαιτείται από τον τύπο πόρου k για τη συνολική εκτέλεση και ολοκλήρωση της δραστηριότητας i, με $E_{ik} = \sum_{q=0}^{q_{total}} din_k \cdot r_{ik,q} \cdot t_{alloc_{ik,q}}$, όπου:
  - q, η διαμέριση της δραστηριότητας i κατά τη χρήση του τύπου πόρων k, και q_{total} ο συνολικός προκύπτων αριθμός διαμερίσεων της δραστηριότητας i κατά τη χρήση του τύπου πόρων k. Κάθε διαμέριση $q \equiv q_{ik}$, σηματοδοτεί την αλλαγή της ποσότητας των χρησιμοποιούμενων πόρων τύπου k για τη δραστηριότητα i, με

q_{max} να δηλώνει το δεδομένο μέγιστο επιτρεπόμενο αριθμό διαμερίσεων των δραστηριοτήτων του έργου.

- $r_{ik,q}$, το πλήθος των χρησιμοποιούμενων πόρων τύπου k από τη δραστηριότητα i στη διαμέριση q_{ik}.
- din_k, η παραγωγική δυναμικότητα του τύπου πόρου k στη μονάδα του χρόνου.
- $t_{alloc_{ik,q}}$, ο συνολικός χρόνος που χρησιμοποιείται ο πόρος k από τη δραστηριότητα i στη διαμέριση q_{ik}.

- Για κάθε τύπου πόρων k, ορίζεται η ελάχιστη και η μέγιστη επιτρεπόμενη χρησιμοποιούμενη ποσότητά του ανά χρονική περίοδο, ως r_k^{min} και r_k^{max} αντίστοιχα.
- Ορίζεται ως $t_{alloc_i}^{min}$, το ελάχιστο χρονικό διάστημα σταθερής ποσότητας των χρησιμοποιούμενων πόρων της δραστηριότητας i, μεταξύ των διαμερίσεών της. Έγκειται στη χρονική μεταβητή περιορισμού αναφορικά με τη συχνότητα των αλλαγών στις χρησιμοποιούμενες ποσότητες των πόρων.
- Η διάρκεια της δραστηριότητας i είναι δυναμική και εξαρτώμενη από τη χρησιμοποιούμενη ποσότητα του κάθε τύπου πόρων και τα αντίστοιχα χρονικά διαστήματα της κάθε ανάθεσης πόρων. Υπολογίζεται ως:

  $d_{i,P} = max_{k \in K_i}\{\sum_{q=0}^{q_{total}} t_{alloc_{ik,q}}\}$.
- Ορίζεται ως ανάθεση p_{ik}, η κατανομή των πόρων στην κάθε διαμέριση q_{ik} της δραστηριότητας i και τα αντίστοιχα χρονικά διαστήματα χρησιμοποίησης, με τέτοιο τρόπο ώστε: $p_{ik} = \begin{bmatrix} r_{ik,0} & t_{alloc_{ik,0}} \\ \vdots & \vdots \\ r_{ik,q_{total}} & t_{alloc_{ik,q_{total}}} \end{bmatrix}$. Οι συνολικές αναθέσεις $p_{ik}, \forall k \in K_i$ για όλους τους απαιτούμενους πόρους για την εκτέλεση της δραστηριότητας i, συνθέτουν το συνολικό προφίλ εκτέλεσης $P_i \equiv P$ της κάθε δραστηριότητας i.
- Ορίζεται ως $s_{i,P}$ η μοναδική χρονική στιγμή έναρξης της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i.
- Ορίζεται ως $f_{i,P}$ η μοναδική χρονική στιγμή ολοκλήρωσης της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i.
- Οι βοηθητικές δραστηριότητες έναρξης και λήξης λαμβάνουν μηδενική τιμή διάρκειας και απαίτησης σε πόρους. Για τη βοηθητική δραστηριότητα έναρξης ορίζονται οι χρόνοι έναρξης και λήξης ως $s_{0,0} = f_{0,0} = 0$, ενώ για τη βοηθητική δραστηριότητα λήξης ως $s_{n+1,0} = f_{n+1,0} = max_{i \in V}\{f_{i,P}\}$.
- Δεν επιτρέπεται η προεκχώρηση (*preemption*), γεγονός που υποδηλώνει ότι από τη στιγμή της έναρξης μιας δραστηρίοτητας i δεν επιτρέπεται η διακοπή της μέχρι και την ολοκλήρωσή της ($f_{i,P} = s_{i,P} + d_{i,P}$).
- Οι δεδομένες σχέσεις προτεραιότητας που αναπτύσσονται μεταξύ των δραστηριοτήτων έγκεινται σε τέσσερις διαφορετικούς τύπους: (1) σχέση αρχής-αρχής $SS_{iP,jP}$, (2) σχέση τέλους-τέλους $FF_{iP,jP}$, (3) σχέση τέλους-αρχής $FS_{iP,jP}$ και σχέση αρχής-τέλους $SF_{iP,jP}$ με ελάχιστα και μέγιστα χρονικά περιθώρια καθυστέρησης μεταξύ των δραστηριοτήτων i υπό το προφίλ εκτέλεσης P_i και j υπό το προφίλ εκτέλεσης P_j. Οι χρονικές καθυστερήσεις δίδονται ως ποσοστιαίες μονάδες των -

άγνωστων εκ των προτέρων- διαρκειών, των προαπαιτούμενων σε κάθε σχέση δραστηριοτήτων. Πιο συγκεκριμένα:

- $SS_{iP,jP}^{min}$ ορίζει ότι η δραστηριότητα j υπό το προφίλ εκτέλεσης P_j δεν δύναται να ξεκινήσει νωρίτερα από $SS_{iP,jP}^{min}$ χρονικές μονάδες ύστερα από την έναρξη της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i, ενώ παράλληλα είναι εκφρασμένο ως ένα ποσοστό α της διάρκειας της προαπαιτούμενης δρατηριότητας i, $SS_{iP,jP}^{min} = \alpha\% \cdot d_{i,P}$.
- $SS_{iP,jP}^{max}$ ορίζει ότι η δραστηριότητα j υπό το προφίλ εκτέλεσης P_j δεν δύναται να ξεκινήσει αργότερα από $SS_{iP,jP}^{min}$ χρονικές μονάδες ύστερα από την έναρξη της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i, ενώ παράλληλα είναι εκφρασμένο ως ένα ποσοστό α της διάρκειας της προαπαιτούμενης δρατηριότητας i, $SS_{iP,jP}^{max} = \alpha\% \cdot d_{i,P}$.
- $SF_{iP,jP}^{min}$ ορίζει ότι η δραστηριότητα j υπό το προφίλ εκτέλεσης P_j δεν δύναται να ολοκληρωθεί νωρίτερα από $SF_{iP,jP}^{min}$ χρονικές μονάδες ύστερα από την έναρξη της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i, ενώ παράλληλα είναι εκφρασμένο ως ένα ποσοστό α της διάρκειας της προαπαιτούμενης δρατηριότητας i, $SF_{iP,jP}^{min} = \alpha\% \cdot d_{i,P}$.
- $SF_{iP,jP}^{max}$ ορίζει ότι η δραστηριότητα j υπό το προφίλ εκτέλεσης P_j δεν δύναται να ολοκληρωθεί αργότερα από $SF_{iP,jP}^{max}$ χρονικές μονάδες ύστερα από την έναρξη της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i, ενώ παράλληλα είναι εκφρασμένο ως ένα ποσοστό α της διάρκειας της προαπαιτούμενης δρατηριότητας i, $SF_{iP,jP}^{max} = \alpha\% \cdot d_{i,P}$.
- $FS_{iP,jP}^{min}$ ορίζει ότι η δραστηριότητα j υπό το προφίλ εκτέλεσης P_j δεν δύναται να ξεκινήσει νωρίτερα από $FS_{iP,jP}^{min}$ χρονικές μονάδες ύστερα από την ολοκλήρωση της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i, ενώ παράλληλα είναι εκφρασμένο ως ένα ποσοστό α της διάρκειας της προαπαιτούμενης δρατηριότητας i, $FS_{iP,jP}^{min} = \alpha\% \cdot d_{i,P}$.
- $FS_{iP,jP}^{max}$ ορίζει ότι η δραστηριότητα j υπό το προφίλ εκτέλεσης P_j δεν δύναται να ξεκινήσει αργότερα από $FS_{iP,jP}^{max}$ χρονικές μονάδες ύστερα από την ολοκλήρωση της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i, ενώ παράλληλα είναι εκφρασμένο ως ένα ποσοστό α της διάρκειας της προαπαιτούμενης δρατηριότητας i, $FS_{iP,jP}^{max} = \alpha\% \cdot d_{i,P}$.
- $FF_{iP,jP}^{min}$ ορίζει ότι η δραστηριότητα j υπό το προφίλ εκτέλεσης P_j δεν δύναται να ολοκληρωθεί νωρίτερα από $FF_{iP,jP}^{min}$ χρονικές μονάδες ύστερα από την ολοκλήρωση της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i, ενώ παράλληλα είναι εκφρασμένο ως ένα ποσοστό α της διάρκειας της προαπαιτούμενης δρατηριότητας i, $FF_{iP,jP}^{min} = \alpha\% \cdot d_{i,P}$.
- $FF_{iP,jP}^{max}$ ορίζει ότι η δραστηριότητα j υπό το προφίλ εκτέλεσης P_j δεν δύναται να ολοκληρωθεί αργότερα από $FF_{iP,jP}^{max}$ χρονικές μονάδες ύστερα από την ολοκλήρωση της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i, ενώ παράλληλα είναι

εκφρασμένο ως ένα ποσοστό a της διάρκειας της προαπαιτούμενης δρατηριότητας i, $FF_{iP,jP}^{max} = \alpha\% \cdot d_{i,P}$.

- Όλες οι σχέσεις προτεραιότητας μετασχηματίζονται ώστε να αναπαρίστανται μέσω ενός τύπου, σχέσεων αρχής-αρχής SS, δεδομένου ότι έχει λάβει χώρα ο υπολογισμός και η σταθεροποίηση των διαρκειών των δραστηριοτήτων, σύμφωνα με τους ακόλουθους κανόνες μετασχηματισμού:
  - Χρονικές καθυστερήσεις αρχής-αρχής:

    $$s_{i,P} + SS_{iP,jP}^{min} \leq s_{j,P} \rightarrow s_{i,P} + \delta_{iP,jP} \leq s_{j,P},\ \delta_{iP,jP} = SS_{iP,jP}^{min}$$

    $$s_{i,P} + SS_{iP,jP}^{max} \geq s_{j,P} \rightarrow s_{j,P} + \delta_{jP,iP} \leq s_{i,P},\ \delta_{jP,iP} = -SS_{iP,jP}^{max}$$

  - Χρονικές καθυστερήσεις αρχής-τέλους:

    $$s_{i,P} + SF_{iP,jP}^{min} \leq f_{j,P} \rightarrow s_{i,P} + \delta_{iP,jP} \leq s_{j,P},\ \delta_{iP,jP} = SF_{iP,jP}^{min} - d_{j,P}$$

    $$s_{i,P} + SF_{iP,jP}^{max} \geq f_{j,P} \rightarrow s_{j,P} + \delta_{jP,iP} \leq s_{i,P},\ \delta_{jP,iP} = d_{j,P} - SF_{iP,jP}^{max}$$

  - Χρονικές καθυστερήσεις τέλους-αρχής: [5.1]

    $$f_{i,P} + FS_{iP,jP}^{min} \leq s_{j,P} \rightarrow s_{i,P} + \delta_{iP,jP} \leq s_{j,P},\ \delta_{iP,jP} = d_{i,P} + FS_{iP,jP}^{min}$$

    $$f_{i,P} + FS_{iP,jP}^{max} \geq s_{j,P} \rightarrow s_{j,P} + \delta_{jP,iP} \leq s_{i,P}\ ,\ \delta_{jP,iP} = -d_{i,P} - FS_{iP,jP}^{max}$$

  - Χρονικές καθυστερήσεις τέλους-τέλους:

    $$f_{i,P} + FF_{iP,jP}^{min} \leq f_{j,P} \rightarrow s_{i,P} + \delta_{iP,jP} \leq s_{j,P},\ \delta_{iP,jP} = d_{i,P} - d_{j,P} + FF_{iP,jP}^{min}$$

    $$f_{i,P} + FF_{iP,jP}^{max} \geq f_{j,P} \rightarrow s_{j,P} + \delta_{jP,iP} \leq s_{i,P},\ \delta_{jP,iP} = d_{j,P} - d_{i,P} - FF_{iP,jP}^{max}$$

- Σε περίπτωση που το έργο αναπαρίσταται ως δίκτυο δραστηριοτήτας-σε-κόμβο (*AoN*) ορίζεται ο δίγραφος $G = (V, A)$, όπου το διάνυσμα $V = \{0,1, \dots, n, n+1\}$ περιλαμβάνει όλες της δραστηριότητες του έργου και το $A = \{(i,j) \mid i,j \in V; i \rightarrow j\}$ αναπαριστά τις γενικευμένες σχέσεις προτεραιότητας.
- Το διάνυσμα $S = (s_{i,P})_{\forall i \in V}$ αποτελεί ένα παραχθέν χρονοπρόγραμμα, το οποίο θεωρείτω εφικτό, αν ικανοποιούνται οι περιορισμοί των γενικευμένων σχέσεων προτεραιότητας και της διαθεσιμότητας κάθε ανανεώσιμού πόρου.
- Ως $Act(t)$ ορίζεται το σύνολο που περιλαμβάνει όλες τις δραστηριότητες που βρίσκονται υπό εξέλιξη τη χρονική περίοδο $t,\ t = 0,1, \dots, T-1$.

Στόχος είναι ο υπολογισμός των ανατιθέμενων προφίλ P_i και των αντίστοιχων χρόνων έναρξης $s_{i,P}$ των δραστηριοτήτων $i = 1, \dots, n$, ώστε να βελτιστοποιηθούν οι αντικειμενικοί στόχοι με ικανοποίηση όλων των υφιστάμενων περιορισμών.

Στον ακόλουθο πίνακα συνοψίζεται η σημειογραφία που παρουσιάστηκε στην παράγραφο αυτή.

Πίνακας 8. Σημειογραφία υπό μελέτη προβλήματος

Σύμβολο	**Ορισμός**
n	Ο αριθμός των πραγματικών δραστηριοτήτων
$V = \{0, \dots, n+1\}$	Το σύνολο όλων των δραστηριοτήτων i (συμπεριλαβανομένων και των βοηθητικών δραστηριοτήτων 0 και n+1)
$G = (V, A)$	Δίγραφος των σχέσεων προτεραιότητας των δραστηριοτήτων i
$A = \{(i,j) \mid i,j \in V; i \rightarrow j\}$	Το σύνολο των γενικευμένων σχέσεων προτεραιότητας

T	Ο χρονικός ορίζοντας του έργου, ως το άθροισμα των μέγιστων διαρκειών όλων των δραστηριοτήτων
$t = 0, \dots, T$	Χρονική περίοδος
[t,t+1)	Χρονικό διάστημα που αντιστοιχεί στη χρονική περίοδο t
Act(t)	Το σύνολο όλων των υπό εξέλιξη δραστηριοτήτων τη χρονική περίοδο t
R^{ρ}	Το σύνολο των ανανεώσιμων πόρων
$k \in R^{\rho}$	Ανανεώσιμος τύπος πόρων
α_k^{ρ}	Η διαθεσιμότητα του τύπου πόρου k ανά χρονική περίοδο
K_i	Το σύνολο των απαιτούμενων ανανεώσιμων πόρων από τη δραστηριότητα i
$E_{ik} = \sum_{q=0}^{q_{total}} din_k \cdot r_{ik,q} \cdot t_{alloc_{ik,q}}$	Η απαιτούμενη προσπάθεια (*Effort*) που απαιτείται από τον τύπο πόρου k για τη συνολική εκτέλεση της δραστηριότητας i
$q \equiv q_{ik}$	Η διαμέριση της δραστηριότητας i κατά τη χρήση του τύπου πόρων k
$q_{total} \equiv q_{i,total}$	Ο συνολικός αριθμός διαμερίσεων της δραστηριότητας i κατά τη χρήση του τύπου πόρων k
q_{max}	Ο μέγιστος επιτρεπόμενος αριθμός διαμερίσεων των δραστηριοτήτων
$r_{ik,q}$	Το πλήθος των χρησιμοποιούμενων πόρων τύπου k από τη δραστηριότητα i στη διαμέριση q_{ik}
din_k	Η παραγωγική δυναμικότητα του τύπου πόρου k στη μονάδα του χρόνου
$t_{alloc_{ik,q}}$	Ο συνολικός χρόνος που χρησιμοποιείται ο πόρος k από τη δραστηριότητα i στη διαμέριση q_{ik}
r_k^{min}/r_k^{max}	Η ελάχιστη/μέγιστη επιτρεπόμενη χρησιμοποιούμενη ποσότητα κάθε τύπου πόρων k ανά χρονική περίοδο
$t_{alloc_i}^{min}$	Το ελάχιστο χρονικό διάστημα σταθερής ποσότητας των χρησιμοποιούμενων πόρων της δραστηριότητας i
$p_{ik} = \begin{bmatrix} r_{ik,0} & t_{alloc_{ik,0}} \\ \vdots & \vdots \\ r_{ik,q_{total}} & t_{alloc_{ik,q_{total}}} \end{bmatrix}$	Η ανάθεση του τύπου πόρου k για τη δραστηριότητα i
$P_i \equiv P$	Το συνολικό προφίλ ανάθεσης εκτέλεσης της δραστηριότητας i, για τις συνολικές αναθέσεις $p_{ik}, \forall k \in K_i$
$d_{i,P} = max_{k \in K_i}\{\sum_{q=0}^{q_{total}} t_{alloc_{ik,q}}\}$	Η διάρκεια της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i
$s_{i,P}$	Η μοναδική χρονική στιγμή έναρξης της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i
$f_{i,P}$	Η μοναδική χρονική στιγμή λήξης της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i
$s_{0,0} (= f_{0,0} = 0)$	Ο χρόνος έναρξης του έργου (≡ χρόνος λήξης και έναρξης βοηθητικής δραστηριότητας αρχής $i = 0$)
$f_{n+1,0} = max_{i \in V}\{f_{i,P}\} (= s_{n+1,0})$	Ο χρόνος λήξης του έργου (≡ χρόνος λήξης και έναρξης βοηθητικής δραστηριότητας τέλους $i = \nu + 1$)
$S = (s_{i,P})_{\forall i \in V}$	Χρονοπρόγραμμα, διάνυσμα των χρόνων έναρξης κάθε δραστηριότητας i
$SS_{iP,jP}^{min}/SS_{iP,jP}^{max}$	Ελάχιστο/Μέγιστο χρονικό περιθώριο καθυστέρησης μεταξύ των χρόνων έναρξης των δραστηριοτήτων i υπό το προφίλ εκτέλεσης P_i και j υπό το προφίλ εκτέλεσης P_j
$SF_{iP,jP}^{min}/SF_{iP,jP}^{max}$	Ελάχιστο/Μέγιστο χρονικό περιθώριο καθυστέρησης μεταξύ του χρόνου έναρξης της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i και του χρόνου λήξης της j υπό το προφίλ εκτέλεσης P_j
$FS_{iP,jP}^{min}/FS_{iP,jP}^{max}$	Ελάχιστο/Μέγιστο χρονικό περιθώριο καθυστέρησης μεταξύ του χρόνου λήξης της δραστηριότητας i υπό το προφίλ εκτέλεσης P_i και του χρόνου έναρξης της j υπό το προφίλ εκτέλεσης P_j
$FF_{iP,jP}^{min}/FF_{iP,jP}^{max}$	Ελάχιστο/Μέγιστο χρονικό περιθώριο καθυστέρησης μεταξύ των χρόνων λήξης των δραστηριοτήτων i υπό το προφίλ εκτέλεσης P_i

	και j υπό το προφίλ εκτέλεσης P_j
$\delta_{iP,jP}$	Χρονικό περιθώριο καθυστέρησης μεταξύ των χρόνων έναρξης των δραστηριοτήτων i υπό το προφίλ εκτέλεσης P_i και j υπό το προφίλ εκτέλεσης P_j

4.1.2 Αντικειμενικός Στόχος

Κατά την επίλυση του προβλήματος του Προγραμματισμού Έργων Προκαθορισμένης Προσπάθειας με Ευέλικτα Προφίλ Πόρων και Γενικευμένες Σχέσεις Προτεραιότητας, επιδιώκεται η παραγωγή ενός βέλτιστου χρονοπρογράμματος αναφοράς με απώτερο σκοπό την ελαχιστοποίηση της συνολικής διάρκειας του έργου. Η ελαχιστοποίηση της συνολικής διάρκειας του έργου ταυτίζεται με την ελαχιστοποίηση του χρόνου λήξης (ή έναρξης) της βοηθητικής δραστηριότητας τέλους και δύναται να εκφραστεί ως εξής:

$$\min f_{n+1,0} \quad [5.2]$$

4.1.3 Περιορισμοί

Η χρονική στιγμή έναρξης κάθε πραγματικής δραστηριότητας πρέπει να είναι μεγαλύτερη ή ίση από την αντίστοιχη της βοηθητικής δραστηριότητας αρχής, γεγονός που υποδηλώνει ότι η βοηθητική δραστηριότητα αρχής αποτελεί πάντα την πρώτη προγραμματιζόμενη δραστηριότητα τη χρονική στιγμή $t = 0$:

$$s_{i,P} \geq s_{0,0}, \forall i = 1, \dots, n+1 \quad [5.3]$$

Η χρονική στιγμή λήξης κάθε πραγματικής δραστηριότητας πρέπει να είναι μικρότερη ή ίση από την αντίστοιχη της βοηθητικής δραστηριότητας τέλους, γεγονός που υποδηλώνει ότι η βοηθητική δραστηριότητα τέλους αποτελεί πάντα την τελευταία προγραμματιζόμενη δραστηριότητα τη χρονική στιγμή:

$$f_{i,P} \leq f_{n+1,0}, \forall i = 1, \dots, n \quad [5.4]$$

Η συνολική χρησιμοποιούμενη ποσότητα κάθε τύπου ανανεώσιμων πόρων k, από το σύνολο των υπό εξέλιξη δραστηριοτήτων $Act(t)$, κάθε χρονική στιγμή t, πρέπει να είναι μικρότερη ή ίση από την αντίστοιχη συνολική διαθέσιμη ποσότητά του, α_k^ρ. Ο περιορισμός διαθεσιμότητας των πόρων, εκφράζεται ως εξής:

$$\sum_{Act(t)} r_{ik,q} \leq \alpha_k^\rho, \forall k \in R^\rho, \forall t = 0,1, \dots, T-1 \quad [5.5]$$

Η χρησιμοποιούμενη ποσότητα κάθε τύπου πόρων k από μία δραστηριότητα i ανά διαμέριση q_{ik}, πρέπει να βρίσκεται εντός του διαστήματος που ορίζουν η ελάχιστη και η μέγιστη επιτρεπόμενη χρησιμοποιούμενη ποσότητα του τύπου πόρων k ανά χρονική περίοδο για τη δραστηριότητα i, και εκφράζεται ως εξής:

$$r_k^{min} \leq r_{ik,q} \leq r_k^{max}, \quad \forall i \in V, \forall k \in K_i, \forall q_{ik} = 0, \dots, q_{ik,total} \quad [5.6]$$

Η σειρά εκτέλεσης των δραστηριοτήτων του έργου καθορίζεται από τις γενικευμένες σχέσεις προτεραιότητας και οφείλει να βρίσκεται σε συμφωνία με αυτές. Οι γενικευμένες σχέσεις προτεραιότητας μετασχηματίζονται ώστε να αναπαρίστανται όλες με κοινό τύπο σχέσεων προτεραιότητας, αρχής-αρχής, σύμφωνα με τους κανόνες μετασχηματισμού [5.1]. Ο γενικός περιορισμός των σχέσεων προτεραιότητας αρχής-αρχής μεταξύ των δραστηριοτήτων του έργου, εκφράζεται ως εξής:

$$\delta_{iP,jP} \leq s_{j,P} - s_{i,P}, \forall (i,j) \in A \quad [5.7]$$

Σε κάθε διαμέριση q_{ik} της διάρκειας εκτέλεσης μιας δραστηριότητας i, ο συνολικός χρόνος που γίνεται χρήση του ανανεώσιμου τύπου πόρων k με σταθερή ποσότητα καθ' όλη τη διαμέριση, $t_{alloc_{ik,q}}$, πρέπει να είναι μεγαλύτερος ή ίσος από το ελάχιστο χρονικό διάστημα σταθερής ποσότητας των χρησιμοποιούμενων πόρων της δραστηριότητας i, $t_{alloc_i}^{min}$, και εκφράζεται ως εξής:

$$t_{alloc_{ik,q}} \geq t_{alloc_i}^{min}, \forall i \in V, \forall k \in K_i, \forall q_{ik} = 0, \dots, q_{ik,total} \quad [5.8]$$

Ο συνολικός αριθμός διαμερίσεων, $q_{ik,total}$, που λαμβάνει χώρα σε κάθε δραστηριότητα i κατά τη χρήση του κάθε απαιτούμενου πόρου k, πρέπει να είναι μικρότερος ή ίσος από το μέγιστο επιτρεπόμενο αριθμό διαμερίσεων των δραστηριοτήτων του έργου q_{max}, και εκφράζεται ως εξής:

$$q_{ik,total} \leq q_{max}, \forall i \in V, \forall k \in K_i \quad [5.9]$$

Τα επιμέρους γινόμενα της χρησιμοποιούμενης ποσότητας κάθε τύπου πόρου k από τη δραστηριότητα i επί του χρονικού διαστήματος χρησιμοποίησης επί της παραγωγικής δυναμικότητας του πόρου ανά διαμέριση q_{ik}, πρέπει να αθροίζουν στην απαιτούμενη προσπάθεια, E_{ki}, που απαιτείται από τον τύπο πόρου k για τη συνολική εκτέλεση και ολοκλήρωση της δραστηριότητας i, και εκφράζεται ως εξής:

$$E_{ik} = \sum_{q=0}^{q_{total}} din_k \cdot r_{ik,q} \cdot t_{alloc_{ik,q}} \quad [5.10]$$

Τέλος, όλες οι μεταβλητές θεωρούνται μη αρνητικές και ακέραιας τιμής.

4.2 Δυαδική Μαθηματική Μοντελοποίηση

Το δυαδικό μαθηματικό μοντέλου του υπό μελέτη, ως άνωθι διατυπωμένου, προβλήματος, υλοποιείται κάνοντας χρήση της δυαδικής μεταβλητής x_{iktq}, η οποία ορίζεται ως εξής:

$$x_{iktq} = \begin{cases} 1, & \text{αν η δραστηριότητα } i, \text{υπό τη χρήση του πόρου } k \text{ και τη διαμέριση} \\ & q_{ik}, \text{ξεκινάει την εκτέλεσή της τη χρονική στιγμή } t \\ 0, & \text{σε κάθε άλλη περίπτωση} \end{cases}$$

Εν συνεχεία παρουσιάζεται η μαθηματική μοντελοποίηση:

$$\min f_{n+1,0} \quad [5.11]$$

υπό τους περιορισμούς:

$$\sum_{t=0}^{T-1} x_{iktq} = 1, \qquad \forall i \in V, \forall k \in K_i, \forall q = 0, \dots, q_{ik,total} \quad [5.12]$$

$$\sum_{t=0}^{T-1} t \cdot x_{ikt,q-1} + t_{alloc_{ik,q-1}} = \sum_{t=0}^{T-1} t \cdot x_{iktq}, \forall i \in V, \forall k \in K_i, \forall q = 0, \dots, q_{ik,total} \quad [5.13]$$

$$\delta_{i\kappa,j\lambda} \leq \sum_{t=0}^{T-1} t \cdot x_{jkt0} - \sum_{t=0}^{T-1} t \cdot x_{ikt0}, \qquad \forall (i,j) \in A, \kappa \equiv P_i, \lambda \equiv P_j \quad [5.14]$$

$$\sum_{\forall i \in Act(t)} \sum_{q_{ik}=0}^{q_{ik,total}} r_{ik,q} \cdot x_{iktq} \leq a_k^{\rho}, \qquad \forall k \in K_i, \forall t = 0, \dots, T-1 \qquad [5.15]$$

$$r_k^{min} \leq \sum_{\forall i \in Act(t)} \sum_{q_{ik}=0}^{q_{ik,total}} r_{ik,q} \cdot x_{iktq} \leq r_k^{max}, \qquad \forall k \in K_i, \forall t = 0, \dots, T-1 \qquad [5.16]$$

$$\sum_{t=0}^{T-1} t \cdot x_{0kt0} \leq \sum_{t=0}^{T-1} t \cdot x_{iktq} \leq \sum_{t=0}^{T-1} t \cdot x_{n+1,kt0}, \forall i \in V, \forall k \in K_i, \forall q = 0, \dots, q_{ik,total} \qquad [5.17]$$

$$x_{iktq} \in \{0,1\}, \qquad \forall i \in V, \forall k \in K_i, \forall q = 0, \dots, q_{ik,total} \qquad [5.18]$$

Η αντικειμενική συνάρτηση [5.11] ελαχιστοποιεί τη χρονική στιγμή λήξης της βοηθητικής δραστηριότητας τέλους, η οποία ταυτίζεται με τη συνολική διάρκεια ολοκλήρωσης του έργου. Ο περιορισμός [5.12] εξασφαλίζει τη μοναδικότητα της χρονικής στιγμής έναρξης κάθε δραστηριότητας υπό τη χρήση κάθε απαιτούμενου πόρου, ενώ ο περιορισμός [5.13] εξασφαλίζει τη χρονική συνέχεια των διαστημάτων των διαμερίσεων χρησιμοποίησης του κάθε πόρου. Ο συνδυασμός των περιορισμών [5.12] και [5.13] καθορίζει τη μη δυνατότητα προεκχώρησης, γεγονός που υποδηλώνει ότι από τη στιγμή της έναρξης μιας δραστηρίοτητας δεν επιτρέπεται η διακοπή της μέχρι και την ολοκλήρωσή της. Ο περιορισμός [5.14] εκφράζει τις γενικευμένες σχέσεις προτεραίοτητας αρχής-αρχής με τις αντίστοιχες χρονικές καθυστερήσεις μεταξύ των δραστηριοτήτων του έργου. Ο περιορισμός των πόρων [5.15] εξασφαλίζει ότι η συνολική κατανάλωση του κάθε πόρου ανά χρονική περίοδο δεν υπερβαίνει τη συνολική διαθεσιμότητα του πόρου, ενώ ο περιορισμός των πόρων [5.16] καθορίζει το ελάχιστο και το μέγιστο επιτρεπτό όριο της χρησιμοποιούμενης ποσότητας του κάθε πόρου από μία δραστηριότητα ανά χρονική περίοδο. Ο περιορισμός [5.17] εξασφαλίζει ότι η έναρξη κάθε δραστηριότητας θα βρίσκεται αργότερα και νωρίτερα από την έναρξη της βοηθητικής δραστηριότητας αρχής και τέλους, αντίστοιχα. Το γεγονός αυτό υποδηλώνει ότι η πρώτη δραστηριότητα του χρονοπρογράμματος είναι η βοηθητική δραστηριότητα αρχής, ενώ η τελευταία είναι η βοηθητική δραστηριότητα τέλους. Η παράσταση [5.18] καθορίζει ότι η μεταβλητή απόφασης είναι δυαδική. Τέλος, βάσει των ορισμών του άνωθι διατυπωμένου μοντέλου πρέπει να ισχύει η ακόλουθη εξίσωση: $\sum_{q_{ik}=0}^{q_{ik,total}} \sum_{t=0}^{T-1} r_{ik,q} \cdot x_{iktq} = E_{ki}$. Σημειώνεται ότι η εξίσωση αυτή αποτελεί συνεπαγωγή του τρόπου ορισμού των δεδομένων και σκιαγραφεί την απαίτηση ικανοποίησης της απαιτούμενης προσπάθειας κάθε δραστηριότητας ανά απαιτούμενο πόρο.

5 Διαδικασία Επίλυσης

5.1 Εισαγωγή

Ο προτεινόμενος αλγόριθμος επίλυσης, υπό το προσωνύμιο του *"do4u",* του υπό μελέτη προβλήματος του Προγραμματισμού Έργων Προκαθορισμένης Προσπάθειας με Ευέλικτα Προφίλ Πόρων και Γενικευμένες Σχέσεις Προτεραιότητας (*Effort Driven Resource Constrained Project Scheduling with Flexible Resource Profiles and Generilized Precedecne Relations with minimal and maximal Time Lags*), συνίσταται στη βέλτιστη αντιμετώπιση τόσο απλών προβλημάτων RCPSP όσο και RCPSP/max, υπό την αιγίδα επίλυσης των ευέλικτων προφίλ πόρων. Αποτελεί μια μεταευρετική υπολογιστική διαδικασία βελτιστοποίησης, που βασίζεται στη συνδυαστική εξελικτική δυνατότητα δύο εμφωλευμένων γενετικών αλγορίθμων, οι οποίοι αποτελούν τις πιο αποτελεσματικές μεθόδους αντιμετώπισης των προβλημάτων που υπόκεινται στην κατηγορία των RCPSP προβλημάτων (Hartmann and Briskorn, 2010).

Τα συστατικά στοιχεία κατά την επίλυση ενός προβλήματος μέσω του *"do4u"* συνοψίζονται σε τρία επιμέρους στάδια, όπως σκιαγραφούνται στο Σχήμα 14.

1. Προσδιορισμός των παραμέτρων του προβλήματος:

   Τα δεδομένα του προς εκάστοτε επίλυσης προβλήματος αναλύονται ώστε να αποτελέσουν τις απαραίτητες εισόδους για την έναρξη της υπολογιστικής διαδικασίας επίλυσης του αλγορίθμου *"do4u"*. Πιο συγκεκριμένα, προσδιορίζονται η κατηγορία που έγκειται το προς επίλυση έργο, οι δραστηριότητες του έργου, το σύνολο των διαθέσιμων πόρων, η απαιτούμενη προσπάθεια για την ολοκλήρωση κάθε δραστηριότητας ανά απαιτούμενο πόρο, η συνολική διαθεσιμότητα και η παραγωγική δυναμικότητα κάθε πόρου, η ελάχιστη και η μέγιστη επιτρεπόμενη χρησιμοποιούμενη ποσότητα κάθε πόρου ανά χρονική περίοδο, οι σχέσεις προτεραιότητας και οι αντίστοιχες ενδεχόμενες χρονικές καθυστερήσεις μεταξύ των δραστηριοτήτων, ο μέγιστος επιτρεπτός αριθμός διαμερίσεων, το ελάχιστο χρονικό διάστημα σταθερής χρησιμοποιούμενης ποσότητας πόρων. Ανάλογα με την κατηγορία του προς επίλυση προβλήματος και με την μορφή των δεδομένων εισόδου αυτού, γίνονται οι απαραίτητοι μετασχηματισμοί και υπολογισμοί, ώστε τα δεδομένα να αποτελέσουν συμβατές εισόδους για την έναρξη της υπολογιστικής διαδικασίας.

2. Δημιουργία προφίλ εκτέλεσης:

   Για κάθε μία δραστηριότητα του έργου και για κάθε απαιτούμενο πόρο για την ολοκλήρωσή της, δημιουργείται ένα προφίλ εκτέλεσης με σεβασμό σε όλους τους

περιορισμούς (διαθεσιμότητας των πόρων, ελάχιστης και μέγιστης χρησιμοποιούμενης ποσότητας του κάθε πόρου ανά χρονική περίοδο, ελάχιστου χρονικού διαστήματος σταθερής χρησιμοποιούμενης ποσότητας κάθε πόρου), καθώς και στη συνολική απαιτούμενη προσπάθεια από τον πόρο για την ολοκλήρωση της δραστηριότητας. Το προφίλ εκτέλεσης κάθε δραστηριότητας σκιαγραφεί τη χρησιμοποιούμενη ποσότητα από κάθε τύπο πόρων ανά χρονική περίοδο· ο ανά χρονική περίοδο, υπολογισμός, σηματοδοτεί την επερχόμενη χρονική στιγμή έναρξης της δραστηριότητας ως σημείο αναφοράς. Η δημιουργία των προφίλ εκτέλεσης των δραστηριοτήτων του έργου αποτελεί και μία ταυτόχρονη εκτίμηση των άγνωστων διαρκειών τους. Η τεχνική που χρησιμοποιείται για τη δημιουργία κάθε ευέλικτου προφίλ στηρίζεται στον πίνακα διαμερίσεων και στην αντίστοιχη τιμή του συνολικού αριθμού διαμερίσεων ανά δραστηριότητα και πόρο. Ο συνολικός αριθμός διαμερίσεων, σηματοδοτεί τις φορές αλλαγής της χρησιμοποιούμενης ποσότητας του κάθε πόρου από τη δραστηριότητα που αφορά. Σημειώνεται ότι τόσο οι χρησιμοποιούμενες ποσότητες και τα αντίστοιχα χρονικά διαστήματα χρησιμοποίησης ανά διαμέριση, δραστηριότητα και πόρο, όσο και συνολικός προκύπτων αριθμός διαμερίσεων, βασίζονται στην τυχαιότητα της τιμής που λαμβάνουν κάθε φορά από μια γεννήτρια τυχαίων αριθμών εντός των φραγμένων συνόλων τιμών και των περιορισμών που οφείλουν να ικανοποιούν.

3. Εύρεση εφικτού χρονοπρογράμματος:

Δεδομένου του ότι για το σύνολο των δραστηριοτήτων του έργου έχει λάβει χώρα ο υπολογισμός τυχαίων προφίλ εκτέλεσης, το τρίτο στάδιο της επίλυσης αφορά τον υπολογισμό εφικτών χρονοπρογραμμάτων για κάθε δεδομένο σύνολο προφίλ εκτέλεσης των δραστηριοτήτων, με απώτερο στόχο την εύρεση του βέλτιστου χρονοπρογράμματος.

Σχήμα 14. Στάδια κατά την επίλυση του υπό μελέτη προβλήματος

Τα προαναφερθέντα συστατικά στοιχεία αποτελούν τον προθάλαμο στον οποίο θα εκκολαφθεί η εξελικτική υπολογιστική διαδικασία του αλγορίθμου *"do4u"*. Η διαδικασία επίλυσης αποτελείται από δύο εξελικτικούς γενετικούς αλγορίθμους, εκ των οποίων ο εξωτερικός διαχειρίζεται τη δημιουργία προφίλ εκτέλεσης ανά δραστηριότητα και

απαιτούμενο πόρο, συνθέτοντας σύνολα δραστηριοτήτων του έργου με συγκεκριμένα προφίλ εκτέλεσης αυτών, ενώ ο εσωτερικός διαχειρίζεται τη βελτιστοποίηση της συνολικής διάρκειας του έργου με δεδομένα τα υπολογισμένα, πλέον, προφίλ εκτέλεσης της κάθε δραστηριότητας για το σύνολο των δραστηριοτήτων του έργου.

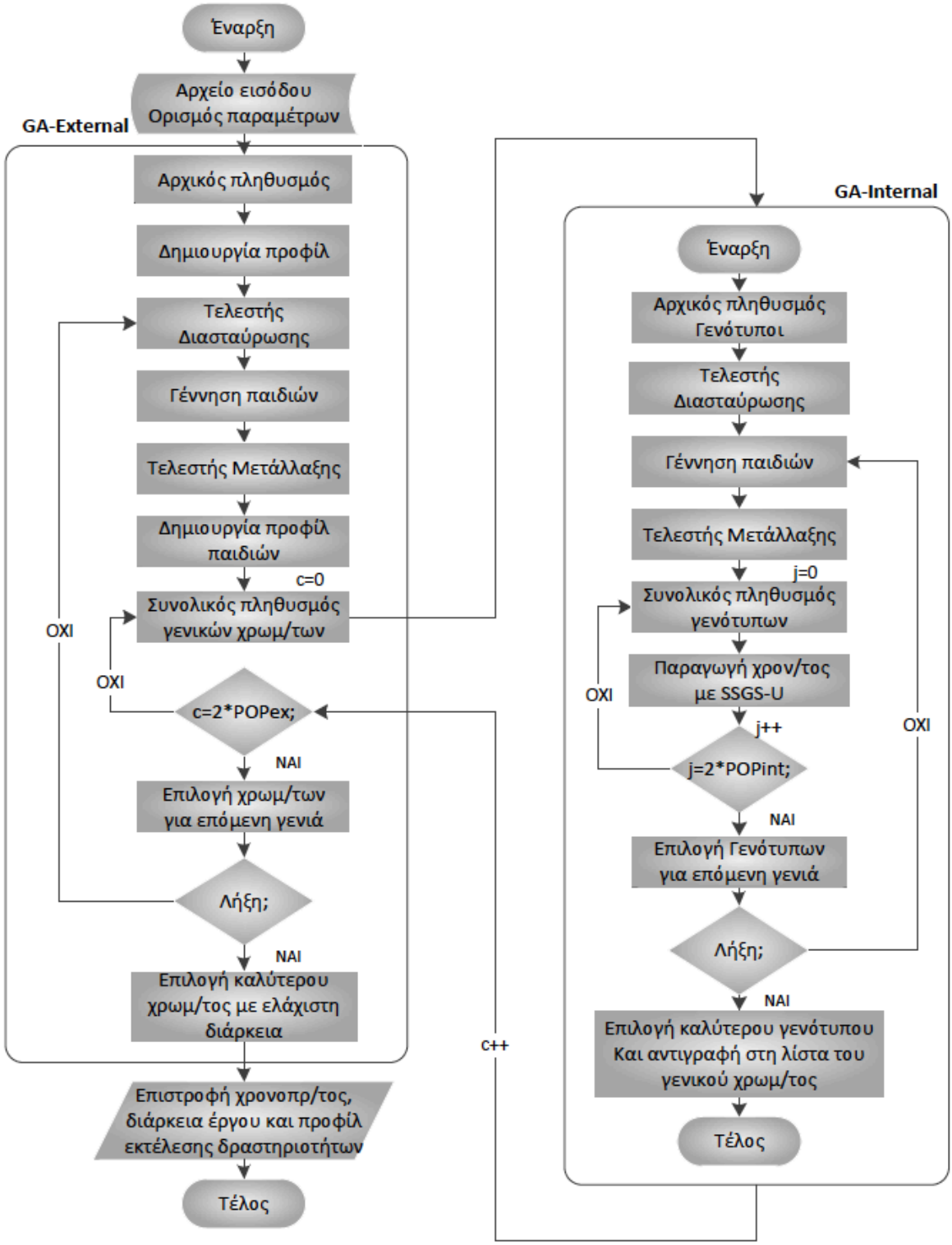
5.2 Προτεινόμενος Αλγόριθμος Επίλυσης

5.2.1 Βασικό Σχέδιο Αλγορίθμου

Ο προτεινόμενος αλγόριθμος επίλυσης *"do4u",* αποτελεί μια συνδυαστική υπολογιστική διαδικασία βελτιστοποίησης ικανή να αντιμετωπίσει το υπό μελέτη πρόβλημα με στόχο τη συνολική ελαχιστοποίηση της διάρκειας του έργου και την υπόδειξη των αντίστοιχων προφίλ εκτέλεσης όλων των δραστηριοτήτων του έργου. Επιπρόσθετα, έχει τη δυνατότητα να αντιμετωπίσει και να επιλύσει αποτελεσματικά και αποδοτικά τόσο το κλασικό RCPSP πρόβλημα όσο και παραλλαγές του, όπως το RCPSP/GPR και το RCPSP/max, δηλαδή το RCPSP με τη χρήση γενικευμένων σχέσεων προτεραιότητας συμπεριλαμβανομένων ελάχιστων ή/και μέγιστων χρονικών καθυστερήσεων, έχοντας απαιτήσεις και περιορισμούς σε ανανεώσιμους πόρους, υπό την αιγίδα επίλυσης και παρουσιάσης των τελικών ευέλικτων προφίλ πόρων.

Ο *"do4u"* αποτελείται από δύο εξελικτικούς αλγορίθμους, τον εξωτερικό γενετικό (*GA-External*) και τον εσωτερικό γενετικό (*GA-Internal*), όπως φαίνεται και στο διάγραμμα ροής του Σχήματος 15. Η δομή αναπαράστασης κάθε γενικού χρωμοσώματος έγκειται σε 3 συστατικά στοιχεία: τη *Λίστα Δραστηριοτήτων* (*actsList*) που αποτελεί την αναπαράσταση των δραστηριοτήτων του έργου που επρόκειτω να προγραμματιστούν, τον *Πίνακα Διαμερίσεων* (*Partition Table*) με το συνολικό αριθμό διαμερίσεων ανά δραστηριότητα και πόρο, και το Προφίλ Εκτέλεσης (*Execution Profile*) κάθε δραστηριότητας ανά απαιτούμενο πόρο υποδηλώνοντας τη χρησιμοποιούμενη ποσότητα ανά χρονική περίοδο ως την ικανοποίηση της απαιτούμενης προσπάθειας της δραστηριότητας.

Ο εξωτερικός γενετικός αλγόριθμος διαχειρίζεται τη δημιουργία προφίλ εκτέλεσης ανά δραστηριότητα και απαιτούμενο πόρο συνθέτοντας χρωμοσώματα με τις δραστηριότητες του έργου να λαμβάνουν συγκεκριμένα τυχαία συνολικά προφίλ εκτέλεσης σε κάθε ένα από αυτά. Η διαδικασία γέννησης κάθε προφίλ εκτέλεσης στηρίζεται στον πίνακα διαμερίσεων, ο οποίος λαμβάνει τυχαίες τιμές εντός του φραγμένου συνόλου από το δεδομένο μέγιστο επιτρεπόμενο αριθμό διαμερίσεων του έργου. Κάθε χρωμόσωμα, με υπολογισμένο πίνακα διαμερίσεων, δύναται να γεννήσει ένα τυχαίο συνολικό προφίλ εκτέλεσης για κάθε δραστηριότητα και αντίστοιχο πόρο που θα ικανοποιεί την απαιτούμενη προσπάθεια για την ολοκλήρωση της δραστηριότητας, μέσα από την υπολογστική διαδικασία του *"do4u"*. Η εξελικτική διαδικασία του εξωτερικού γενετικού αλγορίθμου συνίσταται στις διαδικασίες διασταύρωσης και μετάλλαξης των πινάκων διαμερίσεων των χρωμοσωμάτων του αρχικού πληθυσμού. Η νέα γενιά χρωμοσωμάτων με υπολογισμένους πλέον πίνακες διαμερίσεων, υπόκειται στη διαδικασία υπολογισμού των προφίλ εκτέλεσης όλων των δραστηριοτήτων με σεβασμό σε όλους τους περιορισμούς και τις απαιτήσεις του εκάστοτε προβλήματος. Για κάθε χρωμόσωμα του συνολικού πληθυσμού πλέον, έχουν υπολογιστεί τα προφίλ εκτέλεσης κάθε δραστηριότητας και κατά συνέπεια έχουν εκτιμηθεί και οι άγνωστες ως τώρα διάρκειες των δραστηριοτήτων ανά αντίστοιχο προφίλ.



Σχήμα 15. Διάγραμμα ροής αλγόριθμου επίλυσης "do4u"

Ο εσωτερικός γενετικός αλγόριθμος καλείται για κάθε ένα χρωμόσωμα του συνολικού πληθυσμού του εξωτερικού γενετικού αλγορίθμου και επιστρέφει το βέλτιστο χρονοπρόγραμμα στη λίστα δραστηριοτήτων του χρωμοσώματος αυτού. Διαχειρίζεται την αναζήτηση του βέλτιστου χρονοπρογράμματος στο χώρο λύσεων του προβλήματος και κατ' επέκταση τη βελτιστοποίηση της συνολικής διάρκειας του έργου μέσα από τον χρονοπρογραμματισμό των δραστηριοτήτων κάθε ενός γενικού χρωμοσώματος, με δεδομένα

τα υπολογισμένα πλέον προφίλ εκτέλεσης της κάθε δραστηριότητας. Πιο συγκεκριμένα, με εισόδους τα δεδομένα ενός γενικού χρωμοσώματος, δημιουργεί έναν αρχικό πληθυσμό από γενότυπους (γίνεται η χρήση του όρου γενότυπου εν αποφυγή σύγχυσης μεταξύ αυτού και του γενικού χρωμοσώματος), κάθε ένας από τους οποίους αποτελείται από μια τυχαία λίστα δραστηριοτήτων με σεβασμό στις σχέσεις προτεραιότητας των δραστηριοτήτων του έργου. Έπειτα, λαμβάνουν χώρα οι εξελικτικές διαδικασίες της κλασικής διασταύρωσης και μετάλλαξης μεταξύ των γενότυπων του αρχικού πληθυσμού, συνθέτοντας ένα συνολικό πληθυσμό γενότυπων. Στο σημείο αυτό σημειώνεται ότι όλοι οι γενότυποι αποτελούν δυνητικές λύσεις του αρχικού γενικού χρωμόσωματος που αφορούν. Για κάθε γενότυπο υλοποιείται η ευρετική μέθοδος της σειριακής παραγωγής χρονοπρογραμμάτων για την αποκωδικοποίηση και τη δημιουργία ενός εφικτού χρονοπρογράμματος με συνεκτίμηση της συνολικής διάρκειας του έργου. Οι γενότυποι με την καλύτερη τιμή αντικειμενικής συνάρτησης συνολικής διάρκειας του έργου, επιλέγονται για τον αρχικό πληθυσμό της επόμενης γενιάς λύσεων. Κατά το πέρας του εσωτερικού γενετικού αλγορίθμου, η χρονικά προγραμματισμένη λίστα δραστηριοτήτων του καλύτερου γενότυπου, αποτελεί τη λίστα δραστηριοτήτων του γενικού χρωμοσώματος που κάλεσε τον εσωτερικό αλγόριθμο. Με αυτό τον τρόπο, κάθε γενικό χρωμόσωμα του εξωτερικού αλγορίθμου λαμβάνει ένα βέλτιστο, σύμφωνα με τα δεδομένα προφίλ του, χρονοπρόγραμμα, και εν αντιστοιχία με την τιμή της αντικειμενικής συνάρτησης της διάρκειας του έργου του χρονοπρογράμματος που έχει κωδικοποιήσει, αξιολογείται και κρίνεται η επιλογή του για την επόμενη γενιά του εξωτερικού γενετικού αλγορίθμου.

Οι διαδικασίες αξιολόγησης και επιλογής κάθε γενικού χρωμοσώματος του εξωτερικού γενετικού αλγορίθμου, καθώς και κάθε γενότυπου του εσωτερικού γενετικού αλγορίθμου, πραγματοποιούνται σύμφωνα με τις τιμές αντικειμενικής συνάρτησης που τους αποδίδονται από το πέρας του εσωτερικού γενετικού και από κάθε παραγόμενο χρονοπρόγραμμα μέσα από την σειριακή παραγωγή χρονοπρογραμμάτων, αντίστοιχα. Ο προτεινόμενος αλγόριθμος επίλυσης *"do4u"*, με το πέρας της επαναληπτικής διαδικασίας του εξωτερικού γενετικού αλγορίθμου για τον απαιτούμενο αριθμό γενεών, τερματίζεται και το καλύτερο χρωμόσωμα αποτελεί την κωδικοποιημένη λύση του υφιστάμενου προβλήματος. Ως έξοδοι δίδονται η συνολική διάρκεια του έργου, οι χρονικές στιγμές έναρξης και οι διάρκειες των δραστηριοτήτων του έργου, καθώς και οι χρησιμοποιούμενες ποσότητες κάθε πόρου από κάθε δραστηριότητα ανά χρονική περίοδο. Τέλος, η υπολογιστική διαδικασία του *"do4u"*, μπορεί να τερματιστεί αβίαστα, σε καταστάσεις όπου η αντικειμενική συνάρτηση κάθε χρωμοσώματος του πληθυσμού του εξωτερικού γενετικού αλγορίθμου έχει λάβει μια πρακτικά άπειρη τιμή, λόγω της αδυναμίας παραγωγής εφικτών χρονοπρογραμμάτων.

5.2.2 Δομή Αναπαράστασης Χρωμοσώματος

Ο προτεινόμενος αλγόριθμος επίλυσης *"do4u"* αποτελεί μια υπολογιστική διαδικασία επίλυσης προβλημάτων προγραμματισμού έργων μέσα από τη δημιουργία ευέλικτων προφίλ πόρων. Κατά το πέρας της υπολογιστικής διαδικασίας, πέραν του βέλτιστου χρονοπρογράμματος, επιστρέφεται και το προφίλ εκτέλεσης όλων των δραστηριοτήτων ανά απαιτούμενο πόρο και χρονική μονάδα από τη στιγμή της έναρξής του. Αναμφισβήτητα, στη δομή αναπαράστασης κάθε χρωμοσώματος υπάρχει η ανάγκη αντικατοπτρισμού όλων των προαναφερθέντων συστατικών στοιχείων κατά την επίλυση.

Η δομή αναπαράστασης κάθε γενικού χρωμοσώματος αποτελείται από σε 3 συστατικά στοιχεία:

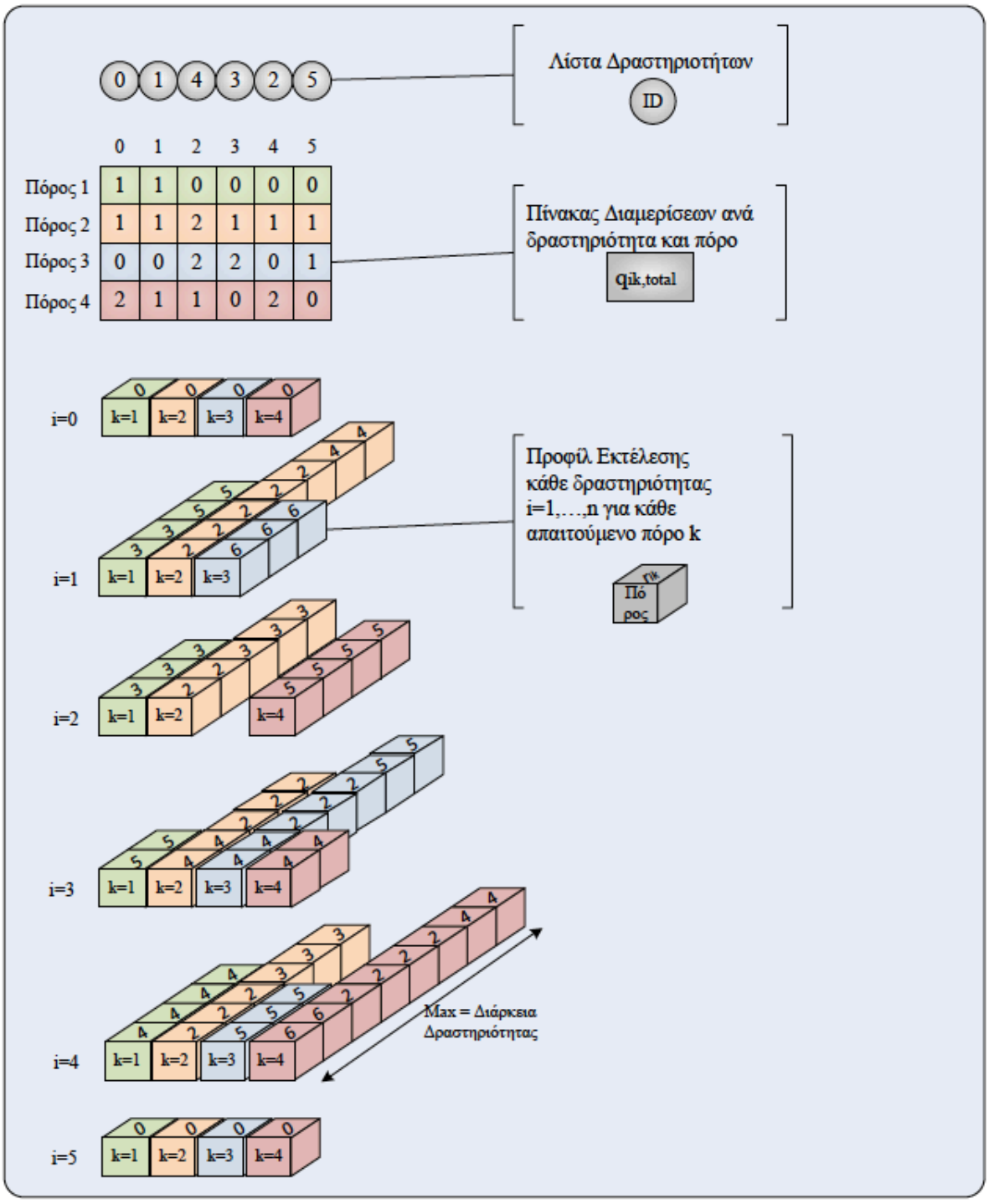
- τη *Λίστα Δραστηριοτήτων* (*actsList*), που αποτελεί την αναπαράσταση των δραστηριοτήτων του έργου που επρόκειτω να προγραμματιστούν.
- τον *Πίνακα Διαμερίσεων* (*Partition Table*) με το συνολικό αριθμό διαμερίσεων ανά δραστηριότητα και πόρο.
- το *Προφίλ Εκτέλεσης* (*Execution Profile*) κάθε δραστηριότητας ανά απαιτούμενο πόρο, υποδηλώνοντας τη χρησιμοποιούμενη ποσότητα ανά χρονική περίοδο ως την ικανοποίηση της απαιτούμενης προσπάθειας της δραστηριότητας από τον αντίστοιχο πόρο.

Ο *Πίνακας Διαμερίσεων* αποτελεί ένα δισδιάστατο διάνυσμα, όπου οι στήλες αντιπροσωπεύουν τις δραστηριότητες του έργου και οι γραμμές τους υφιστάμενους ανανεώσιμους πόρους. Κάθε τιμή υποδηλώνει το συνολικό αριθμό διαμερίσεων που θα περιλαμβάνει το προφίλ εκτέλεσης της δραστηριότητας υπό τη χρήση του αντίστοιχου πόρου. Όπως έχει ήδη αποσαφηνιστεί, ο Πίνακας Διαμερίσεων αποτελεί κομβικό στοιχείο κάθε γενικού χρωμοσώματος, καθώς η γέννηση ενός προφίλ εκτέλεσης αναφορικά με μία δραστηριότητα και έναν απαιτούμενο πόρο, στηρίζεται στην αντίστοιχη τιμή του συνολικού αριθμού διαμερίσεων του πίνακα. Επιπρόσθετα, οι εξελικτικές διαδικασίες της διασταύρωσης και της μετάλλαξης των γενικών χρωμοσωμάτων του εξωτερικού γενετικού αλγορίθμου, λαμβάνουν χώρα αναφορικά με τους πίνακες διαμερίσεων. Σημειώνεται ότι οι τιμές που λαμβάνει ο πίνακας διαμερίσεων κάθε γενικού χρωμοσώματος του αρχικού πληθυσμού του εξωτερικού γενετικού αλγορίθμου, βασίζονται στην τυχαιότητα της τιμής που λαμβάνουν κάθε φορά από μια γεννήτρια τυχαίων αριθμών εντός των φραγμένων συνόλων τιμών και των περιορισμών που οφείλουν να ικανοποιούν. Παράλληλα, ο Πίνακας Διαμερίσεων των γενικών χρωμοσωμάτων-παιδιών λαμβάνει τιμές μέσα από την υλοποίηση των τελεστών διασταύρωσης και μετάλλαξης των πινάκων διαμερίσεων των χρωμοσωμάτων-γονέων σε κάθε γενιά λύσεων.

Το *Προφίλ Εκτέλεσης* αποτελεί ένα τρισδιάστατο διάνυσμα, όπου οι δύο διαστάσεις αντιπροσωπεύουν τη δραστηριότητα και τον πόρο που αφορά το προφίλ, ενώ η τρίτη διάσταση αντιπροσωπεύει την χρησιμοποιούμενη ποσότητα του πόρου ανά χρονική περίοδο· η αναπαράσταση ανά χρονική περίοδο σηματοδοτεί την επερχόμενη χρονική στιγμή έναρξης της δραστηριότητας ως σημείο αναφοράς. Για κάθε δραστηριότητα ενός γενικού χρωμοσώματος και για κάθε απαιτούμενο πόρο, δημιουργείται ένα τυχαίο προφίλ εκτέλεσης με σεβασμό σε όλους τους περιορισμούς, καθώς και στη συνολική απαιτούμενη προσπάθεια από τον πόρο για την ολοκλήρωση της δραστηριότητας. Στο Κεφάλαιο 5.2.3.1 υλοποιείται μια λεπτομερής παρουσίαση της διαδικασίας γέννησης ενός προφίλ εκτέλεσης.

Η *Λίστα Δραστηριοτήτων* αποτελεί ένα διάνυσμα, κάθε θέση του οποίου αντιπροσωπεύει το αναγνωριστικό (*id*) της δραστηριότητας του έργου. Επιπρόσθετα, συνιστά μια εφικτή, αναφορικά με τις υφιστάμενες σχέσεις προτεραιότητας, λίστα δραστηριοτήτων, υπό την έννοια της τοποθέτησης κάθε δραστηριότητας στο διάνυσμα της λίστας, σε θέση ύστερη από αυτή όλων των άμεσων προαπαιτούμενων δραστηριοτήτων της (*direct predecessors*). Οι βοηθητικές δραστηριότητες αρχής και τέλους βρίσκονται πάντοτε στην πρώτη και στην τελευταία θέση, αντίστοιχα, του διανύσματος. Η Λίστα Δραστηριοτήτων κάθε γενικού χρωμοσώματος αποτελεί το γενότυπο κάθε εσωτερικού γενετικού αλγόριθμου και λαμβάνει τη μορφή της κατά την επίλυση αυτού, ενώ ο Πίνακας Διαμερίσεων και το Προφίλ Εκτέλεσης αποτελούν το χρωμόσωμα της διαδικασίας επίλυσης του εξωτερικού γενετικού αλγορίθμου.

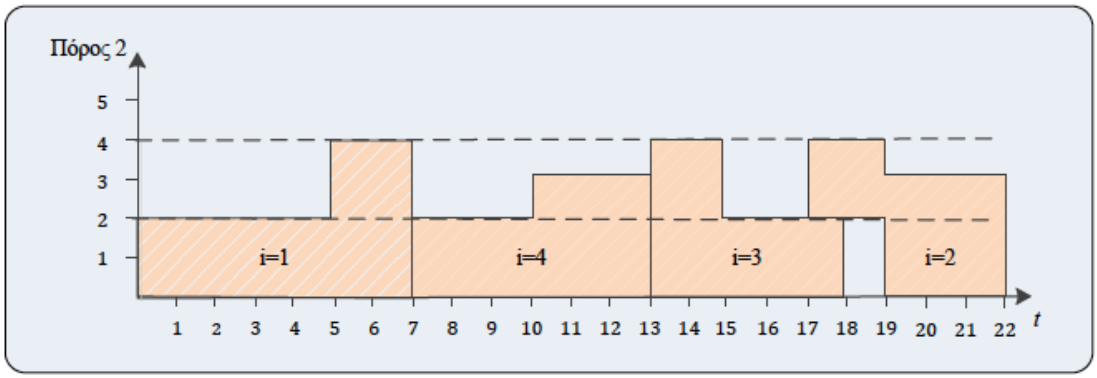
Στο Σχημα 16 παρουσιάζεται μια τυπική δομή αναπαράστασης ενός γενικού χρωμοσώματος.



Σχήμα 16. Δομή αναπαράστασης χρωμοσώματος

Η Λίστα Δραστηριοτήτων αποτελείται από τις δραστηριότητες του παραδείγματος με ικανοποίηση των σχέσεων προτεραιότητας, ενώ ο Πίνακας Διαμερίσεων λαμβάνει τυχαίες, μη αρνητικές, τιμές μεταξύ του διαστήματος που ορίζει ο μέγιστος επιτρεπόμενος αριθμός διαμερίσεων. Παράλληλα, το Προφίλ Εκτέλεσης ανά δραστηριότητα και πόρο, περιλαμβάνει τυχαίες τιμές χρησιμοποιούμενης ποσότητας ανά χρονική περιόδο της κάθε διαμέρισης που ικανοποιούν την απαιτούμενη προσπάθεια της κάθε δραστηριότητας από τον αντίστοιχο

πόρο, χωρίς να παραβιάζει τους δεδομένους περιορισμούς. Τέλος, παρουσιάζεται ένα εφικτό χρονοπρόγραμμα του πόρου 2, αναφορικά με τα υπολογισμένα προφίλ εκτέλεσης όλων των δραστηριοτήτων σε σχέση με αυτόν.



Σχήμα 17. Εφικτό χρονοπρόγραμμα πόρου 2 του Σχήματος 16

Για την κατά το δυνατόν καλύτερη επεξήγηση της τυπικής δομής ενός γενικού χρωμοσώματος, στον ακόλουθο πίνακα δίδονται τα στοιχεία του παραδείγματος του Σχήματος 16 :

Πίνακας 9. Δεδομένα παραδείγματος τυπικού χρωμοσώματος Σχηματος 16

Δραστηριότητες	**Απαιτούμενη Προσπάθεια**				**Προαπαι-**
	Πόρος 1	**Πόρος 2**	**Πόρος 3**	**Πόρος 4**	**τούμενοι**
0	0	0	0	0	-
1	16	18	18	20	4
2	9	13	0	8	1
3	10	14	24	28	0,1
4	16	15	15	0	2,3
5	0	0	0		

Πόροι	**Διαθεσιμότητα**	**Ελάχιστη Χρησιμοποιούμενη Πόσοτητα ανά περίοδο**	**Μέγιστη Χρησιμοποιούμενη Πόσοτητα ανά περίοδο**
1	5	1	5
2	4	1	4
3	6	1	6
4	6	1	6
Μέγιστος επιτρεπόμενος αριθμός διαμερίσεων			2
Ελάχιστο χρονικό διάστημα σταθερής χρησιμοποιούμενης ποσότητας			1

5.2.3 Εξωτερικός Γενετικός Αλγόριθμος

Ο εξωτερικός γενετικός αλγόριθμος, ο ψευδοκώδικας του οποίου παρουσιάζεται στον Αλγόριθμο 6, διαχειρίζεται τη δημιουργία προφίλ εκτέλεσης ανά δραστηριότητα και απαιτούμενο πόρο, συνθέτοντας χρωμοσώματα με τις δραστηριότητες του έργου να λαμβάνουν συγκεκριμένα τυχαία συνολικά προφίλ εκτέλεσης σε κάθε ένα από αυτά. Η διαδικασία γέννησης κάθε προφίλ εκτέλεσης στηρίζεται στον πίνακα διαμερίσεων, ο οποίος

λαμβάνει τυχαίες τιμές εντός του φραγμένου συνόλου από το δεδομένο μέγιστο επιτρεπόμενο αριθμό διαμερίσεων του έργου. Κάθε χρωμόσωμα με υπολογισμένο πίνακα διαμερίσεων δύναται να γεννήσει ένα τυχαίο συνολικό προφίλ εκτέλεσης για κάθε δραστηριότητα και αντίστοιχο πόρο που θα ικανοποιεί την απαιτούμενη προσπάθεια για την ολοκλήρωση της δραστηριότητας. Η εξελικτική διαδικασία του εξωτερικού γενετικού αλγορίθμου συνίσταται στις διαδικασίες των τελεστών διασταύρωσης και μετάλλαξης των πινάκων διαμερίσεων των χρωμοσωμάτων του αρχικού πληθυσμού. Η νέα γενιά χρωμοσωμάτων με υπολογισμένους πλέον πίνακες διαμερίσεων, υπόκειται στη διαδικασία υπολογισμού των προφίλ εκτέλεσης όλων των δραστηριοτήτων με σεβασμό σε όλους τους περιορισμούς και τις απαιτήσεις. Για κάθε χρωμόσωμα του συνολικού πληθυσμού, πλέον, έχουν υπολογιστεί τα προφίλ εκτέλεσης κάθε δραστηριότητας και κατά συνέπεια έχουν εκτιμηθεί και οι άγνωστες ως τώρα διάρκειες των δραστηριοτήτων ανά αντίστοιχο προφίλ.

Εν συνεχεία, για κάθε γενικό χρωμόσωμα του συνολικού πληθυσμού καλείται ο εσωτερικός γενετικός αλγόριθμος, ο οποίος πραγματοποιεί βελτιστοποίηση χρόνου μέσα από την παραγωγή του βέλτιστου χρονοπρογράμματος για τα δεδομένα προφίλ εκτέλεσης του χρωμοσώματος. Η χρονοπρογραμματισμένη λίστα που επιστρέφει κάθε εσωτερικός γενετικός αλγόριθμος, αποτελεί τη Λίστα Δραστηριοτήτων του χρωμοσώματος που τον κάλεσε. Η επιλογή των χρωμοσωμάτων που θα αποτελέσουν τον αρχικό πληθυσμό της επόμενης γενιάς, λαμβάνει χώρα με κριτήριο τη συνολική διάρκεια του έργου, που έχει αποδοθεί στην αντικειμενική συνάρτηση κάθε χρωμοσώματος μέσα από το χρονοπρογραμματισμό του εσωτερικού γενετικού αλγορίθμου. Στο σημείο αυτό, σημειώνεται ότι σε περιπτώσεις που για ένα χρωμόσωμα με το δεδομένο Προφίλ Εκτέλεσής του, ο εσωτερικός γενετικός αλγόριθμος δεν έχει τη δυνατότητα να επιστρέψει ένα εφικτό χρονοπρόγραμμα, τότε το χρωμόσωμα αντικαθίσταται από ένα νέο μέσω της διακασίας του ελιτισμού (*elitism*). Ο εξωτερικός γενετικός αλγόριθμος με την παραγωγή του δεδομένου αριθμού γενιών, τερματίζεται και το καλύτερο χρωμόσωμα αποτελεί τη λύση του υφιστάμενου προβλήματος. Επιστρέφει, λοιπόν, τη συνολική διάρκεια του έργου, τις χρονικές στιγμές έναρξης και τις διάρκειες των δραστηριοτήτων του έργου, καθώς και τις χρησιμοποιούμενες ποσότητες κάθε πόρου από κάθε δραστηριότητα ανά χρονική περίοδο. Τέλος, η υπολογιστική διαδικασία του εξωτερικού γενετικού αλγορίθμου, ενδέχεται να τερματιστεί σε καταστάσεις όπου η αντικειμενική συνάρτηση κάθε χρωμοσώματος του πληθυσμού έχει λάβει μια πρακτικά άπειρη τιμή, λόγω της αδυναμίας παραγωγής εφικτών χρονοπρογραμμάτων.

Αλγόριθμος 6. Ψευδοκώδικας GA-External

Input : jobNumber, resourceNumber, projectActivities, GenerilizedPrecedenceRelations, TimeLags, resourceAvailabilities, actEfforts, ProductionCapacities, Min/MaxResourcesUsage, MaxPartitionnumber, MinAllocationTimes
Output : makespan, schedule, actsProfiles
`Set` populationSize = POP_{ext};
`Set` evolutionSteps = e;
`Set` timeHorizon = T;
`Set` probabilityOfMutationChromExt = $p_{mut,Chrom}$;
`Set` probabilityOfMutationResourceExt = $p_{mut,Res}$;
`Set` generationCounter $g = 0$;
`//Generate initial population` P_0 `of` POP_{ext} `individuals;`
for $c = 0 \dots POP_{ext}$ **do**
    $P_0 = P_0 \cup$ `GenerateRandomCromosomes;`
    $P_0[c].actsList = projectActivities$;
    `//Give random values in partition tables`

```
    for i = 0 ... jobNumber do
        for j = 0 ... resourceNumber do
            P_0[c].partition[j][i] = rand.nextInt(MaxPartitionNum + 1);
            //Compute Profile
            P_0[c].profile = computeProfile(actEfforts_{i,j}, productionCapacities_j, T,
                P_0[c].partition[j][i], min/maxResourcesUsage_j, minAllocationTime_i)
        end
    end
end
P_g = P_0;
while stopping criteria not met do
    //Generate offspring population P_{g,children}
    //One-point Crossover and Mutation based on the partition tables
    P_{g,children} = Crossover P_g;
    P_{g,children} = Mutate P_g;
    //Compute profiles for children
    for c = 0 ... POP_ext do
        for i = 0 ... jobNumber do
            for j = 0 ... resourceNumber do
                P_0[c].profile = computeProfile(actEfforts_{i,j}, productionCapacities_j, T,
                    P_0[c].partition[j][i], min/maxResourcesUsage_j, minAllocationTime_i)
            end
        end
    end
    //Combine current and offspring population
    R_g = P_g ∪ P_{g,children};
    //Calculate activity durations per resource and total activity
      duration (max of durations)
    //Compute fitness value for each chromosome through GA-Internal
    for i = 0 ... 2 * POP_ext do
        R_g[i].fitnessValue = GA − Internal( R_g[i], ResourceAvailabilities, jobNumber,
                                             resourceNumber);
        if R_g[i].fitnessValue == 1.000.000.000 then
            stop++;
            K.add(R_g[i]);
        end
    end
    //If no feasible schedule have been found then exit
    if stop == 2 * POP_ext then
        exit;
    end
    //Elitism
    /* for each chromosome with duplicates or with fv=1000000000
       generate new chromosome*/
    R_g\K;
    while stopping criteria not met do
        Generate new chromosome chrom;
        chrom.actsList = projectActivities;
        for i = 0 ... jobNumber do
            for j = 0 ... resourceNumber do
                chrom.partition[j][i] = rand.nextInt(MaxPartitionNum + 1);
                chrom.profile = computeProfile(actEfforts_{i,j}, productionCapacities_j, T,
```

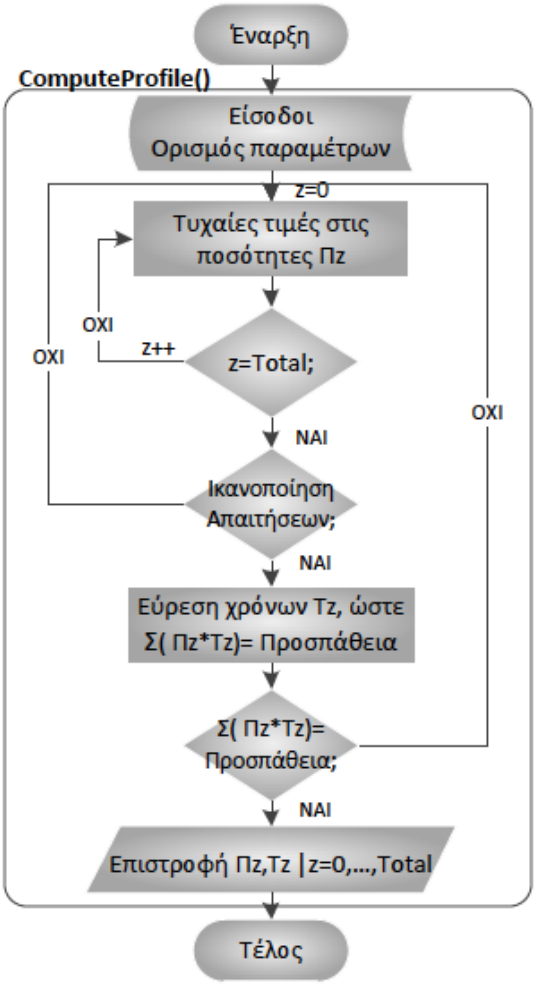
chrom. partition[j][i], min/maxResourcesUsage$_j$, $minAllocationTime_i$);
        end
    end
    $chrom.fitnessValue = \boldsymbol{GA - Internal}$(chrom, ResourceAvailabilities, jobNumber, resourceNumber);
    $R_g = R_g \cup chrom$;
  end
  //Select POP_{ext} chromosomes with best fitnnesValues for next //generation
  P_g = RankingMethod(R_g);
  $g = g + 1$;
end
//Select chromosome with best fitness value and put it to
$bestChromosome = P_g$[i] | min { $P_g[i].fitnessValue$, $\forall i = 0, \dots, POP_{int}$};
$chromoExternal.actsList = bestIndividual$;
$makespan = bestChromosome.fitnessValue$;
//Schedule S with start times ST of all project activities
$S = \{bestChromosome.actsList[i].ST, \forall i \in projectActivites\}$;
return(makespan**)**;
return(Schedule S**)**;
return(bestChromosome.profile[i,j] | ∀i ∈ projectActivites, j ∈ projectResource**)**;

5.2.3.1 Δημιουργία Προφίλ Εκτέλεσης

Για κάθε μία δραστηριότητα του έργου που περιλαμβάνει κάθε γενικό χρωμόσωμα, και για κάθε απαιτούμενο πόρο δημιουργείται ένα προφίλ εκτέλεσης, αναφορικά με τους υφιστάμενους περιορισμούς, τον Πίνακα Διαμερίσεων του χρωμοσώματος και την απαιτούμενη προσπάθεια προς ικανοποίηση. Οι περιορισμοί που λαμβάνουν χώρα κατά τη δημιουργία ενός προφίλ έγκεινται σε περιορισμούς ελάχιστης και μέγιστης χρησιμοποιούμενης ποσότητας κάθε πόρου ανά χρονική στιγμή, συνολικού αριθμού διαμερίσεων, καθώς και ελάχιστου χρονικού διαστήματος σταθερής χρησιμοποιούμενης ποσότητας ανά διαμέριση.

Ειδικότερα, για μία δραστηριότητα i και για έναν πόρο k με παραγωγική δυναμικότητα din, υπάρχει η δεδομένη απαιτούμενη προσπάθεια E_{ik} που απαιτείται από τον πόρο k για την ολοκλήρωση της δραστηριότητας i. Απώτερος στόχος είναι η εύρεση της χρησιμοποιούμενης ποσότητας του πόρου k από την i ανά χρονική στιγμή, καθώς και η αντίστοιχη διάρκεια χρησιμοποίησης του πόρου, ώστε να ικανοποιείται η προσπάθεια E_{ik}. Ανάλογα με τον προκύπτων αριθμό διαμερίσεων για το προφίλ της δραστηριότητας i υπό τον πόρο k, έστω z, επιδιώκεται να βρεθεί το γινόμενο ποσότητας επί χρόνου χρησιμοποίησης, έστω $r_z * t_z$, $z+1$ ζευγών, το συνολικό άθροισμα των οποίων θα ισοδυναμεί με την απαιτούμενη προσπάθεια δια της παραγωγικής δυναμικότητας του πόρου, δηλαδή $\sum_0^{z+1} r_z * t_z = E_{ik}/din$. Η τεχνική που χρησιμοποιείται, όπως φαίνεται και στον ψευδοκώδικα της διαδικασίας υπολογισμού προφίλ (Αλγόριθμος 7) καθώς και στο διάγραμμα ροής του Σχήματος 18, είναι να δίδονται τυχαίες τιμές στις ποσότητες των $z+1$ ζευγών εντός του άνω και κάτω φραγμένου συνόλου που ορίζουν η ελάχιστη και μέγιστη επιτρεπόμενη χρησιμοποιούμενη ποσότητα του πόρου k, αντίστοιχα. Σημειώνεται ότι εφαρμόζονται περιορισμοί στις δοσμένες ποσότητες για την αποφυγή επερχόμενων μη υλοποιήσιμων σεναριών με μη αποδεκτές ποσότητες, οι τιμές των οποίων επαναυπολογίζονται μέχρις ότου ικανοποιηθούν οι περιορισμοί. Χαρακτηριστικά αναφέρεται ένα μη εφικτό σενάριο, όπου σε περίπτωση που το άθροισμα των δοσμένων ποσοτήτων επί του ελάχιστου επιτρεπτού

χρονικού διαστήματος χρησιμοποίησης προκύπτει μεγαλύτερο από την απαιτούμενη προσπάθεια, δίδονται νέες ποσότητες.



Σχήμα 18. Διάγραμμα ροής διαδικασίας υπολογισμού προφίλ εκτέλεσης

Έπειτα, με αποδεκτές πλέον τις χρησιμοποιούμενες ποσότητες ανά διαμέριση, αναζητούνται τα αντίστοιχα χρονικά διαστήματα χρησιμοποίησης, τα επιμέρους γινόμενα των οποίων θα αθροίζουν σε E_{ik}/din. Πραγματοποιείται μια επαναληπτική διαδικασία εύρεσης των χρονικών διαστημάτων που αντιστοιχούν σε κάθε διαμέριση, μέσω της βηματικής αύξησης κάθε χρόνου ανά διαμέριση. Σε κάθε βήμα αύξησης του χρόνου μιας διαμέρισης, με το χρονικό ορίζοντα προγραμματισμού να αποτελεί το άνω όριο, και με σταθερές τιμές στους χρόνους των υπολοίπων διαμερίσεων, ελέγχεται η ικανοποίηση της συνολικής απαιτούμενης προσπάθειας. Σε περίπτωση που δεν βρεθούν χρονικά διαστήματα ώστε με τις αντίστοιχες ποσότητες να ικανοποιούν τη συνολική απαιτούμενη προσπάθεια, δίδονται νέες τιμές στις ποσότητες των διαμερίσεων και η διαδικασία επαναλαμβάνεται, μέχρις ότου βρεθούν τα απαραίτητα γινόμενα ποσοτήτων και χρονικών διαστημάτων ανά διαμέριση που θα αθροίζουν στο ζητούμενο αποτέλεσμα.

Στο σημείο αυτό, αναφέρεται ότι η διαδικασία εύρεσης τυχαίων τιμών στις ποσότητες της κάθε διαμέρισης προηγείται του υπολογισμού των αντίστοιχων χρονικών διαστημάτων χρησιμοποίησης, καθώς επιδιώκεται η υποβοήθηση της υπολογιστικής προσπάθειας και η αποφυγή υπολογισμού όλων των επιμέρους δυνατών συνδυασμών για να ακολουθήσει η επιλογή από αυτούς. Επιπρόσθετα, πριν την έναρξη της δημιουργίας ενός προφίλ, έχει

ελεγχθεί η δυνατότητα εύρεσης ενός εφικτού προφίλ που ικανοποιεί την απαιτούμενη προσπάθεια βάσει των δεδομένων παραμέτρων. Το γεγονός αυτό αιτιολογεί την αέναη προσπάθεια εύρεσης συνδυασμών ζευγών ποσοτήτων και χρονικών διαστημάτων, ανάλογα με τον αριθμό διαμερίσεων, μέχρις ότου να αθροίσουν στο ζητούμενο αποτέλεσμα.

Αλγόριθμος 7. Ψευδοκώδικας υπολογισμού Προφίλ Εκτέλεσης-ComputeProfile()

Input : effort, productionCapacity, timeHorizon, partitionNumber, minResUsage, maxResUsage, minAllocationTime
Output : profile[][]
Set profile = [2][partitionNumber + 1];
Set partitionNumber = q_{total};
Set minResUsage = r_{min};
Set maxResUsage = r_{max};
Set productionCapacity = din;
Set timeHorizon = T;
Set minAllocationTime = $t_{alloc,min}$;
Set okProducts = false;
/*Create as many pairs of quantity*time as partitionNum, the total sum of which equals Effort*/
while *stopping criteria not met* **do**
    //Give random values in quantities profile[0][] for each pair
    while *stopping criteria not met* **do**
        for $z = 0 \dots q_{total} + 1$ **do**
            $profile[0][z] = rand.nextInt(r_{max} - r_{min}) + r_{min}$;
            $t[z] = t_{alloc,min}$;
        end
    end
    //Compute total time period of resource usage for each quantity in the corresponding partition
    do
        $computeEff == 0$;
        for $i = 0 \dots q_{total} + 1$ **do**
            $computeEff += profile[0][z] * t[i]$;
        end
        if $effort == computeEff * din$ **then**
            $okProducts = true$;
            for $z = 0 \dots q_{total} + 1$ **do**
                $profile[1][z] = t[z]$;
            end
        end
        $o = 0$;
        for $i = 0 \dots q_{total}$ **do**
            if $t[i] \geq T$ **then**
                o++;
                $t[i + 1]$++;
                $t[i] = t_{alloc,min}$;
            end
        end
        $t[z]$++;
    while (*okProducts==false***);**
end
return(profile[][]**);**

5.2.3.2 Αρχικός πληθυσμός

Ο αρχικός πληθυσμός του εξωτερικού γενετικού αλγορίθμου αποτελείται από γενικά χρωμοσώματα μεγέθους ίσου με τον επιλεγμένο αριθμό πληθυσμού (*population size external chromosomes* - POP_{ext}). Τα συστατικά στοιχεία των γενικών χρωμοσωμάτων που υπολογίζονται κατά την παραγωγή του αρχικού πληθυσμού είναι ο Πίνακας Διαμερίσεων και το Προφίλ Εκτέλεσης.

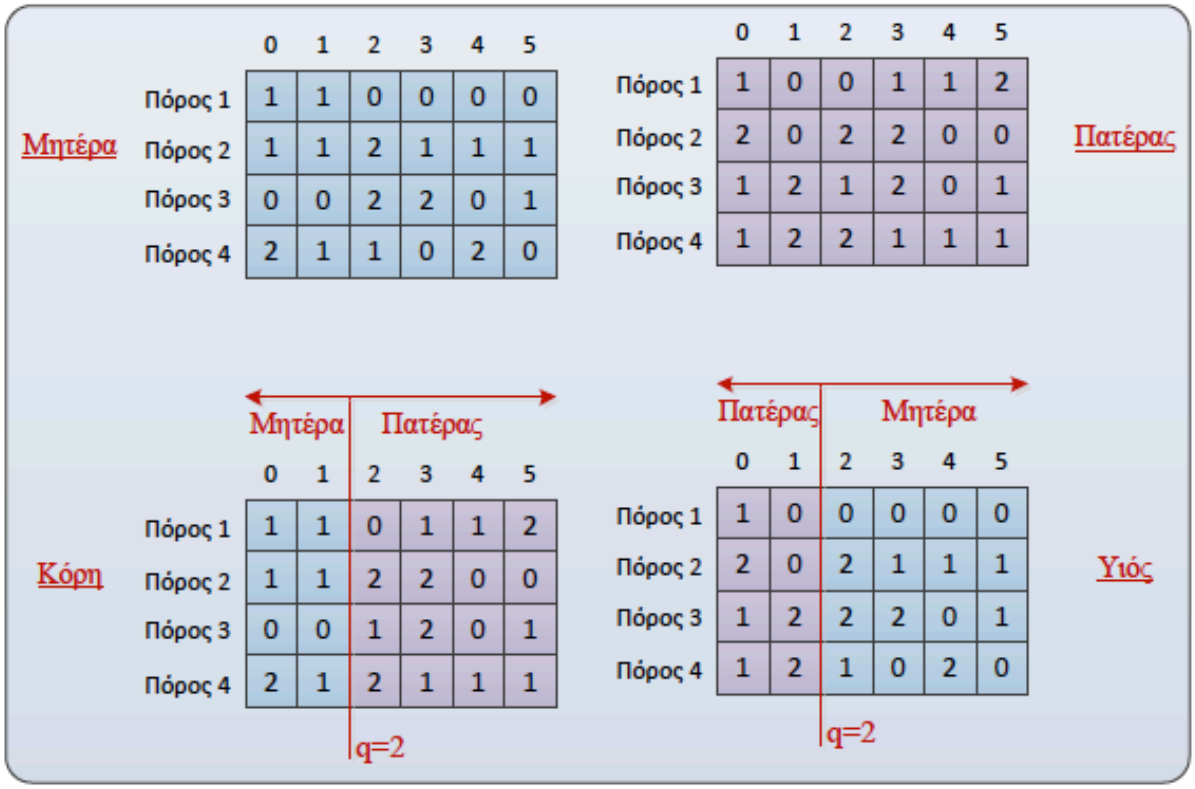
Ειδικότερα, για κάθε ένα χρωμόσωμα παράγεται ο Πίνακας Διαμερίσεων λαμβάνοντας τυχαίες μη αρνητικές τιμές από μια γεννήτρια τυχαίων αριθμών, εντός του άνω φραγμένου συνόλου τιμών που ορίζει ο δεδομένος μέγιστος επιτρεπόμενος αριθμός διαμερίσεων. Ύστερα, παράγεται το συνολικό Προφίλ Εκτέλεσης κάθε χρωμοσώματος για κάθε δραστηριότητα και απαιτούμενο πόρο (Κεφάλαιο 5.2.3.1). Όσον αφορά τη Λίστα Δραστηριοτήτων, για πρακτικούς λόγους, συμπληρώνεται από μια τυχαία λίστα όλων των δραστηριοτήτων του έργου, χωρίς να ικανοποιεί τις σχέσεις προτεραιότητας, καθώς η ικανοποίησή τους θα πραγματοποιηθεί κατά την υλοποίηση του εσωτερικού γενετικού αλγορίθμου, όπου η λίστα του χρωμοσώματος θα λάβει και τις τελικές προγραμματισμένες δραστηριότητες. Κατά το πέρας της δημιουργίας όλων των χρωμοσωμάτων που απαιτεί ο αρχικός πληθυσμός, λαμβάνουν χώρα οι εξελικτικές διαδικασίες της διασταύρωσης και μετάλλαξης για τη δημιουργία του συνολικού πληθυσμού της γενιάς που εκπροσωπεί, μεγέθους $2*POP_{ext}$.

5.2.3.3 Τελεστής Διασταύρωσης

Με δεδομένο τον αρχικό πληθυσμό των γενικών χρωμοσωμάτων και με στόχο την παραγωγή απογόνων της υφιστάμενης γενιάς, εφαρμόζεται ο τελεστής διασταύρωσης. Ο τελεστής διασταύρωσης έγκειται στην τυχαία επιλογή ζευγών γενικών χρωμοσωμάτων του αρχικού πληθυσμού και έπειτα στη γέννηση δύο νέων χρωμοσωμάτων-απόγονων για κάθε ζεύγος, με μετάβαση των γενετικών χαρακτηριστικών των γονέων στους απόγονούς τους (Hartmann, 1998).

Η διασταύρωση που υλοποιείται στην εξελικτική διαδικασία του εξωτερικού γενετικού αλγορίθμου, βασίζεται στους Πίνακες Διαμερίσεων και αποτελεί διασταύρωση ενός σημείου (*1-point crossover*). Αβίαστα, τα γενετικά χαρακτηριστικά που μεταφέρονται από κάθε ζεύγος γονέων στους προκύπτοντες αντίστοιχους απογόνους τους, αφορούν στους συνολικούς αριθμούς διαμερίσεων ανά δραστηριότητα και πόρο που είναι κωδικοποιημένοι στους Πίνακες Διαμερίσεων. Ειδικότερα, για κάθε τυχαίο ζεύγος γενικών χρωμοσωμάτων του αρχικού πληθυσμού, έστω *X* (μητέρα) και *Y* (πατέρας) να αποτελούν τους γονείς, επιλέγεται τυχαία ένα ακέραιος θετικός αριθμός q εντός του άνω φραγμένου συνόλου που ορίζει ο συνολικός αριθμός δραστηριοτήτων του έργου, όπου $0 < q < jobNumber$. Στη συνέχεια, παράγονται δύο νέα χρωμοσώματα, έστω *XY* (κόρη) και *YX* (υιός) να αποτελούν τους απόγονους, με τους Πίνακες Διαμερίσεων των οποίων να λαμβάνουν τιμές από τους αντίστοιχους των γονέων τους, όπως παρουσιάζεται στο Σχήμα 19.

Ο Πινακας Διαμερίσεων του *XY* (κόρη) χρωμοσώματος, λαμβάνει για τις στήλες $1, \dots, q$ τις τιμές των αντίστοιχων στηλών του Πίνακα Διαμερίσεων του *X* (μητέρα), ενώ για τις στήλες $q+1, \dots, jobNumber$, από τις τιμές των αντίστοιχων στηλών του Πίνακα Διαμερίσεων του *Y* (πατέρα). Εφαρμόζοντας τη διαδικασία της διασταύρωσης για όλα τα τυχαία ζέυγη του αρχικού πληθυσμού της γενιάς μεγέθους POP_{ext}, παράγονται POP_{ext} νέα χρωμοσώματα με υπολογισμένους τους Πίνακες Διαμερίσεων που αποτελούν τον πληθυσμό των απογόνων της γενιάς



Σχήμα 19. Τελεστής διασταύρωσης εξωτερικού γενετικού αλγορίθμου.

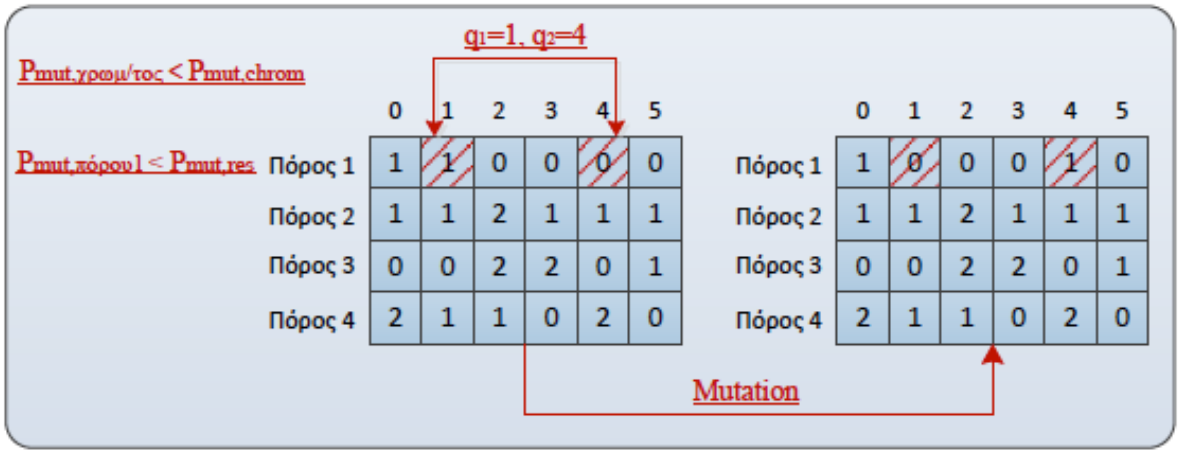
5.2.3.4 Τελεστής Μετάλλαξης

Ο τελεστής μετάλλαξης λαμβάνει χώρα με στόχο την ανανέωση και τη διαφοροποίηση του πληθυσμού κάθε γενιάς με τρόπους που μέσω του τελεστή διασταύρωσης δεν δύναται να πραγματοποιηθούν. Το γεγονός της ανανεώσης του πληθυσμού της κάθε γενιάς του εξωτερικού γενετικού αλγορίθμου αποτελεί υψίστης σημασίας για την αποτελεσματικότητα του αλγορίθμου "*do4u*", καθώς μέσα από αυτήν τη διαδικασία δημιουργούνται νέα γενικά χρωμοσώματα με διαφορετικούς Πίνακες Διαμερίσεων και κατ'επέκταση διαφορετικών Προφίλ Εκτέλεσης που είναι και το ζητούμενο του εξωτερικού γενετικού αλγορίθμου.

Η μετάλλαξη που υλοποιείται στην εξελικτική διαδικασία του εξωτερικού γενετικού αλγορίθμου, βασίζεται στους Πίνακες Διαμερίσεων και εφαρμόζεται στον πληθυσμό των απογόνων που προκύπτουν με την ολοκλήρωση του τελεστή διασταύρωσης (Κεφάλαιο 5.2.3.3). Έγκειται στην τυχαία αλλαγή της τιμής του συνολικού αριθμού διαμερίσεων του Πίνακα Διαμερίσεων δύο δραστηριοτήτων ανά πόρο, με δεδομένους πιθανοτικούς περιορισμούς που πρέπει να ικανοποιούνται για να λάβει χώρα η μετάλλαξη ενός χρωμοσώματος και κάθε πόρου αυτού. Οι πιθανοτικοί περιορισμοί αφορούν την πιθανότητα μετάλλαξης ενός χρωμοσώματος, που ορίζεται ως $P_{mut,chrom}$, καθώς και την πιθανότητα μετάλλαξης κάθε σειράς του Πίνακα Διαμερίσεων. Δεδομένου του ότι κάθε σειρά αντιπροσωπεύει έναν πόρο στον Πίνακα Διαμερίσεων, η πιθανότητα μετάλλαξης κάθε σειράς ισοδυναμεί με την πιθανότητα μετάλλαξης κάθε πόρου και ορίζεται ως $P_{mut,res}$.

Πιο συγκεκριμένα, για κάθε ένα χρωμόσωμα του πληθυσμού των απογόνων δίδεται μια πιθανότητα, έστω $P_{mut,\chi\rho\omega\mu/\tau o\varsigma}$. Σε περίπτωση που ισχύει $P_{mut,\chi\rho\omega\mu/\tau o\varsigma} < P_{mut,chrom}$, τότε το χρωμόσωμα θα μεταλλαχθεί, ενώ σε αντίθετη περίπτωση θα παραμείνει ως έχει. Έπειτα, δίδεται μια πιθανότητα σε κάθε ένα πόρο του έργου $j \in R$ και σε περίπτωση που

ισχύει $P_{mut,\pi\acute{o}\rho ου\ j} < P_{mut,res}$, τότε θα πραγματοποιηθεί η μετάλλαξη της σειράς που αντιπροσωπεύει ο πόρος j στον Πίνακα Διαμερίσεων του χρωμοσώματος. Ενδέχεται για κανέναν πόρο να μην ικανοποιείται η συνθήκη που θα επιτρέψει τη μετάλλαξη του χρωμοσώματος με αποτέλεσμα της παραμονής του Πίνακα Διαμερίσεων ως έχει. Για κάθε πόρο j, όπου $P_{mut,\pi\acute{o}\rho ου\ j} < P_{mut,res}$, πραγματοποιείται η διαδικασία της αλλαγής θέσης στην τιμή του Πίνακα Διαμερίσεων δύο τυχαίων δραστηριοτήτων που αφορούν τον πόρο j, όπως παρουσιάζεται και στο Σχήμα 20. Πιο συγκεκριμένα, λαμβάνονται δύο τυχαίοι αριθμοί q_1, q_2,



Σχήμα 20. Τελεστής μετάλλαξης εξωτερικού γενετικού αλγορίθμου

$1 \leq q_1 \neq q_2 \leq jobNumber$ και οι θέσεις $[j, q_2]$ και $[j, q_1]$ στον Πίνακα Διαμερίσεων ανταλλάσσουν τιμές συνολικού αριθμού διαμερίσεων.

Με το πέρας της διαδικασίας του τελεστή μετάλλαξης για κάθε χρωμόσωμα του πληθυσμού απογόνων με υπολογισμένους τους τελικούς Πίνακες Διαμέρισης, λαμβάνει χώρα η διαδικασία υπολογισμού των Προφίλ Εκτέλεσης κάθε γενικού χρωμοσώματος του πληθυσμού των απογόνων (Κεφάλαιο 5.2.3.1). Ο συνολικός πληθυσμός της κάθε γενιάς μεγέθους $2*POP_{ext}$, συντίθεται από τα χρωμοσώματα τόσο του αρχικού πληθυσμού όσο και του πληθυσμού των απογόνων, όλα με υπολογισμένα Προφίλ Εκτέλεσης.

5.2.3.5 Αξιολόγηση & Επιλογή

Για κάθε ένα γενικό χρωμόσωμα του συνολικού πληθυσμού κάθε γενιάς του εξωτερικού γενετικού αλγορίθμου, καλείται ο εσωτερικός γενετικός αλγόριθμος, ο οποίος επιστρέφει το βέλτιστο χρονοπρόγραμμα των δραστηριοτήτων στο πεδίο της Λίστας Δραστηριοτήτων του χρωμοσώματος και αποδίδει την τιμή της αντικειμενικής συνάρτησης (*fitness value*) της συνολικής διάρκειας του έργου, που αφορά τα χαρακτηριστικά του Προφίλ Εκτέλεσης και του χρονοπρογράμματος, κάθε γενικού χρωμοσώματος. Η επιλογή των POP_{ext} γενικών χρωμοσωμάτων του συνολικού πληθυσμού μεγέθους $2*POP_{ext}$, που θα αποτελέσουν τον αρχικό πληθυσμό της επόμενης γενιάς του εξωτερικού γενετικού αλγορίθμου, γίνεται βάσει της τιμής της αντικειμενικής συνάρτησης κάθε γενικού χρωμοσώματος (*Ranking Method*) (Hartmann, 1998). Επιλέγονται, λοιπόν, τα POP_{ext} χρωμοσώματα με την μικρότερη τιμή αντικειμενικής συνάρτησης για να αποτελέσουν εφαλτήριο της εξέλιξης της επόμενης γενιάς, δεδομένου του αντικειμενικού στόχου του προβλήματος της ελαχιστοποίησης της συνολικής διάρκειας του έργου.

5.2.4 Εσωτερικός Γενετικός Αλγόριθμος

Ο εσωτερικός γενετικός αλγόριθμος, ο ψευδοκώδικας της διαδικασίας επίλυσης του οποίου παρουσιάζεται στον Αλγόριθμο 8, διαχειρίζεται την αναζήτηση του βέλτιστου χρονοπρογράμματος στο χώρο λύσεων του προβλήματος και κατ' επέκταση τη βελτιστοποίηση της συνολικής διάρκειας του έργου μέσα από το χρονοπρογραμματισμό των δραστηριοτήτων κάθε ενός γενικού χρωμοσώματος, με δεδομένα προφίλ εκτέλεσης της κάθε δραστηριότητας.

Πιο συγκεκριμένα, με εισόδους τα δεδομένα ενός γενικού χρωμοσώματος, δημιουργεί έναν αρχικό πληθυσμό από γενότυπους (γίνεται η χρήση του όρου γενότυπου εν αποφυγή σύγχυσης μεταξύ αυτού και του γενικού χρωμοσώματος), κάθε ένας από τους οποίους αποτελείται από μια τυχαία λίστα δραστηριοτήτων με σεβασμό στις σχέσεις προτεραιότητας των δραστηριοτήτων του έργου. Στη συνεχεία, λαμβάνουν χώρα οι εξελικτικές διαδικασίες των τελεστών διασταύρωσης και μετάλλαξης μεταξύ των γενότυπων του αρχικού πληθυσμού, συνθέτοντας ένα συνολικό πληθυσμό γενότυπων. Για κάθε γενότυπο υλοποιείται η ευρετική μέθοδος της σειριακής παραγωγής χρονοπρογραμμάτων για την αποκωδικοποίηση και τη δημιουργία ενός εφικτού χρονοπρογράμματος με συνεκτίμηση της συνολικής διάρκειας του έργου. Οι γενότυποι με την καλύτερη τιμή αντικειμενικής συνάρτησης συνολικής διάρκειας του έργου, επιλέγονται για τον αρχικό πληθυσμό της επόμενης γενιάς λύσεων. Κατά το πέρας του εσωτερικού γενετικού αλγορίθμου, η χρονικά προγραμματισμένη λίστα δραστηριοτήτων του καλύτερου γενότυπου, αποτελεί τη λίστα δραστηριοτήτων του γενικού χρωμοσώματος που κάλεσε τον εσωτερικό γενετικό αλγόριθμο.

Αλγόριθμος 8. Ψευδοκώδικας GA-Internal

Input : chromoExternal, ResourceAvailabilities, jobNumber, resourceNumber, timeHorizon
Output : fitnessValue, chromoExternal.actsList
`Set` populationSizeInternal = POP_{int};
`Set` evolutionStepsInternal = e;
`Set` probabilityOfMutationInternal= $p_{mut,int}$;
`Set` generationCounter $g = 0$;
`//Generate initial population` Q_0 `of` POP_{int} `individuals;`
for $i = 0 \dots POP_{int}$ **do**
    $Q_0 = Q_0 \cup$ `GenerateRandomPrecedenceFeasibleActivityList;`
end
$Q_g = Q_0$;
while *stopping criteria not met* **do**
    `//Generate offspring population` $Q_{g,children}$
    $Q_{g,children} =$ `Crossover` Q_g;
    $Q_{g,children} =$ `Mutate` Q_g;
    `//Combine current and offspring population`
    $W_g = Q_g \cup Q_{g,children}$;
    `//Compute fitness value of each individual through sSSGS-U`
    for $i = 0 \dots 2 * POP_{int}$ **do**
        $W_g[i].fitnessValue = \boldsymbol{SSGS - U}($ $W_g[i]$, chromoExternal, ResourceAvailabilities, jobNumber, resourceNumber);
    end
    `/*ELITISIM: for each individual with duplicates generate new individual*/`

while *stopping criteria not met* **do**
Generate new $individual$= RandomPrecedenceFeasibleActivityList;
$individual.fitnessValue = \boldsymbol{SSGS - U}$(individual, chromoExternal, ResourceAvailabilities, jobNumber, resourceNumber)
$W_g = W_g \cup individual$;
end
//Select POP_{int} individuals with best fitnnesValues for next //generation
Q_g = RankingMethod(W_g);
$g = g + 1$;
end
//Select individual with best fitness value and put it to //chromoExternal.actsList
$bestIndividual = Q_g$[i] | min { $Q_g[i].fitnessValue$, $\forall i = 0, \dots, POP_{int}$};
$chromoExternal.actsList = bestIndividual$;
$fitnessValue = bestIndividual.fitnessValue$;
return (fitnessValue**)**;

5.2.4.1 Αρχικός πληθυσμός

Ο αρχικός πληθυσμός του εσωτερικού γενετικού αλγορίθμου αποτελείται από γενότυπους μεγέθους ίσου με τον επιλεγμένο αριθμό πληθυσμού (*population size internal chromosomes – POP_{int}*). Κάθε γενότυπος του αρχικού πληθυσμού αποτελεί μια τυχαία λίστα δραστηριοτήτων, μεγέθους ίση με το σύνολο των δραστηριοτήτων του έργου, που ικανοποιεί τις υφιστάμενες γενικευμένες σχέσεις προτεραιότητας.

Κάθε γενότυπος του αρχικού πληθυσμού παράγεται επαναλαμβάνοντας την ακόλουθη διαδικασία: η επόμενη δραστηριότητα που τοποθετείται στη λίστα δραστηριοτήτων επιλέγεται τυχαία από το σύνολο των, ως τώρα, μη τοποθετημένων δραστηριοτήτων, όλες οι προαπαιτούμενες δραστηριότητες των οποίων έχουν ήδη επιλεχθεί και τοποθετηθεί στη λίστα. Ειδικότερα, στη λίστα δραστηριοτήτων κάθε γενότυπου, η πρώτη και η τελευταία θέση αντιπροσωπεύει τη βοηθητική δραστηριότητα αρχής και τέλους, αντίστοιχα. Στη συνέχεια, κάθε διαδοχική θέση, ύστερα της πρώτης, καταλαμβάνεται μέσα από την άνωθι επαναληπτική διαδικασία.

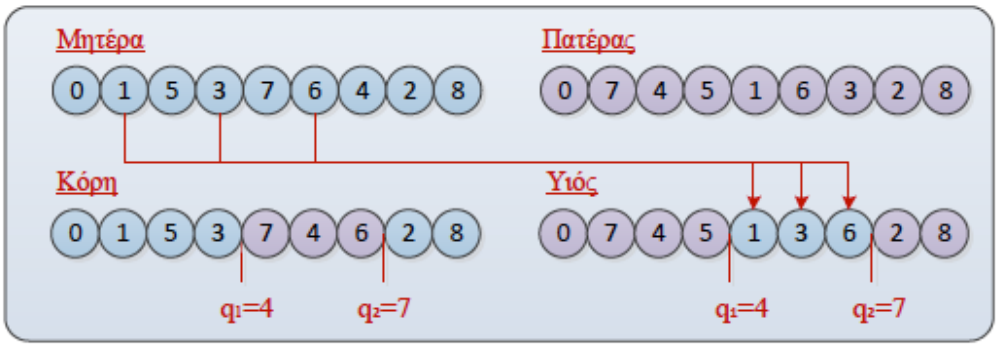
Κατά το πέρας της δημιουργίας όλων των γενότυπων που απαιτεί ο αρχικός πληθυσμός, υλοποιούνται οι εξελικτικές διαδικασίες της διασταύρωσης και μετάλλαξης για τη δημιουργία του συνολικού πληθυσμού της γενιάς που εκπροσωπεί, μεγέθους *2*POP_{iint}*.

5.2.4.2 Τελεστής Διασταύρωσης

Με δεδομένο τον αρχικό πληθυσμό των γενότυπων και με στόχο την παραγωγή απογόνων της υφιστάμενης γενιάς, εφαρμόζεται ο τελεστής διασταύρωσης. Ο τελεστής διασταύρωσης έγκειται στην τυχαία επιλογή ζευγών γενότυπων του αρχικού πληθυσμού και μετέπειτα στη γέννηση δύο νέων γενότυπων-απόγονων για κάθε ζεύγος, με μετάβαση των γενετικών χαρακτηριστικών των γονέων στους απόγονούς τους (Hartmann, 1998). Η διασταύρωση που υλοποιείται κατά την εξελικτική διαδικασία του εσωτερικού γενετικού αλγορίθμου, αποτελεί διασταύρωση δύο σημείων (*2-point crossover*), η επιλογή της οποίας στηρίζεται στα πειραματικά αποτελέσματα του Hartmann (Hartmann, 1998). Σημειώνεται ότι παρότι υπάρχουν γενικευμένες σχέσεις προτεραιότητας, οι απόγονοι που παράγονται από κάθε ζεύγος γενότυπων του αρχικού πληθυσμού μέσα από τον τελεστή διασταύρωσης, αποτελούν λίστες δραστηριοτήτων που ικανοποιούν τις υφιστάμενες σχέσεις προτεραιότητας,

δεδομένου του ότι οι γονείς αποτελούν αντίστοιχες λίστες δραστηριοτήτων (Neumann-Braun et al., 2003).

Ειδικότερα, για κάθε τυχαίο ζεύγος γενότυπων του αρχικού πληθυσμού, έστω X (μητέρα) και Y (πατέρας) να αποτελούν τους γονείς, επιλέγονται τυχαία δύο ακέραιοι θετικοί αριθμοί q_1 και q_2 εντός του άνω φραγμένου συνόλου που ορίζει ο συνολικός αριθμός δραστηριοτήτων του έργου, όπου $0 < q_1 < q_2 < jobNumber$. Έπειτα, παράγονται δύο νέα χρωμοσώματα, έστω XY (κόρη) και YX (υιός) να αποτελούν τους απόγονους, όπως παρουσιάζεται στο Σχήμα 21.



Σχήμα 21. Τελεστής διασταύρωσης εσωτερικού γενετικού αλγορίθμου

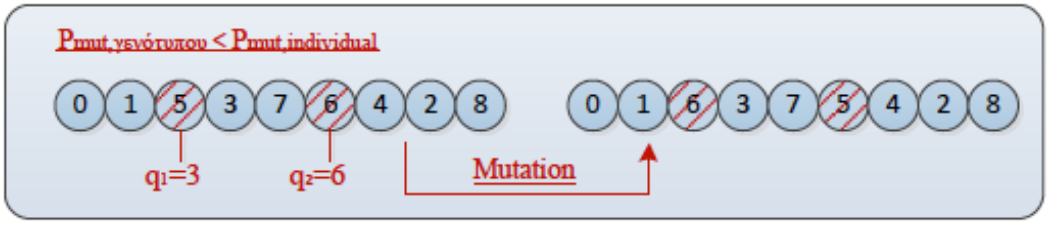
Ο γενότυπος XY (κόρη), λαμβάνει για τις θέσεις $1, \dots, q_1$ τις πρώτες q_1 δραστηριότητες από το γενότυπο X (μητέρα), για τις θέσεις $q_1 + 1, \dots, q_2$ τις δραστηριότητες από το γενότυπο Y (πατέρα), και για τις παραμένουσες $q_2 + 1, \dots, jobNumber$ θέσεις, τις δραστηριότητες από το γενότυπο X (μητέρα). Σημειώνεται ότι για τις θέσεις $q_1 + 1, \dots, q_2$ του γενότυπου XY (κόρη), επιλέγονται $q_2 - q_1$ δραστηριότητες σταδιακά από τις θέσεις $1, \dots, q_2$ του γενότυπου Y (πατέρα) που δεν βρίσκονται στη λίστα του XY, ενώ για τις παραμένουσες $q_2 + 1, \dots, jobNumber$ θέσεις του XY (κόρη), επιλέγονται $jobNumber - q_2$ δραστηριότητες σταδιακά από τις θέσεις $1, \dots, jobNumber$ του γενότυπου X (μητέρα), που δεν βρίσκονται στη λίστα του XY. Εφαρμόζοντας τη διαδικασία της διασταύρωσης για όλα τα τυχαία ζεύγη του αρχικού πληθυσμού της γενιάς μεγέθους POP_{int}, παράγονται POP_{int} νέοι γενότυποι που αποτελούν τον πληθυσμό των απογόνων της γενιάς

5.2.4.3 Τελεστής Μετάλλαξης

Ο τελεστής μετάλλαξης πραγματοποιείται με στόχο την ανανέωση και τη διαφοροποίηση του πληθυσμού κάθε γενιάς με τρόπους που ο τελεστής διασταύρωσης δεν μπορεί να πραγματοποιήσει. Η μετάλλαξη που υλοποιείται στην εξελικτική διαδικασία του εσωτερικού γενετικού αλγορίθμου εφαρμόζεται στον πληθυσμό των απογόνων που προκύπτουν με την ολοκλήρωση του τελεστή διασταύρωσης (Κεφάλαιο 5.2.4.2). Έγκειται, σε τυχαία αλλαγή των θέσεων που καταλαμβάνουν δύο δραστηριότητες στη λίστα δραστηριοτήτων του γενότυπου, με δεδομένους πιθανοτικούς περιορισμούς που πρέπει να ικανοποιούνται για να λάβει χώρα η μετάλλαξη. Οι πιθανοτικοί περιορισμοί αφορούν την πιθανότητα μετάλλαξης ενός γενότυπου, που ορίζεται ως $P_{mut,individual}$.

Ειδικότερα, για κάθε ένα γενότυπο του πληθυσμού των απογόνων δίδεται μια πιθανότητα, έστω $P_{mut,γενότυπου}$. Σε περίπτωση που ισχύει $P_{mut,γενότυπου} < P_{mut,individual}$, τότε ο γενότυπος θα μεταλλαχθεί όπως παρουσιάζεται στο Σχήμα 22, ενώ σε αντίθετη περίπτωση θα παραμείνει ως έχει. Πιο συγκεκριμένα, λαμβάνονται δύο τυχαίοι αριθμοί

q_1, q_2, με $1 \leq q_1 \neq q_2 \leq jobNumber$ και οι δραστηριότητες στις θέσεις q_2 και q_1 ανταλλάσσουν θέσεις. Εν συνεχεία, η λίστα δραστηριοτήτων του γενότυπου ελέγχεται για ικανοποίηση των σχέσεων προτεραιότητας, και σε περίπτωση προσβολής τους ο γενότυπος επανέρχεται στην αρχική μορφή του. Η διαδικασία μετάλλαξης επαναλαμβάνεται με επιλογή νέων δραστηριοτήτων για αλλαγή θέσεων, μέχρις ότου ο προκύπτων γενότυπος ικανοποιεί τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων.



Σχήμα 22. Τελεστής μετάλλαξης εσωτερικού γενετικού αλγορίθμου

Με το πέρας της διαδικασίας του τελεστή μετάλλαξης, προκύπτει ο συνολικός πληθυσμός της κάθε γενιάς μεγέθους $2*POP_{int}$, ο οποίος συντίθεται από τους γενότυπους τόσο του αρχικού πληθυσμού όσο και του αντίστοιχου των απογόνων. Για κάθε γενότυπο του συνολικού πληθυσμού, ακολουθεί η σειριακή μέθοδος παραγωγής χρονοπρογραμμάτων και στη συνέχεια η επιλογή των γενότυπων που θα αποτελέσουν τον αρχικό πληθυσμό της επόμενης γενιάς.

5.2.4.4 Αξιολόγηση & Επιλογή

Για κάθε ένα γενότυπο του συνολικού πληθυσμού κάθε γενιάς του εσωτερικού γενετικού αλγορίθμου, καλείται η σειριακή μέθοδος παραγωγής χρονοπρογραμμάτων, η οποία επιστρέφει ένα εφικτό χρονοπρόγραμμα και αποδίδει την τιμή της αντικειμενικής συνάρτησης (*fitness value*) της συνολικής διάρκειας του έργου βάσει του παραγόμενου χρονοπρογράμματος. Η επιλογή των POP_{int} γενικών χρωμοσωμάτων του συνολικού πληθυσμού μεγέθους $2*POP_{int}$, που θα αποτελέσουν τον αρχικό πληθυσμό της επόμενης γενιάς του εσωτερικού γενετικού αλγορίθμου, γίνεται βάσει της τιμής της αντικειμενικής συνάρτησης κάθε γενικού χρωμοσώματος (*Ranking Method*) (Hartmann, 1998). Επιλέγονται, λοιπόν, τα POP_{int} χρωμοσώματα με την μικρότερη τιμή αντικειμενικής συνάρτησης για να αποτελέσουν τον αρχικό πληθυσμό της επόμενης γενιάς.

5.2.5 Σειριακή Μέθοδος Παραγωγής Χρονοπρογραμμάτων με βήμα αποπρογραμματισμού –sSGSU

Οι μέθοδοι παραγωγής προγραμμάτων (*Schedule Generation Schemes- SGS*) αποτελούν τον πυρήνα της πλειοψηφίας των ευρετικών μεθόδων επίλυσης προβλημάτων προγραμματισμού έργων, καθώς καθίστανται διαισθητικές, εύκολες στη χρήση και γρήγορες από τη σκοπιά της υπολογιστικής προσπάθειας (Kolisch, 1996b). Δυο διαφορετικοί αλγόριθμοι SGS είναι διαθέσιμοι. Ο σειριακός αλγόριθμος παραγωγής προγραμμάτων (*serial Schedule Generation Schemes- sSGS*), ο οποίος παρουσιάζει προσαύξηση δραστηριοτήτων (*activity-incrementation*), και ο παράλληλος αλγόριθμος παραγωγής προγραμμάτων (*parallel Schedule Generation Schemes- pSGS*), ο οποίος παρουσιάζει προσαύξηση χρόνου (*time-incrementation*). Και οι δύο αλγόριθμοι παράγουν ένα εφικτό χρονοπρόγραμμα μέσα από την

επέκταση ενός μερικού προγράμματος (*partial schedule*) σε διαδοχικά στάδια. Σε κάθε στάδιο, ο SGS παράγει το σύνολο των δραστηριοτήτων που μπορούν να εκτελεστούν, δεδομένων των περιορισμών πόρων και των σχέσεων προτεραιότητας. Έπειτα, ένας συγκεκριμένος κανόνας προτεραιότητας εφαρμόζεται με στόχο την επιλογή της δραστηριότητας ή των δραστηριοτήτων που θα προγραμματιστούν για εκτέλεση.

Ο προτεινόμενος αλγόριθμος επίλυσης "*do4u*", για την παραγωγή εφικτών χρονοπρογραμμάτων για κάθε ένα γενότυπο του εσωτερικού γενετικού αλγορίθμου, κάνει χρήση της μεθόδου σειριακής παραγωγής χρονοπρογραμμάτων, για την αποφυγή ενδεχόμενης απόκλισης μιας βέλτιστης λύσης από την παράλληλη μέθοδο παραγωγής χρονοπρογραμμάτων (Kolisch, 1996b, Sprecher et al., 1995). Δεδομένου του ότι το υπό μελέτη πρόβλημα διαχειρίζεται γενικευμένες σχέσεις προτεραιότητας με ελάχιστες και μέγιστες χρονικές καθυστερήσεις μεταξύ των δραστηριοτήτων του έργου, ο "*do4u*" υλοποιεί μια παραλλαγή του τυπικού sSGS αλγορίθμου προσθέτοντας μία διαδικασία αποπρογραμματισμού, σε στάδια που καμία δραστηριότητα δεν μπορεί να προγραμματιστεί χωρίς να παραβιάσει είτε περιορισμούς διαθεσιμότητας πόρων είτε περιορισμούς που πηγάζουν από τις χρονικές καθυστερήσεις νωρίτερης και αργότερης δυνατής έναρξης των σχέσεων προτεραιότητας των δραστηριοτήτων. Η παραλλαγή του τυπικού sSGS, που υλοποιείται είναι η μέθοδος της σειριακής παραγωγής χρονοπρογραμμάτων με βήμα αποπρογραμματισμού (*serial Schedule Generation Scheme with Unscheduling –sSGSU*) (Neumann-Braun et al., 2003), ο ψευδοκώδικας της οποίας παρουσιάζεται στον Αλγόριθμο 9.

Κατά την κλίση του ο sSGSU από κάθε γενότυπο του εσωτερικού γενετικού αλγορίθμου, λαμβάνει ως εισόδους τη λίστα δραστηριοτήτων με σεβασμό στις σχέσεις προτεραιότητας, το γενικό χρωμόσωμα του εξωτερικού γενετικού αλγορίθμου με τα προφίλ εκτέλεσης και τη συνολική διάρκεια της κάθε δραστηριότητας, τις διαθεσιμότητες των πόρων του έργου, καθώς και το συνολικό αριθμό δραστηριοτήτων και πόρων του έργου. Η διαδικασία υπολογισμού ενός εφικτού χρονοπρογράμματος μέσω του sSGSU, αποτελεί μία επαναληπτική διαδικασία, σε κάθε βήμα της οποίας υπολογίζεται το σύνολο των επιλέξιμων δραστηριοτήτων που δεν έχουν προγραμματιστεί ως τώρα και μπορούν να προγραμματιστούν αναφορικά με τις προαπαιτούμενες δραστηριότητές τους. Από το σύνολο των επιλέξιμων δραστηριοτήτων λαμβάνεται εκείνη με την υψηλότερη προτεραιότητα που της δίδεται από τη θέση που κατέχει στη δεδομένη λίστα δραστηριοτήτων, δηλαδή εκείνη που βρίσκεται πιο κοντά στην βοηθητική δραστηριότητα αρχής. Για την επιλεγμένη, πλέον, δραστηριότητα εκτιμάται η νωρίτερη δυνατή χρονική στιγμή έναρξης, αναφορικά με τις διαθέσιμες ποσότητες των πόρων και τις χρησιμοποιούμενες, από αυτή ποσότητές τους, σύμφωνα με το δεδομένο προφίλ εκτέλεσης, για κάθε χρονική στιγμή ως την ολοκλήρωσή της. Σε περίπτωση που η προκύπτουσα χρονική στιγμή έναρξης της επιλεγμένης δραστηριότητας υπερβαίνει την αργότερη δυνατή έναρξη αυτής, εφαρμόζεται το βήμα αποπρογραμματισμού. Σε αντίθετη περίπτωση, η δραστηριότητα προγραμματίζεται στην προκύπτουσα χρονική στιγμή έναρξης και οι νωρίτερες και αργότερες στιγμές έναρξης όλων των μη προγραμματισμένων δραστηριοτήτων ανανεώνονται αντίστοιχα (Αλγόριθμος 9). Για την αποφυγή ατέρμονων επαναλήψεων καθορίζεται ένα μέγιστος αριθμός βημάτων αποπρογραμματισμού. Σε περιπτώσεις που διακοπεί η διαδικασία υπολογισμού του sSGSU και δεν βρεθεί ένα εφικτό χρονοπρόγραμμα δεν τερματίζεται όλη υπολογιστική διαδικασία του "*do4u*". Αντίθετα, επιστρέφεται μία πρακτικά άπειρη τιμή διάρκειας έργου που στις μελλοντικές μεθόδους επιλογής χρωμοσωμάτων θα απορριφθεί.

Κατά την υλοποίηση του βήματος του αποπρογραμματισμού, επιλέγονται και αποπρογραμματίζονται όλες οι προγραμματισμένες δραστηριότητες που σχετίζονται με την επιλεγμένη δραστηριότητα και που επηρεάζουν την τιμή της αργότερης δυνατής έναρξής της.

Το γεγονός αυτό λαμβάνει χώρα, ώστε να αποπρογραμματιστούν εκείνες οι δραστηριότητες στις οποίες οφείλεται η μικρή τιμή της αργότερης δυνατής έναρξης της επιλεγμένης δραστηριότητας σε σχέση με την προκύπτουσα χρονική στιγμή έναρξης αυτής και που οδήγησαν στο βήμα του αποπρογραμματισμού. Για την αποφυγή μελλοντικής επανάληψης της υφιστάμενης κατάστασης, μετακινείται χρονικά η νωρίτερη δυνατή έναρξη όλων των υπό αποπρογραμματισμό δραστηριοτήτων, κατά ένα χρονικό διάστημα ίσο με τη διαφορά της προκύπτουσας χρονικής στιγμής έναρξης της επιλεγμένης δραστηριότητας με την αργότερη δυνατή έναρξη αυτής. Παράλληλα, αποπρογραμματίζονται όλες οι οι υπόλοιπες προγραμματισμένες δραστηριότητες, που λόγω της χρονικής μεταβολής των προηγούμενων αποπρογραμματισμένων δραστηριοτήτων, μπορούν να προγραμματιστούν σε νωρίτερη χρονική στιγμή. Ύστερα, ανανεώνονται οι νωρίτερες και αργότερες στιγμές έναρξης όλων των μη προγραμματισμένων δραστηριοτήτων, όπως υποδηλώνεται στον ψευδοκώδικα του sSGSU (Αλγόριθμος 9). Τέλος, σε περιπτώσεις που δεν δύναται να αποπρογραμματιστεί καμία δραστηριότητα, ο sSGSU τερματίζει την επαναληπτική διαδικασία και επιστρέφει μια πρακτικά άπειρη τιμή διάρκειας έργου.

Αλγόριθμος 9. Ψευδοκώδικας σειριακής Μεθόδου Παραγωγής Χρονοπρογραμμάτων με Αποπρογραμματισμό- sSGS-U()

Input : activityList, chromoExt, ResourceAvailabilities, jobNumber, resourceNumber,
Output : fitnessValue
Set $V = \{activityList\}$;
Set $C = \{i \in V \mid i\ completed\}$;
Set $E = \{i \in V \mid i\ eligible\}$;
Set $S_i = start\ time\ of\ activity\ i$;
Set $A = \{i \in C \mid S_i \leq t \leq S_i + d_{i,total}\}$;
Set $d_{i^*i} = longest\ path\ from\ i^*\ to\ i$;
$S_0 = 0$;
$C = \{0\}$;
$u = 0$;
/*Compute ES, LS and d_{i^*i} of each activity through Floyd-Warshall calculations*/
//Calculate direct predecessors
for $i = 0 \ldots jobNumber - 1$ **do**
  for $j = 0 \ldots jobNumber - 1$ **do**
    if $i \neq j$ && $distMatrix[i][j] \geq 0 || distMatrix[i][j] < 0 \&\& distMatrix[i][j] -1000000000$ **then**
      Set $j\ as\ predecessor\ of\ i$;
    end
  end
end
//Generate feasible schedule
while $C \neq V$ **do**
  Select $(j^* \in E)$;
  $t^* = \min \left\{ t \geq ES_{j^*} \mid \text{chromoExt.profile}_{i,k}(S^C, T) + \text{chromoExt.profile}_{j^*,k} \leq R_k, \forall t \leq \tau \leq t + j^*.duration_k, \forall k \in R \right\}$,
  if $t^* \geq LS_{j^*}$ **then**
    //Unschedule step
    Set $U = \{i \in C \mid LS_{j^*} = S_{i^*} + d_{j^*i}\}$;
    //No feasible schedule is found
    if $0 \in U || u > u_{max}$ **then**
      exit();

```
            end
            //Right-shift of activities i ∈ U
            for all i ∈ U do
                ES_i = S_i + t* − LS_j*;
                C = C\{i};
                //No feasible schedule is found
                if ES_i > −d_i0 then
                    exit();
                end
            end
            //Unschedule all activities i with S_i > min_{h∈U} S_h
            for all i ∈ C with S_i > min_{h∈U} S_h do
                C = C\{i};
            end
            //Compute ES_j, LS_j for all j ∈ V\C
            for all j ∈ V do
                ES_j = max{ d_0j, max_{h∈U}(ES_j + d_hj)};
                LS_j = −d_j0;
                for all j ∈ C do
                    ES_j = max{ ES_j, S_i + d_ij};
                    LS_j = min{ LS_j, S_i − d_ji};
                end
            end

            u = u + 1;
        else
            //Schedule j* at time t*
            S_j* = t*;
            C = C ∪ j*;
            //Update ES_j*, LS_j*;
            for all j ∈ V\C do
                ES_j = max{ ES_j, S_j* + d_j*j};
                LS_j = min{ LS_j, S_j* − d_jj*};
            end
        end
    end
    /*copy schedule set C with S_i, ∀i ∈ V, in the inputed activityList in
      order to return the schedule with the fitness value*/
    activityList = C;
    fitnessValue = S_dummy,end;
    return(fitnessValue);
```

5.2.5.1 Υπολογισμοί Floyd Warshall

Πριν την έναρξη της επαναληπτικής διαδικασίας υπολογισμού ενός εφικτού χρονοπρογράμματος μέσω του sSGSU, πρέπει να πραγματοποιηθούν οι υπολογισμοί των νωρίτερων και αργότερων δυνατών ενάρξεων των δραστηριοτήτων (*Early Start and Late Start times – ES and LS*). Δεδομένων των γενικευμένων σχέσεων προτεραιότητας, οι κλασικοί τρόποι υπολογισμού των ES και LS μέσω διακασιών όπως το CPM, δεν μπορούν να χρησιμοποιηθούν. Η ύπαρξη ελάχιστων και μέγιστων χρονικών καθυστερήσεων (*minimal and maximal time lags*) οδηγεί σε δίκτυα έργων με κύκλους, όπου κατ'ελάχιστο ένα τόξο σε κάθε κύκλο αντιπροσωπεύει μία μέγιστη χρονική καθυστέρηση. Κύκλοι θετικού μήκους δεν έχουν

φυσικό νόημα, καθώς αντικατοπτρίζουν περιορισμούς όπως $s_i \geq s_i + l,\ l > 0$. Επομένως, πρέπει να χρησιμοποιηθεί μία τεχνική που να λαμβάνει υπόψιν όλες τις πιθανές διαδρομές μεταξύ των δραστηριοτήτων του έργου, όπως ο αλγόριθμος Floyd-Warshall (Bartusch et al., 1988), ο ψευδοκώδικας του οποίου παρουσιάζεται στον Αλγόριθμο 10.

Ο αλγόριθμος Floyd-Warshall υπολογίζει τις μέγιστες διαδρομές l_{ij} μεταξύ όλων των ζευγών $(i,j) \in V$ δραστηριοτήτων του έργου. Πριν πραγματοποιηθεί η έναρξη του αλγορίθμου, όλες οι γενικευμένες σχέσεις προτεραιότητας μετατρέπονται σε σχέσεις αρχής-αρχής με υστερήσεις (*lags*) (όπως παρουσιάστηκαν στα Κεφάλαια 4.1.1 & 2.2.1.1). Τα προκύπτοντα δεδομένα χρησιμοποιούνται από τον αλγόριθμο για τη σύνθεση του αρχικού Πίνακα Αποστάσεων (*Distance Matrix*) $distMatrix_{ij}^{(k=1)}, i, j = 0, \dots, n$, ως ακολούθως:

$$distMatrix_{ij}^{(k=1)} = \begin{cases} l_{ij} = lag_{ij}, & \forall (i,j) \mu \varepsilon\ \sigma \chi \acute{\varepsilon} \sigma \eta\ \pi \rho o \tau \varepsilon \rho \alpha \iota \acute{o} \tau \eta \tau \alpha \varsigma \\ 0, & i = j \\ -\infty, & \sigma \varepsilon\ \alpha \nu \tau \acute{\iota} \theta \varepsilon \tau \eta\ \pi \varepsilon \rho \acute{\iota} \pi \tau \omega \sigma \eta \end{cases}$$

Ο αρχικός Πίνακας Αποστάσεων υποδεικνύει τις άμεσες διαδρομές από κάθε κόμβο i σε κάθε κόμβο j. Στη συνεχεία, εφαρμόζεται μια επαναληπτική υπολογιστική διαδικασία του αλγορίθμου, όπου σε κάθε επανάληψη $k+1$, υπολογίζεται η μέγιστη διαδρομή από τον κόμβο i στον j μέσα από τον ενδιαμέσο κόμβο k και υπολογίζεται το μήκος της προαναφερθείσας διαδρομής. Σε περίπτωση που ο ενδιάμεσος κόμβος k δεν αποτελεί μέρος μιας διαδρομής από τον κόμβο i στον j, η μέγιστη διαδρομή έχει μήκος ίσο με $l_{ij}^{(k)}$ από τον Πίνακα Αποστάσεων της προηγούμενης επανάληψης. Σε αντίθετη περίπτωση, η μέγιστη διαδρομή υπολογίζεται ως $l_{ik}^{(k)} + l_{kj}^{(k)}$. Επομένως, σε κάθε επανάληψη $k+1$ το μήκος της μέγιστης διαδρομής από τον κόμβο i στον j, υπολογίζεται ως ακολούθως:

$$distMatrix_{ij}^{(k+1)} = \max\{l_{ij}^{(k)}, l_{ik}^{(k)} + l_{kj}^{(k)}\}$$

Η τελευταία επανάληψη $k = n+1$, οδηγεί στον τελικό Πίνακα Αποστάσεων $distMatrix_{ij}^{(n+1)}$, ο οποίος με τη σειρά του δίδει τις μέγιστες διαδρομές μεταξύ κάθε ζεύγους δραστηριοτήτων (i,j).

Αλγόριθμος 10. Υπολογισμοί Floyd-Warshall

Input : activityList, GenerilizedPrecedenceRelations,TimeLags, jobNumber
Output : ES[],LS[],distMatrix[][]
Set $lag_{i,j}$=Time Lag between activities i and j;
//Compute initial distance matrix
for $c = 0 \dots jobNumber - 1$ **do**
    for $l = 0 \dots jobNumber - 1$ **do**
        if $c == l$ **then**
            $distMatrix[c][l] == 0$;
        else if *activity l is successor of activity c* **then**
            $distMatrix[c][l] == lag_{i,j}$;
        else
            $distMatrix[c][l] == -1000000000$;
        end
    end
end
for $k = 0 \dots jobNumber - 1$ **do**
    for $i = 0 \dots jobNumber - 1$ **do**
        for $j = 0 \dots jobNumber - 1$ **do**
            $distMatrix[i][j] = \max\{distMatrix[i][j], distMatrix[i][k] +$

```
                                    +distMatrix[k][j]};
            end
        end
    end
/*Set ES as the values of the 1st line of the final distMatrix and LS
  as the opposite's sign values of the 1st column*/
for i = 0 … jobNumber − 1 do
    ES[i] = distMatrix[0][i];
    LS[i] = −distMatrix[i][0];
end
return (ES[],LS[],distMatrix[][]);
```

Στον τελικό Πίνακα Αποστάσεων, η πρώτη γραμμή, η οποία αντιπροσωπεύει τη βοηθητική δραστηριότητα αρχής, αποτελεί τις νωρίτερες δυνατές ενάρξεις (*ES*) των δραστηριοτήτων του έργου, $ES_i = distMatrix_{0i}^{(n+1)}$. Οι αργότερες δυνατές ενάρξεις των δραστηριοτήτων του έργου υπολογίζονται ως οι μέγιστες αποστάσεις από κάθε δραστηριότητα προς την βοηθητική δραστηριότητα αρχής, δηλαδή από την πρώτη στήλη του τελικού Πίνακα Αποστάσεων ως $LS_i = distMatrix_{i0}^{(n+1)}$. Εν συνεχεία στο Σχήμα 23 παρουσιάζεται ο αρχικός και ο τελικός Πίνακας Αποστάσεων του δίγραφου του Σχήματος 6 (Κεφάλαιο 2.3.2.1).

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{4} & \mathbf{3} & -\infty & \mathbf{3} & -\infty \\ -\infty & \mathbf{0} & -\infty & \mathbf{1} & \mathbf{2} & -\infty & \mathbf{2} \\ -\infty & -\infty & \mathbf{0} & -\infty & -\infty & \mathbf{5} & -\infty \\ -\infty & -\infty & -\infty & \mathbf{0} & -\infty & \mathbf{2} & \mathbf{2} \\ -\infty & -\infty & -\infty & -\infty & \mathbf{0} & -\infty & \mathbf{5} \\ \mathbf{-11} & -\infty & \mathbf{-12} & -\infty & -\infty & \mathbf{0} & -\infty \\ -\infty & \mathbf{-11} & -\infty & \mathbf{-5} & -\infty & -\infty & \mathbf{0} \end{bmatrix} \quad \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{4} & \mathbf{3} & \mathbf{2} & \mathbf{9} & \mathbf{7} \\ \mathbf{-7} & \mathbf{0} & \mathbf{-3} & \mathbf{2} & \mathbf{2} & \mathbf{4} & \mathbf{7} \\ \mathbf{-6} & \mathbf{-6} & \mathbf{0} & \mathbf{-3} & \mathbf{-4} & \mathbf{5} & \mathbf{1} \\ \mathbf{-9} & \mathbf{-9} & \mathbf{-5} & \mathbf{0} & \mathbf{-7} & \mathbf{2} & \mathbf{2} \\ \mathbf{-9} & \mathbf{-6} & \mathbf{-5} & \mathbf{0} & \mathbf{0} & \mathbf{-2} & \mathbf{5} \\ \mathbf{-11} & \mathbf{-11} & \mathbf{-7} & \mathbf{-8} & \mathbf{-9} & \mathbf{0} & \mathbf{-4} \\ \mathbf{-14} & \mathbf{-11} & \mathbf{-10} & \mathbf{-5} & \mathbf{-9} & \mathbf{-3} & \mathbf{0} \end{bmatrix}$$

Σχήμα 23. Αρχικός και τελικός πίνακας αποστάσεων βάσει του Σχήματος 6

6

Πειραματική Επαλήθευση & Αξιολόγηση

6.1 Υλοποίηση και Εφαρμογή

Η υλοποίηση του προτεινόμενου αλγορίθμου "*do4u*", αποτελούμενου από το μοντέλο, τους αλγορίθμους GA-External και GA-Internal, τον sSGS-U καθώς και όλων των επιμέρους διαδικασιών, όπως περιγράφηκαν στο Κεφάλαιο 5, έλαβε χώρα κάνοντας χρήση της γλώσσας προγραμματισμού *Java*, μέσω του ολοκληρωμένου περιβάλλοντος ανάπτυξης (*IDE*) Netbeans 8.0. Η Java αποτελεί μια αντικειμενοστρεφή γλώσσα προγραμματισμού που σχεδιάστηκε από την Sun Microsystems, βασικό πλεονεκτήμα της οποίας έναντι των περισσότερων άλλων γλωσσών προγραμματισμού, είναι η ανεξαρτησία του λειτουργικού συστήματος και πλατφόρμας που δίδει τη δυνατότητα σε προγράμματα που είναι γραμμένα σε *Java*, να τρέχουνε ακριβώς το ίδιο σε Windows, Linux, Unix, Macintosh χωρίς να χρειαστεί να ξαναγίνει μεταγλώττιση (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα.

Επιπρόσθετα, ο κώδικας που δημιουργήθηκε παρέχει μια απλή εφαρμογή κονσόλας (*console application*) που χρησιμοποιήθηκε για την υλοποίηση των πειραματικών διαδικασίων επαλήθευσης, σύγκρισης και αξιολόγησης της αποδοτικότητας του παραχθέντος αλγορίθμου επίλυσης, του "*do4u*".

6.2 Σχεδιασμός Πειραμάτων

Για την πειραματική επαλήθευση της αποτελεσματικότητας και της αποδοτικότητας, καθώς και για την καθολική αξιολόγηση του προτεινόμενου αλγορίθμου επίλυσης, σχεδιάστηκαν και υλοποιήθηκαν τρεις διαφορετικές κατηγορίες πειραμάτων, οι οποίες συνοψίζονται ως εξής:

1. Αναλυτική συγκριτική παρουσίαση προσφερόμενων λύσεων σε παραδείγματα RCPSP και RCPSP/max. Αρχικά, πραγματοποιείται η συγκριτική παρουσίαση ενός παραδείγματος για κάθε κατηγορία προβλήματος RCPSP που επιλύει ο προτεινόμενος αλγόριθμος, μεταξύ της βέλτιστης στη βιβλιογραφία λύσης και μιας τυχαίας λύσης του αλγορίθμου "*do4u*". Απώτερος στόχος της συγκριτικής παρουσιάσης της διαδικασίας επίλυσης των δύο παραδειγμάτων αποτελεί αφενός η εστίαση του ενδιαφέροντος στον τρόπο προσέγγισης, στη διαδικασία επίλυσης και στα παραχθέντα

αποτελέσματα του προτεινόμενου τρόπου επίλυσης προβλημάτων προγραμματισμού έργων μέσα από ευέλικτα προφίλ πόρων, αφετέρου δε η απόδοση μιας πρώτης αίσθησης της ποιότητας των παραγόμενων αποτελέσματων αναφορικά με υπάρχοντα διαθέσιμα βέλτιστα, μέσα από την επίλυση προβλημάτων RCPSP και παραλλαγών του όπως το RCPSP/max, που προσφέρει ο προτεινόμενος αλγόριθμος μέσα από τη διαχείριση ευέλικτων προφίλ πόρων.

2. Πειραματική συκριτική αξιολόγηση. Ύστερα, πραγματοποιείται μια σύγκριση των αποτελεσμάτων που δίδει η προτεινόμενη μέθοδος επίλυσης με τα αντίστοιχα δεδομένα βέλτιστα αποτελέσματα, διαθέσιμα για κάθε σύνολο παραδειγμάτων στη διεθνή βιβλιογραφία (Kolisch and Sprecher, 1997), για κάθε επιμέρους πρόβλημα προγραμματισμού που αφορά το μοντέλο του προβλήματος. Στόχος των πειραμάτων αυτών, αποτελεί η απόδειξη ότι το προτεινόμενο πρόγραμμα, επιλύει αποτελεσματικά επιμέρους προβλήματα, όπως το RCPSP και το RCPSP/max, οδηγώντας σε λύσεις με μικρή απόκλιση από την υπάρχουσα βέλτιστη, αν όχι στη βέλτιστη ή σε μία λύση καλύτερη από αυτήν.
3. Πειραματικά αποτελέσματα παραδειγμάτων υπό μελέτη προβλήματος. Δεδομένης της πρόσφατης εισόδου του αντικειμένου των ευέλικτων προφίλ πόρων στο επιστημονικό προσκήνιο του προγραμματισμού έργων, και κατ' επέκταση της έλλειψης δημοσιεύσεων και πειραματικών αποτελεσμάτων επίλυσης προβλημάτων προγραμματισμού έργων με περιορισμένους πόρους μέσω ευέλικτων προφίλ εκτέλεσης, καθώς και της καινοτομικής σκοπιάς διαχείρισης των εύλικτων προφίλ από το προτεινόμενο μοντέλο αναφορικά με τις ελάχιστες υπάρχουσες ερευνητικές μελέτες (Ranjbar and Kianfar, Naber and Kolisch, 2014b, Tritschler et al., 2014), σχεδιάστηκαν δύο παραδείγματα του υπό μελέτη προβλήματος, τα πειραματικά αποτελέσματα των οποίων δεν υπόκεινται σε κάποια σύγκριση. Αντίθετα, παρουσιάζονται για την παροχή μιας ολοκληρωμένης εικόνας του τρόπου λειτουργίας του προτεινόμενου αλγορίθμου, καθώς και της λογικής της μοντελοποίησης του χρονοπρογραμματισμού με βάση την απαιτούμενη προσπάθεια ανά δραστηριότητα και όχι της συνήθους τακτικής, όπου ο προγραμματισμός βασίζεται σε λιγότερο ή περισσότερο επιτυχείς προσεγγίσεις της διάρκειας κάθε δραστηριότητας.

Όσον αφορά τις παραμέτρους του αλγορίθμου "*do4u*" που επιλέχθηκαν για την επιλύση κάθε πειραματικής διαδικασίας, τα εύρη τιμών καθέ μιας εξ αυτών ορίστηκαν ύστερα από ανάλυση της χρονικής απόδοσης και της ποιότητας των υπολογιζόμενων λύσεων σε πολλαπλές επαναλήψεις εκτέλεσης επιλύσεων στιγμιοτύπων με γνωστά εκ των προτέρων τα βέλτιστα αποτελέσματα. Παρόλα ταύτα, σημειώνεται ότι λόγω της έλλειψης συγκριτικών αποτελεσμάτων στη διεθνή βιβλιογραφία που θα έδιναν τη δυνατότητα προσδιορισμού των τιμών που αντιστοιχούν στη βέλτιστη απόδοση του προτεινόμενου αλγορίθμου, απαιτείται περαιτέρω διερεύνηση ώστε να ολοκληρωθεί ο προσδιορισμός των ιδανικών παραμέτρων που θα οδηγήσουν στη βέλτιστη απόδοση του "*do4u*".

6.3 Ορισμός Παραμέτρων & Εισόδων

Για την επίλυση των προβλημάτων RCPSP και RCPSP/max από τον προτεινόμενο αλγόριθμο "*do4u*" απαιτείται η μετατροπή των αντίστοιχων αρχείων εισόδου, σε αποδεκτές από το προτεινόμενο μοντέλο εισόδους (Κεφάλαιο 2.3). Τα αντίστοιχα, λοιπόν, δεδομένα που προέρχονται από τα αρχεία του κάθε τύπου προβλήματος, υφίστανται μια προεπεξεργαστική

διαδικασία μετατροπής στις απαιτούμενες εισόδους του μοντέλο, ώστε να επιλυθούν μέσω ευέλικτων προφίλ πόρων.

Ειδικότερα, η απαιτούμενη προσπάθεια ανά δραστηριότητα και πόρο προέρχεται από το γινόμενο της διάρκειας της δραστηριότητας και της απαιτούμενης ποσότητας του αντίστοιχου πόρου ανά χρονική περίοδο όπως δίδεται στο κάθε πρόβλημα. Επιπρόσθετα, σε προβλήματα RCPSP, όπου οι συσχετίσεις μεταξύ των δραστηριοτήτων αφορούν σχέσεις τέλους-αρχής, γίνεται μετατροπή τους σε σχέσεις αρχής-αρχής με χρονική καθυστέρηση να λαμβάνεται η μελλοντικά υπολογιζόμενη διάρκεια της προαπαιτούμενης δραστηριότητας (οι διάρκειες των δραστηριοτήτων υπολογίζονται από τα αντίστοιχο προφίλ εκτέλεσης, όπως περιγράφηκε στο Κεφάλαιο 5), ώστε το προκύπτον χρονοπρόγραμμα να σέβεται τις αρχικές σχέσεις τέλους-αρχής μεταξύ των δραστηριοτήτων του προβλήματος. Στο σημείο αυτό σημειώνεται ότι οι χρονικές καθυστερήσεις που δίδονται στα προβλήματα RCPSP/max, θεωρούνται σταθερές κατά την επίλυση του προβλήματος μέσω του προτεινόμενου μοντέλου και δεν υπολογίζεται το ποσοστό τους σε σχέση με τη διάρκεια της δραστηριότητας που αφορούν.

Όσον αφορά τις παραμέτρους που απαιτεί το μοντέλο ως εισόδους για την εκτέλεση των πειραμάτων, λαμβάνουν τιμές με στόχο τη σύγκριση της συνολικής διάρκειας κάθε προβλήματος με ελάχιστες, κατά το δυνατόν, διαφοροποιήσεις σε σχέση με τη δομή των αντίστοιχων προβλημάτων. Οι τιμές των παραμέτρων είναι οι ακόλουθες: (α) οι παραγωγικές δυναμικότητες των πόρων θεωρούνται ίσες με τη μονάδα, (β) οι ελάχιστες και οι μέγιστες επιτρεπόμενες χρησιμοποιούμενες ποσότητες ανά χρονική περίοδο των πόρων λαμβάνουν αντίστοιχα τιμές ίσες με τη μονάδα και ίσες με τη συνολική διαθεσιμότητα του αντίστοιχου πόρου, (γ) ο μέγιστος επιτρεπόμενος αριθμός διαμερίσεων θεωρείται ίσος με τη μονάδα επιτρέποντας ως δύο αλλαγές στις χρησιμοποιούμενες ποσότητες ανά προφίλ εκτέλεσης, (δ) το ελάχιστο χρονικό διάστημα σταθερής ποσότητας των χρησιμοποιούμενων πόρων της κάθε δραστηριότητας λαμβάνει τιμή ίση με μία χρονική μονάδα.

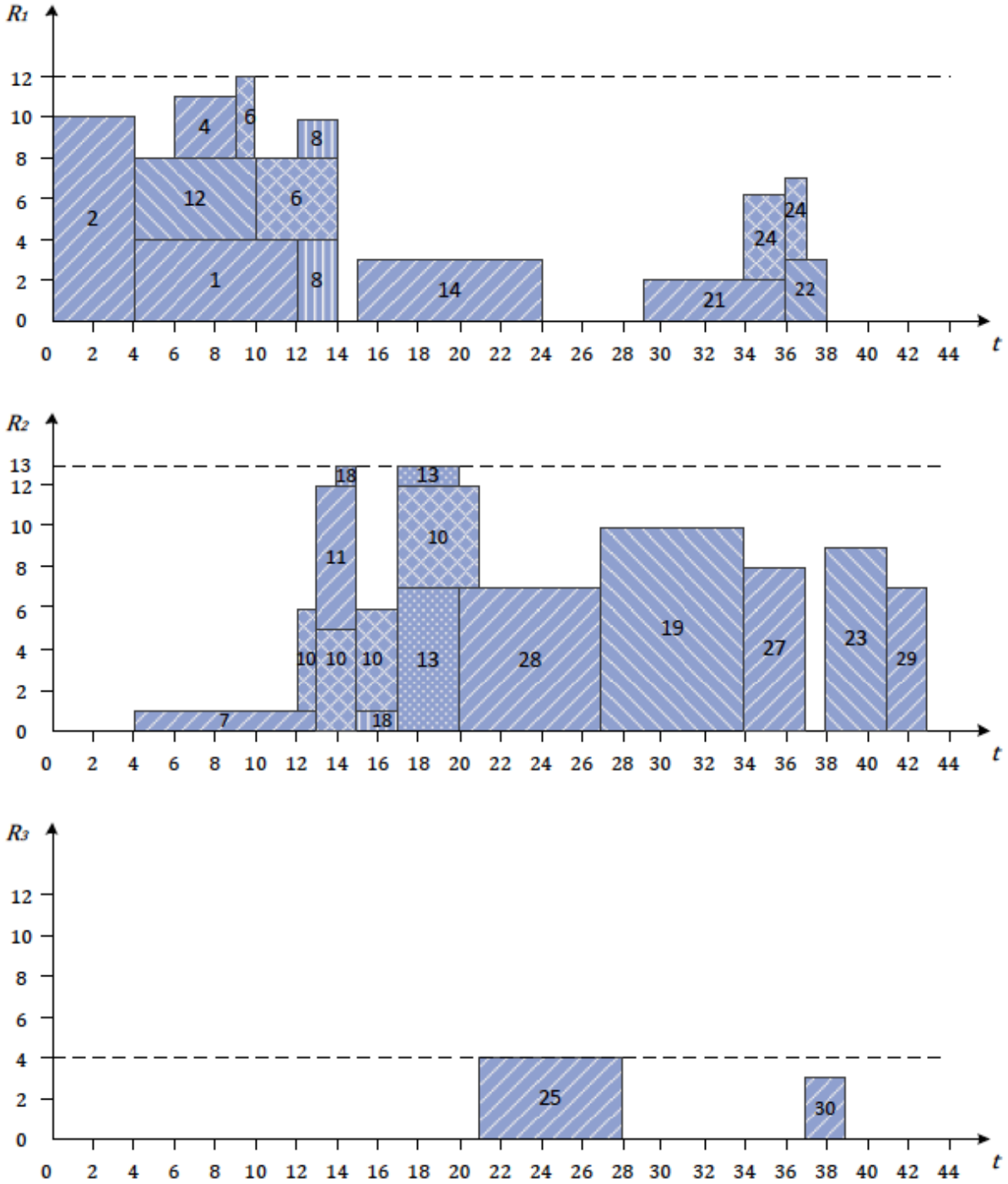
Τέλος, όλα τα πειράματα έλαβαν χώρα με τις βασικές παραμέτρους των γενετικών αλγορίθμων που αποτελούν τον προτεινόμενο αλγόριθμο "*do4u*" (Κεφάλαια 5.2.3 και 5.2.4) να λαμβάνουν τις ακόλουθες τιμές:$POP_{external} = 50, POP_{internal} = 20, Genes_{external} = 20,$ $Genes_{internal} = 15,$ $P_{mut,chrom} = 0.4,$ $P_{mut,res} = 0.4,$ $P_{mut,individ} = 0.2.$ Οι υψηλές τιμές των πιθανοτήτων μετάλλαξης του εξωτερικού γενετικού αλγορίθμου, αποδίδονται στο γεγονός ότι ο τελεστής μετάλλαξης αποτελεί τη γενεσιουργό διαδικασία παραγωγής χρωμοσωμάτων με διαφορετικούς πίνακες διαμερίσεων που εν συνεχεία οδηγούν σε χρωμοσώματα διαφορετικών προφίλ εκτέλεσης.

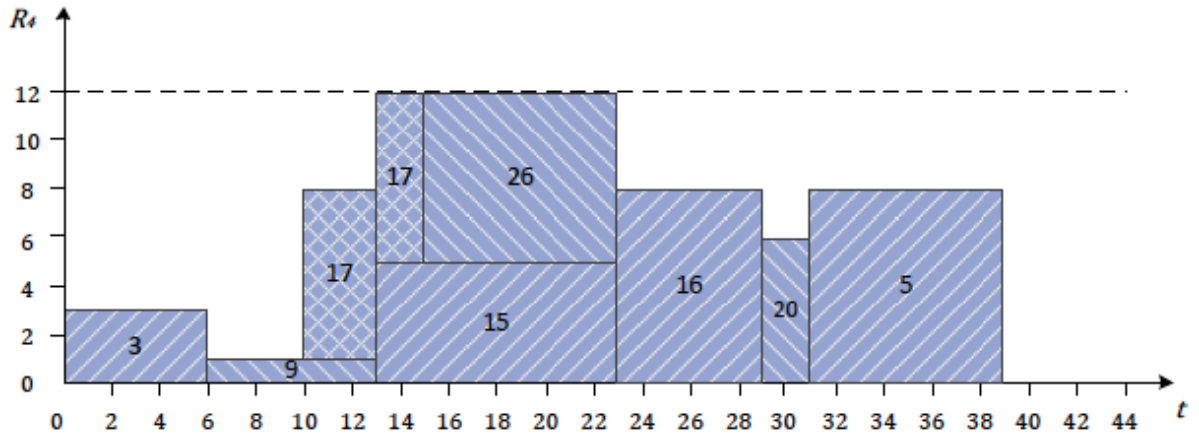
6.4 Αναλυτική Συγκριτική Παρουσιάση Λύσεων

Για τη σύγκριση των λύσεων μεταξύ της βέλτιστης στη βιβλιογραφία λύσης και μιας τυχαίας λύσης του προτεινόμενου αλγορίθμου ενός RCPSP προβλήματος παρουσιάζεται το παράδειγμα J301_1, διαθέσιμο στη διαδικτυακή βιβλιοθήκη προβλημάτων προγραμματισμού έργων PSPLIB, ενώ για τη σύγκριση των αντίστοιχων λύσεων ενός RCPSPS/max προβλήματος παρουσιάζεται το παράδειγμα PSP13, διαθέσιμο στη διαδικτυακή βιβλιοθήκη PSP/max του συνόλου παραδειγμάτων UBO10. Απώτερος στόχος της συγκριτικής παρουσίασης της διαδικασίας επίλυσης των δύο παραδειγμάτων αποτελεί η απόδοση μιας πρώτης αίσθησης της ποιότητας των παραγόμενων αποτελέσματων, καθώς και η εστίαση του ενδιαφέροντος στη διαδικασία επίλυσης του προτεινόμενου τρόπου αντιμετώπισης προβλημάτων προγραμματισμού έργων μέσα από ευέλικτα προφίλ πόρων.

6.4.1 Παράδειγμα J301_1 - RCPSP Πρόβλημα

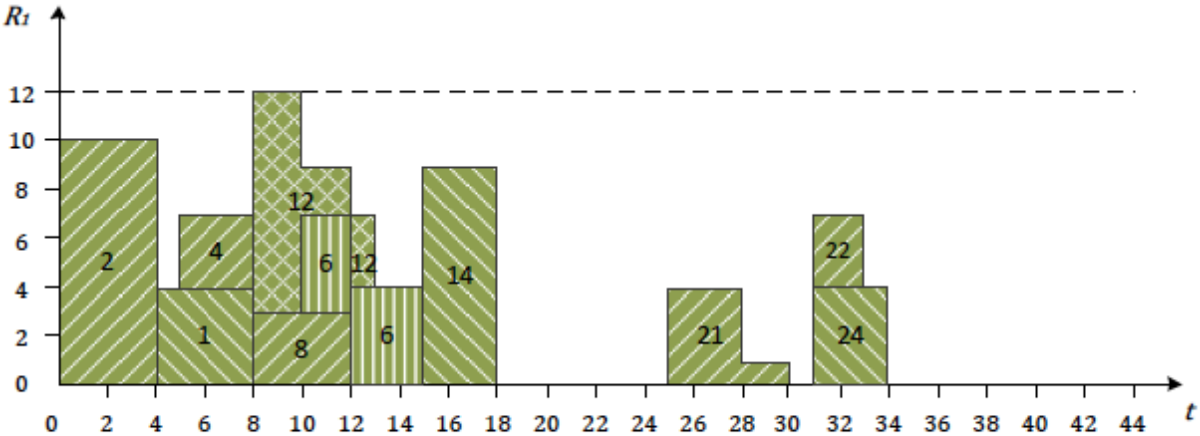
Στα Σχήματα 24 & 25 παρατίθενται τα χρονοπρογράμματα ανά πόρο της βέλτιστης στη βιβλιογραφία λύσης (μπλε χρώμα) και μιας τυχαίας λύσης του προτεινόμενου αλγορίθμου (πράσινο χρώμα), αντίστοιχα, όπου σκιαγραφείται η δομή των παραγόμενων προφίλ εκτέλεσης. Στα Σχήματα 26 & 27 παρατίθενται και τα αντίστοιχα διαγράμματα Gantt, με τις συνολικές διάρκειες και των δύο λύσεων να ανέρχονται στις 43 χρονικές μονάδες.

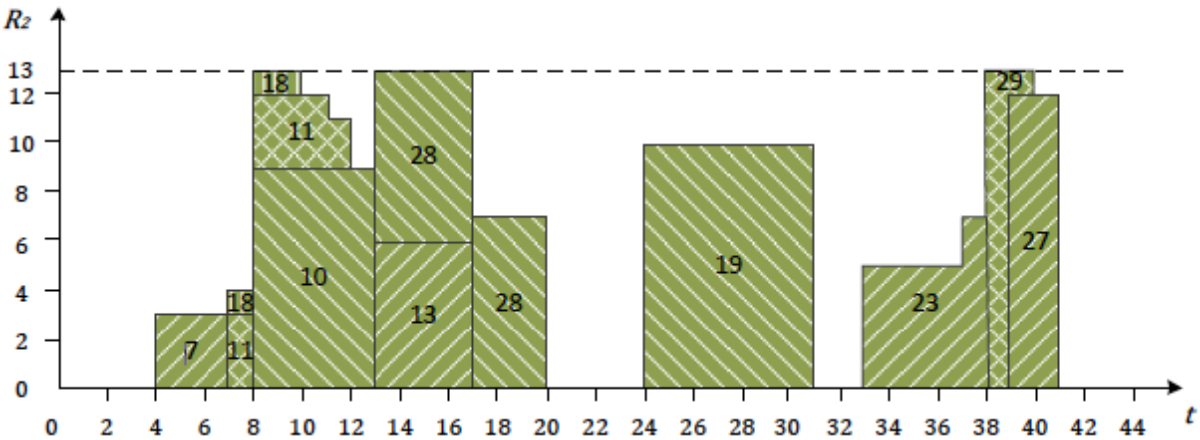


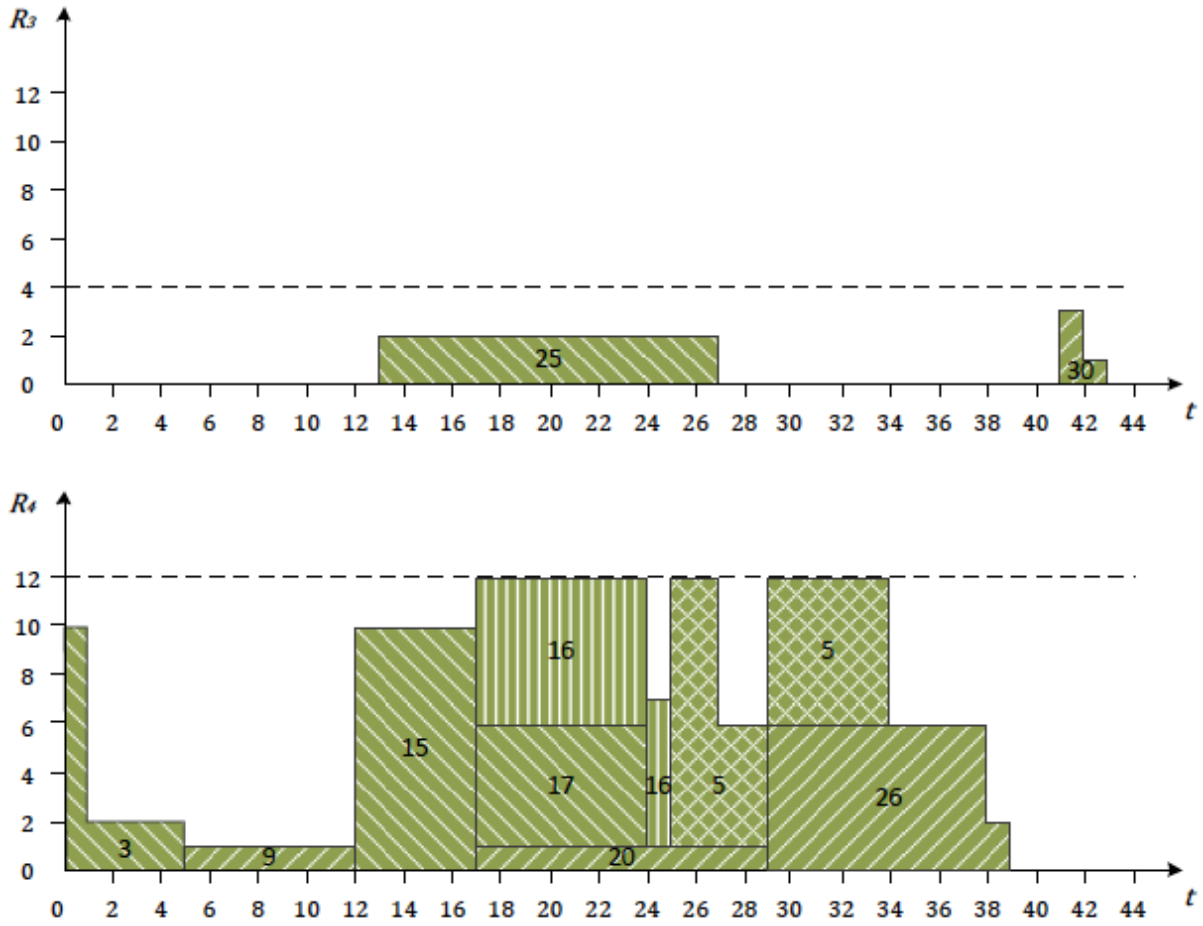


Σχήμα 24. Χρονοπρογράμματα βέλτιστης λύσης παραδείγματος J301_1

Από τη σχηματική μορφή των παραγόμενων χρονοπρογραμμάτων μέσα από την επίλυση με ευέλικτα προφίλ πόρων, σκιαγραφείται η προσπάθεια για ομαλή διαχείριση του κάθε πόρου σε όλη τη διάρκεια ζωής του έργου, παρότι τα αρχικά δεδομένα που έχουν εισαχθεί για την υλοποίηση της σύγκρισης δεν θέτουν περιορισμούς, όπως ελάχιστης χρησιμοποιούμενης ποσότητας ανά χρονική περίοδο. Επιπρόσθετα, σημειώνεται ότι η επίλυση μέσω ευέλικτων προφίλ που υλοποιείται στο συγκεκριμένο σημείο, έχει ως απώτερο στόχο την ελαχιστοποίηση της διάρκειας του έργου, αψηφώντας περαιτέρω στόχους που υπάρχει η δυνατότητα να ικανοποιήσει το προτεινόμενο μοντέλο, γεγονός που θα οδηγούσε σε πιο ομαλή δομή των προφίλ των αντίστοιχων πόρων.



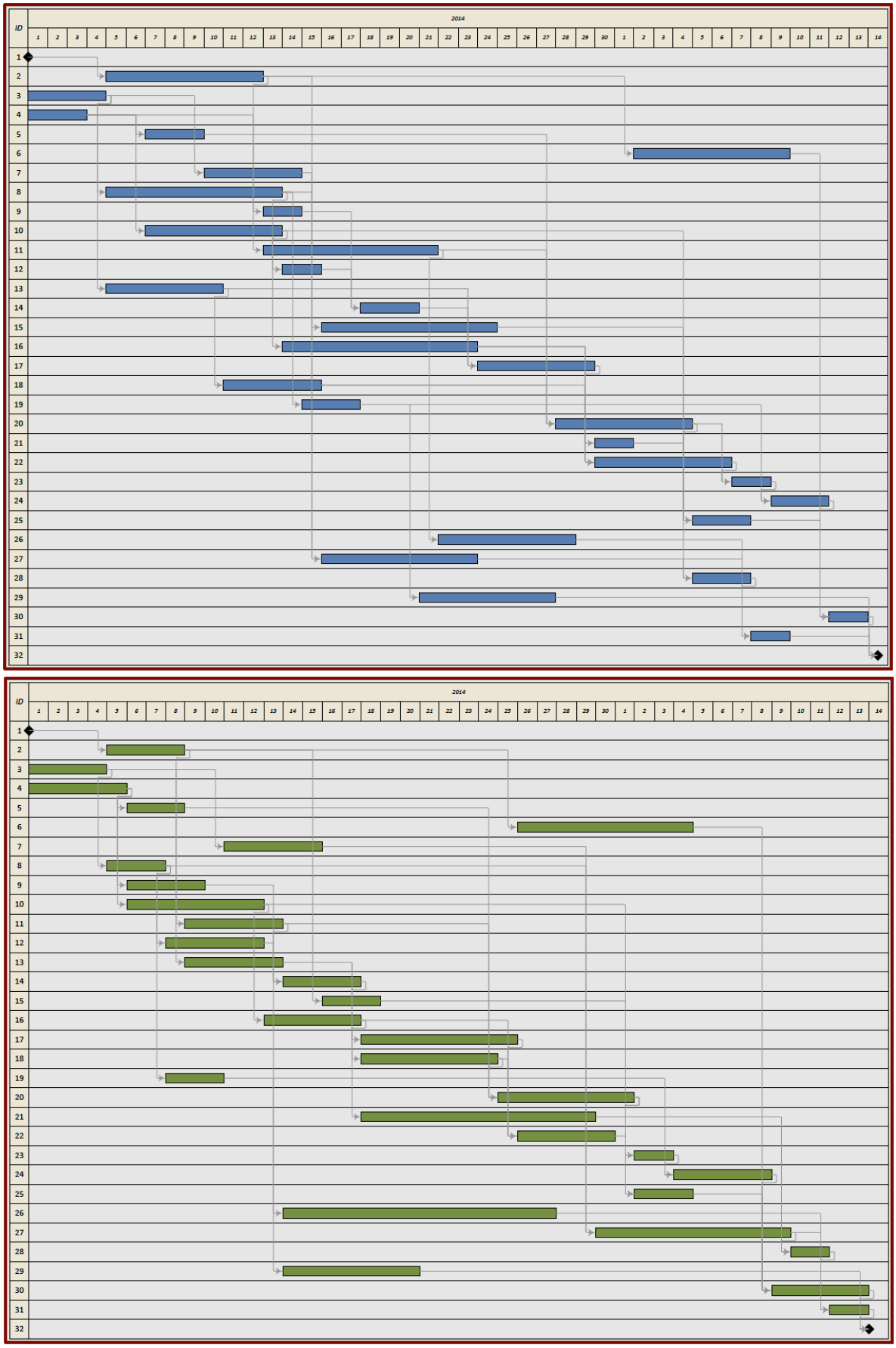




Σχήμα 25. Χρονοπρογράμματα προτεινόμενης λύσης με ευέλικτα προφίλ παραδείγματος J301_1

Η συνεχής προσπάθεια του προτεινόμενου αλγορίθμου να αξιοποιήσει κάθε χρονική στιγμή, και εφόσον οι υφιστάμενες σχέσεις προτεραιότητας το επιτρέπουν, οποιαδήποτε διαθέσιμη ποσότητα κάθε πόρου με στόχο την ελαχιστοποίηση της συνολικής διάρκειας του έργου, απεικονίζεται στη δομή των προφίλ που λαμβάνουν οι δραστηριότητες στο αντίστοιχο χρονοπρόγραμμα. Χαρακτηριστικά, εστιάζεται η προσοχή στη δομή του προφίλ της δραστηριότητα υπ' αριθμόν 29 στα χρονοπρογράμματα του πόρου R_2 των δύο λύσεων. Ο προτεινόμενος αλγόριθμος δημιούργησε ένα προφίλ με στόχο την αξιοποίηση της διαθέσιμης ποσότητας του πόρου R_2 όπου τη χρονική στιγμή 37 όλη η διαθέσιμη ποσότητα του πόρου παραμένει ανεκμετάλλευτη στο χρονοπρόγραμμα της βέλτιστης λύσης. Τέλος, σημειώνεται ότι κάποια μη βέλτιστα προφίλ πόρων που παρουσιάζονται στη λύση του προτεινόμενου αλγορίθμου, οφείλονται στο μη καθορισμό περιορισμών, ώστε να λάβει χώρα η σύγκριση των δύο λύσεων, και στην τυχαιότητα των προφίλ όπου ως μοναδικό στόχο στο συγκεκριμένο παράδειγμα έχουν τη μείωση της διάρκειας του έργου.

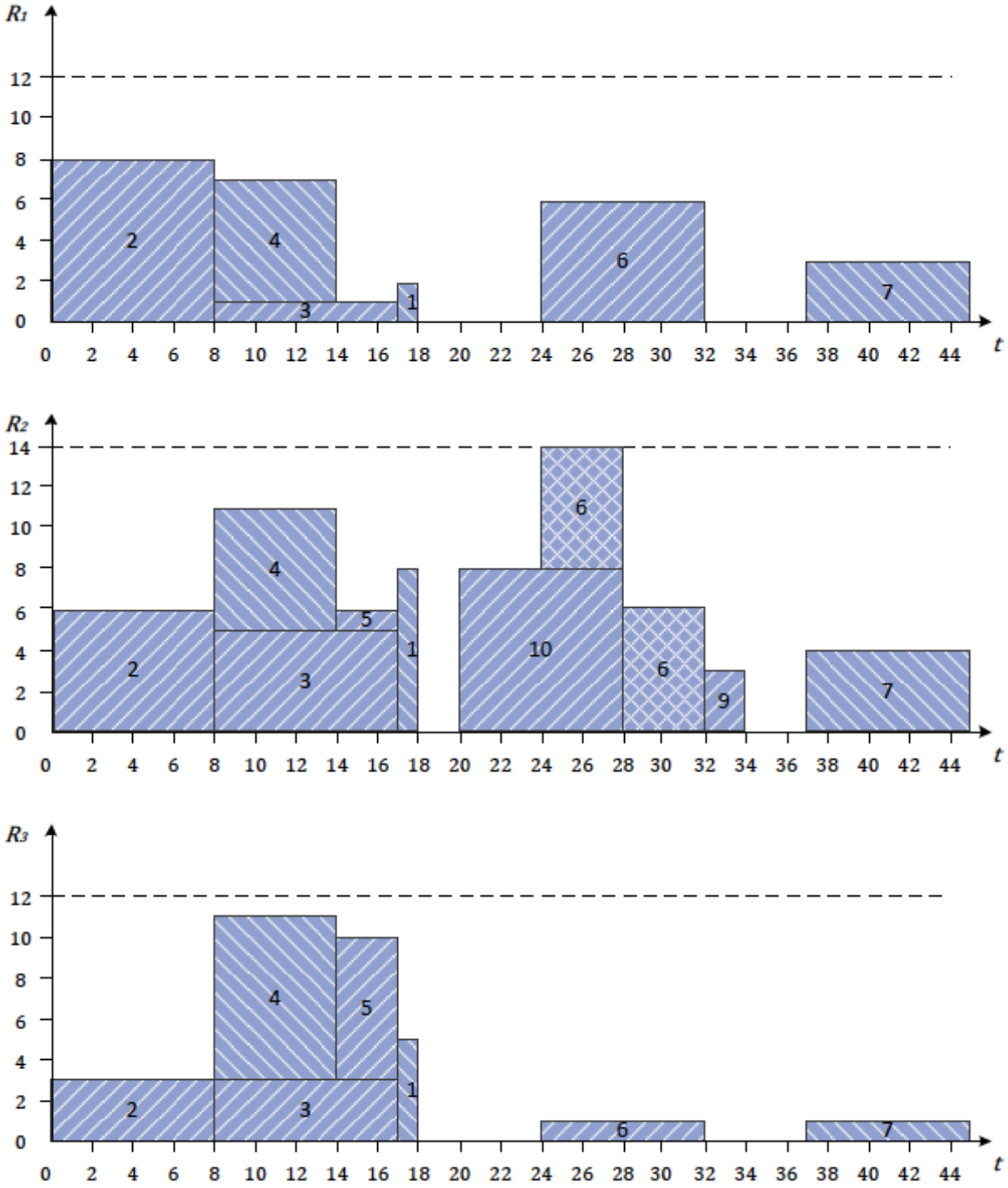
Στα Σχήματα 26 & 27 παρουσιάζονται τα αντίστοιχα διαγράμματα Gantt, με τις συνολικές διάρκειες και των δύο λύσεων να ανέρχονται στις 43 χρονικές μονάδες.

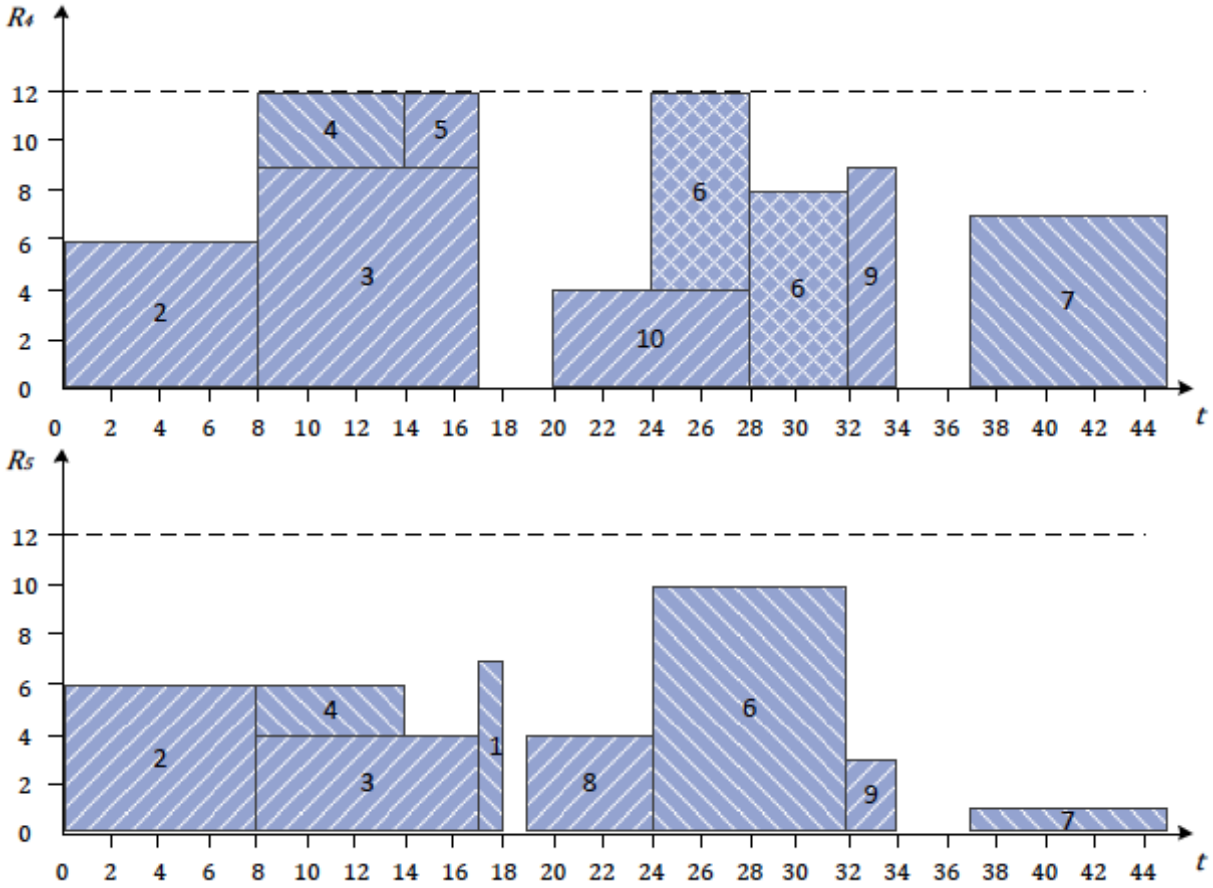


Σχήμα 26. Διαγράμματα Gantt λύσεων παραδείγματος J301_1

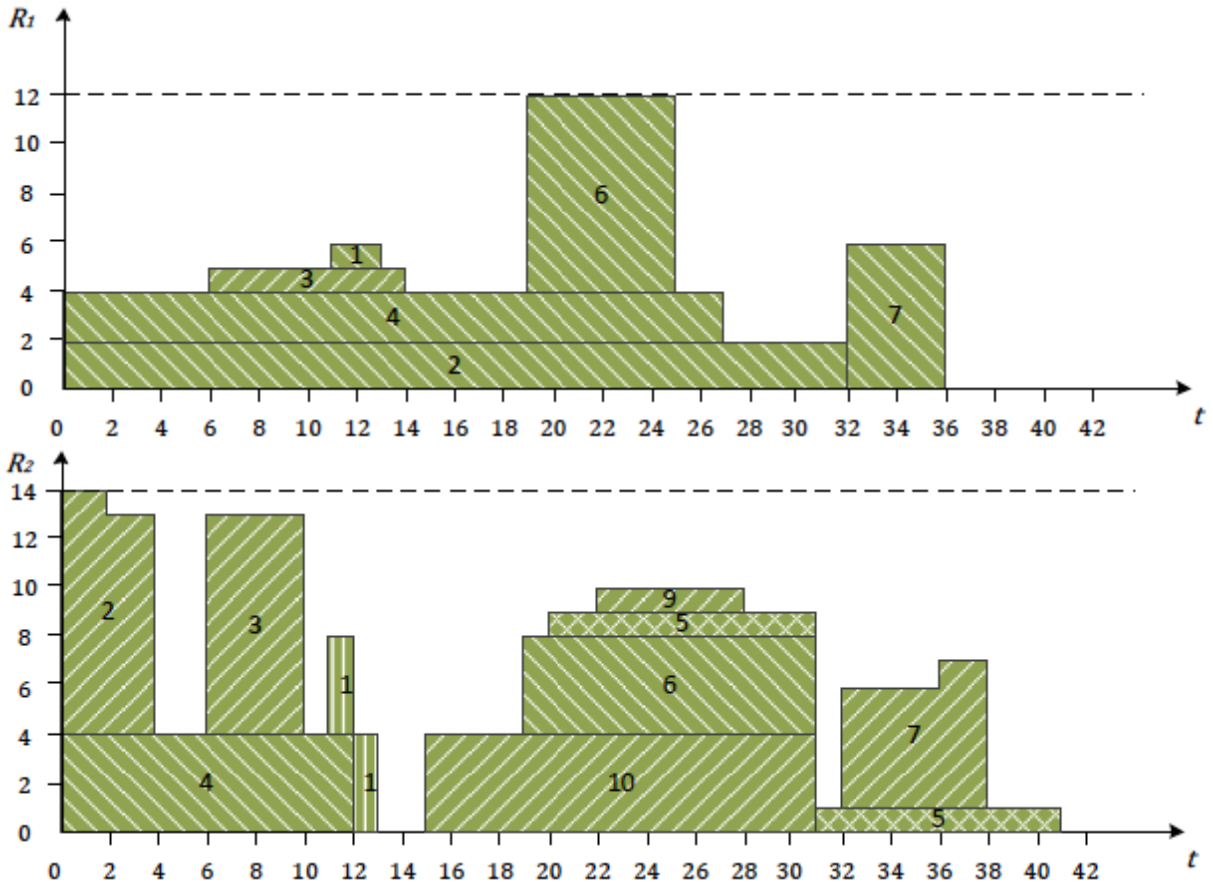
6.4.2 Παράδειγμα PSP13 - RCPSP/max Πρόβλημα

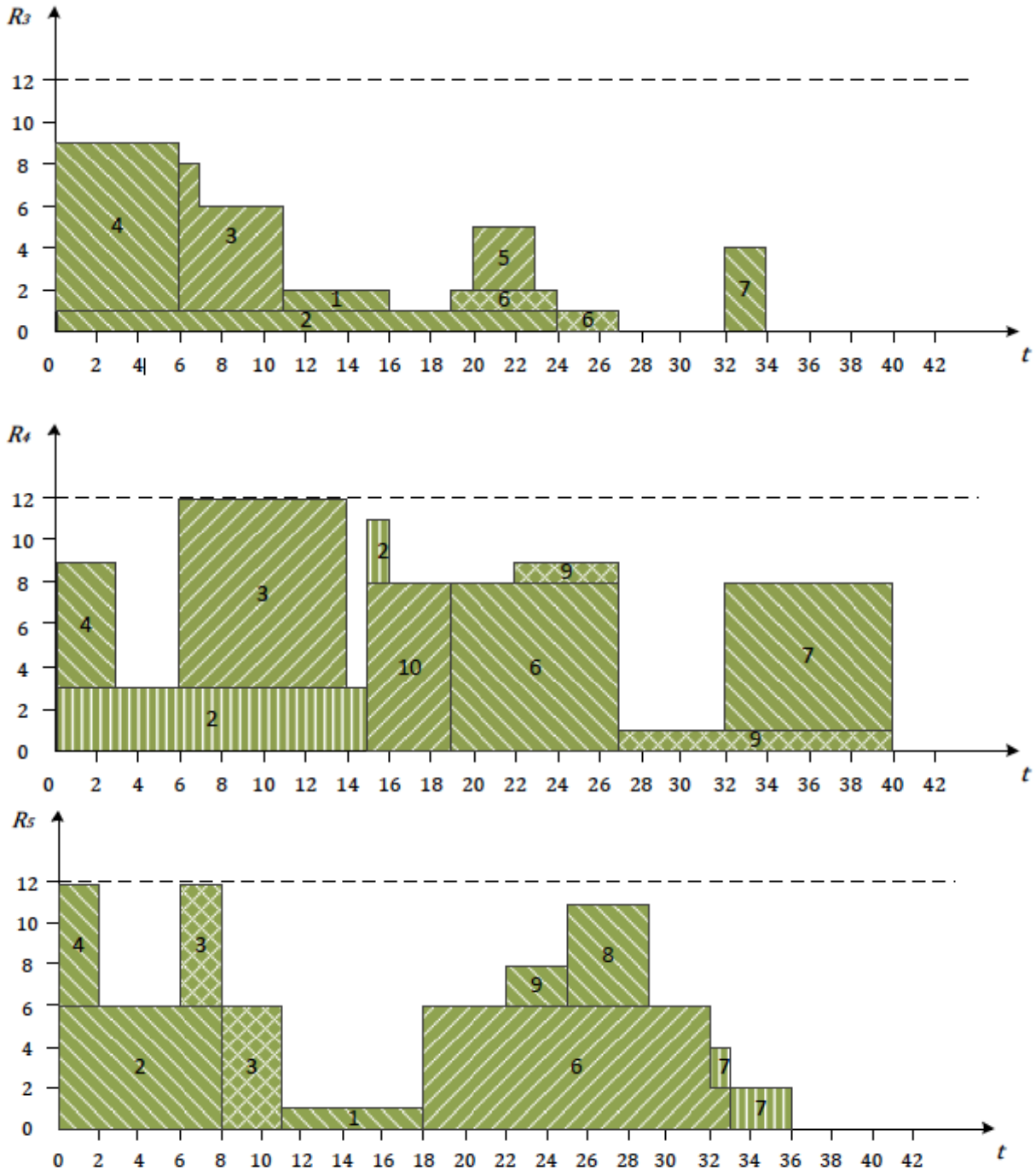
Στα Σχήματα 27 & 28 παρουσιάζονται τα χρονοπρογράμματα ανά πόρο της βέλτιστης στη βιβλιογραφία λύσης (μπλε χρώμα) και μιας τυχαίας λύσης του προτεινόμενου αλγορίθμου (πράσινο χρώμα), αντίστοιχα, όπου σκιαγραφείται η δομή των παραγόμενων προφίλ εκτέλεσης.





Σχήμα 27. Χρονοπρογράμματα βέλτιστης λύσης παραδείγματος PSP13

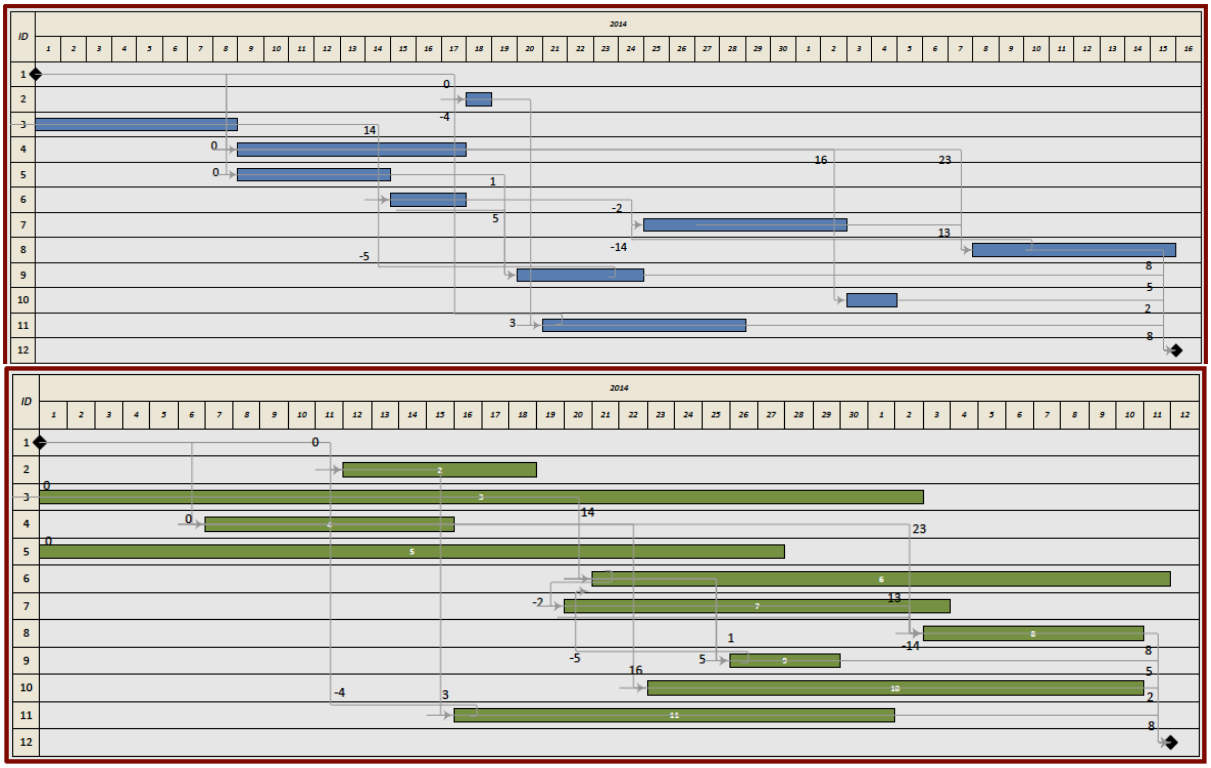




Σχήμα 28. Χρονοπρογράμματα προτεινόμενης λύσης με ευέλικτα προφίλ παραδείγματος PSP13

Από τη σχηματική μορφή των χρονοπρογραμμάτων κάθε πόρου των δύο λύσεων, σκιαγραφείται η καλύτερη αξιοποίηση των διαθέσιμων πόρων από το προτεινόμενο μοντέλο, όπου με σεβασμό στις γενικευμένες σχέσεις προτεραιότητας οδηγεί στην παραγωγή προφίλ εκτέλεσης που όχι μόνο ελαχιστοποιούν τη διάρκεια του έργου, αλλά τη μειώνουν κατά 2 χρονικές μονάδες αναφορικά με την ως τώρα γνωστή στη βιβλιογραφία βέλτιστη λύση.

Στα Σχήματα 29 & 30 παρατίθενται τα αντίστοιχα διαγράμματα Gantt, με τη συνολική διάρκεια της τυχαίας λύσης του προτεινόμενου αλγορίθμου να ανέρχεται σε 41 χρονικές μονάδες, εν αντιθέσει με τις 43 χρονικές μονάδες της βέλτιστης στη βιβλιογραφία λύσης.



Σχήμα 29. Διαγράμματα Gantt λύσεων παραδείγματος PSP13

Σημειώνεται, ότι η βέλτιστη στη βιβλιογραφία λύση έχει επέλθει από αλγόριθμο επίλυσης με χρήση ποινών, γεγονός που αιτιολογεί την μη ικανοποίηση κάποιων σχέσεων προτεραιότητας όπως αποτυπώνονται στο αντίστοιχο διάγραμμα Gantt. Σε κάθε περίπτωση, το διάγραμμα Gantt του προτεινόμενου αλγορίθμου, ικανοποιεί όλες τις υφιστάμενες σχέσεις προτεραιότητας.

6.5 Πειραματική Συγκριτική Αξιολόγηση

Για την πειραματική επαλήθευση της αποτελεσματικότητας και της αποδοτικότητας του υλοποιημένου συστήματος μέσα από την πειραματική αξιολόγηση των αποτελεσμάτων που δίδει η προτεινόμενη μέθοδος επίλυσης, χρησιμοποιήθηκαν δύο διαφορετικά σύνολα περιπτώσεων προβλημάτων, ένα για κάθε παραλλαγή του RCPSP που αντιμετωπίζει το μοντέλο. Για το RCPSP πρόβλημα χρησιμοποιήθηκαν τα σύνολα προβλημάτων J30 και J60 (30 και 60 δραστηριότητες προς προγραμματισμό ανά έργο, αντίστοιχα, με 4 διαφορετικούς ανανεώσιμους πόρους) αποτελούμενα από 480 προβλήματα το κάθε ένα, διαθέσιμα στη διαδικτυακή βιβλιοθήκη προβλημάτων προγραμματισμού έργων PSPLIB. Για το RCPSP/max πρόβλημα χρησιμοποιήθηκαν τα σύνολα προβλημάτων UBO10 και UBO20 (10 και 20 δραστηριότητες προς προγραμματισμό ανά έργο, αντίστοιχα, με 5 διαφορετικούς ανανεώσιμους πόρους) αποτελούμενα από 90 προβλήματα το κάθε ένα, διαθέσιμα στη διαδικτυακή βιβλιοθήκη PSP/max. Για κάθε ένα πρόβλημα έλαβαν χώρα 100 επαναλήψεις, όπου δίχως στατιστική αμφιβολία αναφορικά με την τυχαιότητα των αποτελεσμάτων, υπολογίστηκαν οι μέσες τιμές των αποτελεσμάτων.

Τα αποτελέσματα του προτεινόμενου αλγορίθμου που παρουσιάζονται είναι η ελάχιστη, η μέγιστη και η μέση διάρκεια των λύσεων που βρέθηκαν, καθώς και η μέση απόκλιση όλων των λύσεων από τα αντίστοιχα βέλτιστα αποτελέσματα. Επιπρόσθετα, για τα

RCPSP προβλήματα παρουσιάζεται και ο συντελεστής *σπανιότητας πόρων* (*Resource Strength - RS*). Σημειώνεται, ότι ο RS, αποτελεί ένα μέτρο περιγραφής της σπανιότητας της διαθεσιμότητας των πόρων, ως το λόγο της διαθέσιμης ποσότητας του πόρου πλην την ελάχιστη απαιτούμενη ποσότητα από το συγκεκριμένο πόρο για την εκτέλεση των δραστηριοτήτων (ουσιαστικά αποτελεί τη μέγιστη απαιτούμενη ποσότητα του συγκεκριμένου πόρου από όλες τις δραστηριότητες του έργου ανά χρονική περίοδο) προς τη διαφορά της ελάχιστης απαιτούμενης ποσότητας του πόρου από την αντίστοιχη μέγιστη απαιτούμενη ποσότητα του πόρου όταν οι δραστηριότητες προγραμματιστούν στις χρονικές στιγμές έναρξης των νωρίτερων δυνατών ενάρξεων ($ES_{,CPM}$) (Demeulemeester and Herroelen, 2002). Η χρήση του συντελεστή RS στα RCPSP/max προβλήματα δεν ενδείκνυται, καθώς δεν έχει αποδειχθεί η συσχέτιση της φύσης του συντελεστή με τη διαδικασία επίλυσης και τον τρόπο αντιμετώπισης των πόρων από τα συγκεκριμένα προβλημάτα (Demeulemeester and Herroelen, 2002).

Τα πειράματα πραγματοποιήθηκαν κάνοντας χρήση ενός υπολογιστή με τα εξής χαρακτηριστικά επεξεργαστή 2.2 GHz Intel Core i7 και μνήμης 4 GB 1333 MHz DDR3, αντίστοιχα. Αποσπάσματα των πειραματικών αποτελεσμάτων για κάθε πρόβλημα και σύνολο παραδειγμάτων, παρατίθενται στους Πίνακες 10 & 11, ενώ τα πειραματικά αποτελέσματα όλων των προβλημάτων παρουσιάζονται στους Πίνακες 16 & 17 (Παράρτημα Α). Τα αποτελέσματα που προήλθαν από την εκτέλεση του προτεινόμενου αλγορίθμου για κάθε ένα πρόβλημα, συγκρίνονται είτε με τη βέλτιστη λύση, σε περίπτωση που είνα γνωστή στη διεθνή βιβλιογραφία, είτε με την καλύτερη ως τώρα ευρεθείσα τιμή, απόρροια οποιοδήποτε άλλου ευρετικού αλγορίθμου.

Πίνακας 10. Απόσπασμα πειραματικών αποτελεσμάτων προβλημάτων RCPSP των συνόλων J30&60

Όνομα αρχείου	Βέλτιστη τιμή	Ελάχιστη Διάρκεια	Μέγιστη Διάρκεια	Μέση Διάρκεια	Μέση απόκλιση από βέλτιστη τιμή
J301_5	39	37	42	39	0.00%
J301_6	48	45	52	47	-2.08%
J302_4	43	36	42	40	-6.98%
J302_5	51	47	52	49	-3.92%
J302_6	47	41	53	46	-2.13%
J302_7	47	40	47	43	-8.51%
J302_8	54	48	54	51	-5.56%
J303_5	53	49	55	50	-5.66%
J303_6	54	48	51	49	-9.26%
J303_7	48	34	39	36	-25.00%
J303_8	54	45	52	49	-9.26%
J303_9	65	51	54	52	-20.00%
J601_4	91	91	104	96	5.49%
J602_1	65	65	72	68	4.62%
J602_2	82	73	87	82	0.00%
J602_10	69	73	82	76	10.14%

Όνομα αρχείου	$RS_{1,opt}$	$RS_{2,opt}$	$RS_{3,opt}$	$RS_{4,opt}$	$RS'_{1,avg}$	$RS'_{2,avg}$	$RS'_{3,avg}$	$RS'_{4,avg}$
J301_5	0.19	0.20	0.18	0.20	0.14	0.02	0.11	0.11
J301_6	0.25	0.17	0.00	0.18	0.11	0.12	0.17	0.04

J302_4	0.53	0.00	0.55	0.57	0.09	0.74	0.07	0.08
J302_5	0.50	0.60	0.53	0.60	0.06	0.13	0.06	0.11
J302_6	0.54	0.00	0.50	0.52	0.06	0.09	0.18	0.06
J302_7	0.50	0.50	0.50	0.50	0.13	0.19	0.11	0.04
J302_8	0.50	0.50	0.57	0.50	0.06	0.06	0.09	0.04
J303_5	0.67	0.80	0.75	0.71	0.05	0.07	0.06	0.08
J303_6	0.71	0.75	0.67	0.73	0.07	0.21	0.08	0.11
J303_7	0.00	0.80	0.73	0.70	0.17	0.07	0.16	0.13
J303_8	0.70	1.00	0.67	0.67	0.08	0.04	0.07	0.07
J303_9	0.75	0.73	0.67	0.75	0.08	0.26	0.04	0.03
J601_4	0.19	0.20	0.23	0.22	0.03	0.04	0.04	0.02
J602_1	0.50	0.50	0.56	0.52	0.08	0.02	0.04	0.03
J602_2	0.53	0.51	0.50	0.50	0.02	0.03	0.05	0.05
J602_10	0.50	0.50	0.50	0.54	0.04	0.05	0.02	0.05

Πίνακας 11. Απόσπασμα πειραματικών αποτελεσμάτων προβλημάτων RCPSP/max των συνόλων UBO10&20

Όνομα αρχείου	Ελάχιστη Διάρκεια	Μέγιστη Διάρκεια	Μέση Διάρκεια	Κάτω όριο Διάρκειας LB	Άνω όριο Διάρκειας UB	Μέση απόκλιση από LB	Μέση απόκλιση από UB
UBO10_01	inf	inf	inf	inf	inf	0.00%	0.00%
UBO10_02	43	46	44	45	45	-0.22%	0.00%
UBO10_06	50	54	51	inf	inf	-	0.00%
UBO10_07	55	60	57	58	58	-1.38%	-1.38%
UBO10_13	39	44	42	45	45	-6.22%	-6.22%
UBO10_15	40	41	40	inf	inf	-	-
UBO10_25	54	55	54	55	55	-1.45%	-1.45%
UBO10_30	37	42	39	39	39	1.79%	1.79%
UBO10_31	50	52	51	51	51	0.00%	0.00%
UBO10_32	41	47	43	42	42	2.86%	2.86%
UBO10_33	46	52	49	50	50	-1.40%	-1.40%
UBO10_34	48	50	48	41	50	19.02%	-2.40%
UBO10_35	91	99	94	inf	inf	-	-
UBO10_36	45	50	47	58	58	-17.41%	-17.41%
UBO10_37	47	53	50	inf	inf	-	-
UBO10_39	86	90	88	96	96	-8.33%	-8.33%
UBO10_40	46	46	46	47	47	-2.13%	-2.13%
UBO10_41	34	38	35	38	39	-5.53%	-7.95%
UBO10_42	52	59	56	59	59	-4.75%	-4.75%
UBO20_01	94	107	100	103	103	-2.43%	-2.43%
UBO20_02	92	100	95	inf	inf	-	-
UBO20_05	78	84	81	inf	inf	-	-
UBO20_07	inf	inf	inf	inf	inf	0.00%	0.00%
UBO20_12	111	115	113	126	126	-10.12%	-10.12%
UBO20_14	80	85	83	90	90	-7.78%	-7.78%
UBO20_16	77	78	77	77	77	0.49%	0.49%
UBO20_18	59	62	60	66	66	-8.90%	-8.90%

Στα πειραματικά αποτελέσματα σκιαγραφείται η αποτελεσματικότητα του προτεινόμενου αλγορίθμου και η αποδοτικότητά του σε επίλυση τόσο RCPSP όσο και RCPSP/max προβλημάτων. Οι λύσεις που παρέχονται από την προτεινόμενη διαδικασία επίλυσης καθίστανται καλύτερες από τις υπάρχουσες βέλτιστες, ή τα κάτω όρια, σε αρκετά παραδείγματα και μικρής απόκλισης από αυτές, όταν δεν παράγει καλύτερες ή βέλτιστες λύσεις. Σημειώνεται ότι σε παραδείγματα RCPSP/max που ως τώρα στη διεθνή βιβλιογραφία θεωρούνταν άλυτα, ο προτεινόμενος αλγόριθμος παρέχει λύσεις, γεγονός που υποδηλώνει τη δυναμική του.

6.6 Πειραματικά Αποτελέσματα Παραδειγμάτων υπό Μελέτη Προβλήματος

Δεδομένης της πρόσφατης εισόδου του αντικειμένου των ευέλικτων προφίλ πόρων στο επιστημονικό προσκήνιο του προγραμματισμού έργων, καθώς και της καινοτομικής σκοπιάς διαχείρισης των εύλικτων προφίλ από το προτεινόμενο μοντέλο αναφορικά με τις ελάχιστες υπάρχουσες ερευνητικές μελέτες (Ranjbar and Kianfar, Naber and Kolisch, 2014b, Tritschler et al., 2014), σχεδιάστηκαν και επιλύθηκαν δύο παραδείγματα του υπό μελέτη προβλήματος. Σημειώνεται ότι τα δεδομένα του κάθε παραδείγματος για την έναρξη της διαδικασίας επίλυσης παρέχονται υπό τη μορφή του αρχείου εισόδου του υπό μελέτη προβλήματος (Παράρτημα Β).

Παράδειγμα 1.

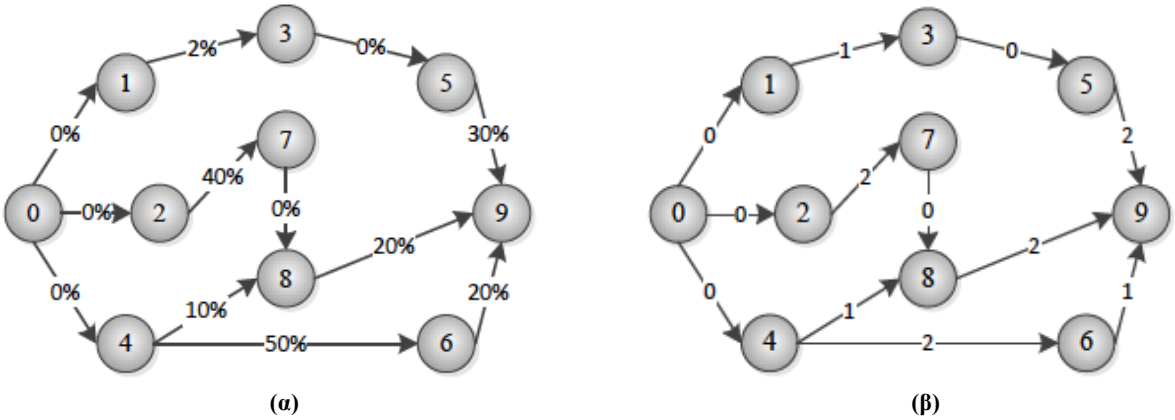
Το πρώτο παράδειγμα, τα δεδομένα του οποίου συνοψίζονται στον Πίνακα 12 αφορά οχτώ δραστηριότητες με έναν ανανεώσιμο πόρο, η συνολική διαθεσιμότητα του οποίου ανέρχεται στις οχτώ μονάδες ανά χρονική περίοδο. Η μέγιστη απαιτούμενη προσπάθεια του πόρου έγκειται στις εικοσιτέσσερις μονάδες και αφορά τη δραστηριότητα πέντε. Δεδομένου του ότι η ελάχιστη χρησιμοποιούμενη ποσότητα του πόρου είναι δύο μονάδες ανά χρονική περίοδο και η παραγωγική δυναμικότητά του μία μονάδα, η διάρκεια της δραστηριότητας πέντε κυμάινεται από 3 ως 8 χρονικές μονάδες.

Πίνακας 12. Δεδομένα Παραδείγματος 1 υπό μελέτη προβλήματος

ID Δραστ/τας	Απαιτούμενη Προσπάθεια από πόρο	Διάδοχοι	Χρονικές καθυστερήσεις (ποσοστό)	Ελάχιστο διάστημα σταθερής ποσότητας
0	0	1,2,4	0,0,0	0
1	4	3	0.02	1
2	21	7	0.4	2
3	12	5	0	2
4	16	6,8	0.5, 0.1	2
5	24	9	0.3	2
6	12	9	0.2	1
7	12	8	0	1
8	8	9	0.2	3
9	0	-	-	0

#Πόρου	Διαθέσιμότητα	Ελάχιστη χρησιμοποιούμενη ποσότητα ανά περίοδο	Μέγιστη χρησιμοποιούμενη ποσότητα ανά περίοδο	Παραγωγική Δυναμικότητα
1	8	2	8	1
Μέγιστος επιτρεπόμενος αριθμός διαμερίσεων:			2	

Στα Σχήματα 30(α)&(β) παρουσιάζονται οι δίγραφοι με τις χρονικές καθυστερήσεις. Στο Σχήμα 30(α) οι χρονικές καθυστερήσεις εμφανίζονται ως ποσοστά των αντίστοιχων διαρκειών των δραστηριοτήτων που αφορούν, ενώ στο Σχήμα 30(β) φαίνονται οι τελικές τιμές που λαμβάνουν οι χρονικές καθυστερήσεις, ύστερα από την επίλυση του παραδείγματος μέσα από τον προτεινόμενο αλγόριθμο και τον καθορισμό των διαρκειών των δραστηριοτήτων. Σημειώνεται ότι η τιμή κάθε χρονικής καθυστέρησης υπολογίζεται ως το γινόμενο του αντίστοιχου ποσοστού επί τη διάρκεια της δραστηριότητας που αφορά, δηλαδή του κόμβου από όπου ξεκινά το τόξο στο δίγραφο και καθορίζει τη σχέση προτεραιότητας.



(α) **(β)**

Σχήμα 30. Αρχικός (α) και τελικός (β) δίγραφος Παραδείγματος 1 υπό μελέτη προβλήματος

Στον Πίνακα 13 απεικονίζονται τα αποτελέσματα της διαδικασίας επίλυσης, τα οποία αφορούν τις χρονικές στιγμές έναρξης, τις διάρκειες και τα προφίλ εκτέλεσης κάθε δραστηριότητας. Τα προφίλ εκτέλεσης δίδονται ανά διαστήματα χρονικών περιόδων με την αντίστοιχη χρησιμοποιούμενη ποσότητα, εντός του διαστήματος, του πόρου από τη δραστηριότητα που αφορά. Η καλύτερη ευρεθείσα συνολική διάρκεια του έργου για το Παράδειγμα 1 ανέρχεται στις δεκατέσσερις χρονικές μονάδες.

Πίνακας 13. Αποτελέσματα Παραδείγματος 1 υπό μελέτη προβλήματος

Δραστηριότητα	Χρονικές στιγμές έναρξης	Διάρκεια	Πόρος 1	
			Χρονικές Περίοδοι	Πόσότητα
0	0	0	0	0
1	0	2	0-1	2
2	0	4	0-2 2-3	6 3
3	8	3	8-10	4
4	4	4	4-8	4
5	8	6	8-13	4
6	11	3	11-13	4
7	2	6	2-7	2

8	4	4	4-7	2
9	14	0	14	0

Παράδειγμα 2.

Το δεύτερο παράδειγμα, τα δεδομένα του οποίου συνοψίζονται στον Πίνακα 14, αφορά πέντε ανανεώσιμους πόρους και δέκα δραστηριότητες, με ελάχιστες και μέγιστες χρονικές καθυστερήσεις μεταξύ των τελευταίων.

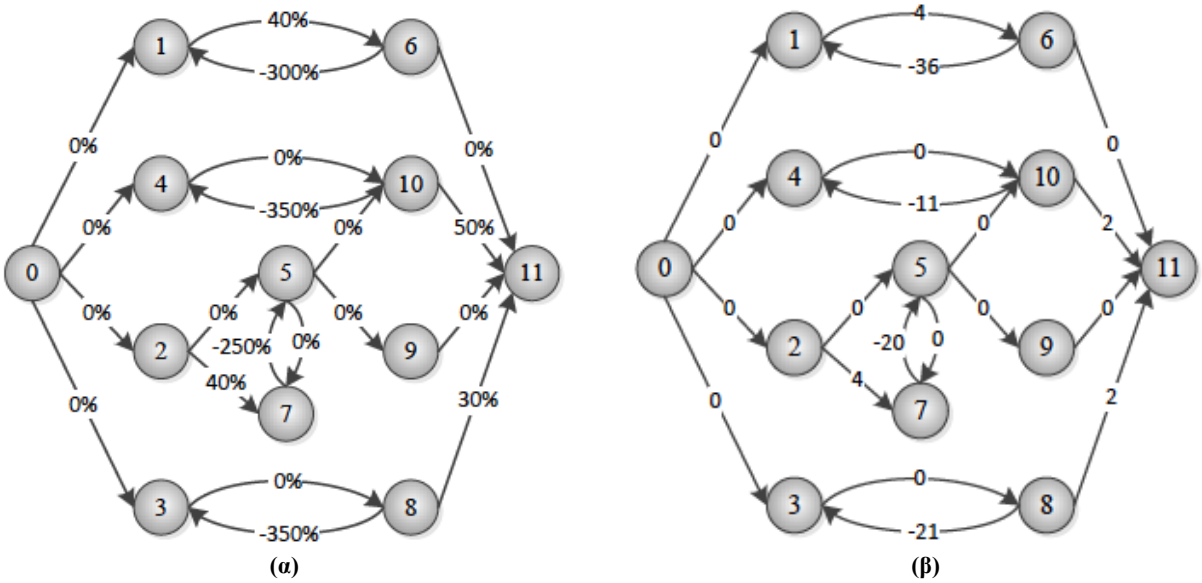
Πίνακας 14. Δεδομένα Παραδείγματος 2 υπό μελέτη προβλήματος

ID	Απαιτούμενη προσπάθεια					Διάδοχοι	Χρονικές καθυστε-ρήσεις (ποσοστό)	Ελάχιστο διάστημα σταθερής ποσότητα
	Πόρος 1	Πόρος 2	Πόρος 3	Πόρος 4	Πόρος 5			
0	0	0	0	0	0	1,2,3,4	0,0,0,0	0
1	60	30	28	20	60	6	0.4	1
2	12	21	10	24	16	7,5	0.4, 0	2
3	30	45	40	52	45	8	0	2
4	18	0	18	0	30	10	0	2
5	0	12	0	0	8	7,10,9	0, 0.4, 0	1
6	70	0	70	0	90	11,1	0, -3	2
7	0	9	16	16	8	5,9	-2.5, 0	1
8	56	81	48	20	24	3,11	-4, 0.3	2
9	12	21	0	6	12	11	0	2
10	0	18	0	0	36	4,11	-3.5, 0.5	2
11	0	0	0	0	0	-	-	0

#Πόρου	Διαθέσιμότητα	Ελάχιστη χρησιμοποιούμε-νη ποσότητα ανά περίοδο	Μέγιστη χρησιμοποιούμενη ποσότητα ανά περίοδο	Παραγωγική Δυναμικότητα
1	16	2	16	2
2	18	2	18	3
3	20	2	20	2
4	15	2	15	1
5	20	2	20	1
Μέγιστος επιτρεπόμενος αριθμός διαμερίσεων:			1	

Στα Σχήματα 31 (α)&(β) παρουσιάζονται οι δίγραφοι με τις χρονικές καθυστερήσεις. Στο Σχήμα 31(α) οι χρονικές καθυστερήσεις εμφανίζονται ως ποσοστά των αντίστοιχων διαρκειών των δραστηριοτήτων που αφορούν, ενώ στο Σχήμα 31(β) φαίνονται οι τελικές τιμές που λαμβάνουν οι χρονικές καθυστερήσεις, ύστερα από την επίλυση του παραδείγματος από τον προτεινόμενο αλγόριθμο και τον καθορισμό των διαρκειών των δραστηριοτήτων. Χαρακτηριστικά αναφέρεται η τιμή της μέγιστης χρονικής καθυστέρησης μεταξύ των δραστηριοτήτων τρία και οχτώ στον αρχικό δίγραφο, όπου λαμβάνει την τιμή -350%. Αυτή η τιμή υποδηλώνει ότι η βραδύτερη χρονική στιγμή έναρξης της δραστηριότητας οχτώ είναι (3.5 * διάρκεια της δραστηριότητας οχτώ) χρονικές μονάδες ύστερα από την έναρξη της δραστηριότητας τρία. Το γεγονός , αυτό σκιαγραφείται και στην τιμή που λαμβάνει η χρονική καθυστέρηση στον τελικό δίγραφο, -21, όπου η διάρκεια της δραστηριότητας οχτώ ανέρχεται σε 6 χρονικές μονάδες, καθώς και στο χρονοπρογραμματισμό των δύο δραστηριοτήτων. Πιο συγκεκριμένα, η έναρξη της δραστηριότητας τρία πραγματοποιείται την τρίτη χρονική

περίοδο, ενώ της δραστηριότητας οχτώ την όγδοη, ικανοποιώντας τον περιορισμό της μέγιστης χρονικής καθυστέρησης των 21 χρονικών μονάδων.



Σχήμα 31. Αρχικός (α) και τελικός (β) δίγραφος Παραδείγματος 2 υπό μελέτη προβλήματος

Στον Πίνακα 15 απεικονίζονται τα αποτελέσματα της διαδικασίας επίλυσης, τα οποία αφορούν τις χρονικές στιγμές έναρξης, τις διάρκειες και τα προφίλ εκτέλεσης κάθε δραστηριότητας. Τα προφίλ εκτέλεσης δίδονται ανά διαστήματα χρονικών περιόδων με την αντίστοιχη χρησιμοποιούμενη ποσότητα, εντός του διαστήματος, του πόρου από τη δραστηριότητα που αφορά. Η καλύτερη ευρεθείσα συνολική διάρκεια του έργου για το Παράδειγμα 2 ανέρχεται στις δεκαοχτώ χρονικές μονάδες.

Πίνακας 15. Αποτελέσματα Παραδείγματος 2 υπό μελέτη προβλήματος

ID	Χρόνοι έναρξης	Διάρκεια	Πόρος 1		Πόρος 2		Πόρος 3		Πόρος 4		Πόρος 5	
			Χρον. Περίοδοι	Ποσότητα	Χρον. Περίοδοι	Ποσότητα	Χρον. Περίοδοι	Ποσότητα	Χρον. Περίοδοι	Ποσότητα	Χρον. Περίοδοι	Ποσότητα
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	10	0-4	6	0-4	2	0-6	2	0-9	2	0-3	15
2	0	8	0-2	3	0	7	0	5	0-2	8	0-7	20-
3	0	15	0-2	5	0-4	3	0-1	10	0-12	4	0-14	3
4	4	5	4	9	4	0	4	9	4	0	4-8	6
5	4	2	4	0	4-5	2	4	0	4	0	4	8
6	5	12	5-15 16	3 2	5	0	5-9	7	5	0	5-14	9
7	9	8	9	0	9	3	9-10	4	9-16	2	9	8
8	10	6	10-11 12	9 10	10 11	10 17	10-13	6	10-14	4	10-15	4
9	10	3	10-12	2	10	7	10	0	10-12	2	10-12	4
10	15	3	15	0	15	6	15	0	15	0	15-17	12
11	18	0	18	0	18	0	18	0	18	0	18	0

7

Συμπεράσματα & Κατευθύνσεις Μελλοντικής Έρευνας

Ο χρονοπρογραμματισμός έργων στοχεύει στη δημιουργία ενός εφικτού χρονοπρογράμματος, που ελαχιστοποιεί το κριτήριο απόδοσης του προγραμματισμού. Το παραχθέν χρονοπρόγραμμα, αποτελεί το πρόγραμμα αναφοράς, βάσει του οποίου θα εκτελεστεί το έργο, θα κατανεμηθούν οι πόροι, θα προγραμματιστούν εξωτερικές και εσωτερικές δραστηριότητες της εφοδιαστικής αλυσίδας του οργανισμού που αναλαμβάνει το έργο, θα γίνει η σύναψη των απαραίτητων συμβάσεων και θα καθοριστούν τα χρονικά όρια παράδοσης και προθεσμιών των αποτελεσμάτων του έργου (Herroelen and Leus, 2005).

Ένας διαχειριστής έργου, κατά το χρονοπρογραμματισμό, καλείται να διαχειριστεί τους περιορισμούς σε πόρους, τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, αυτές καθ' αυτές τις δραστηριότητες του έργου με εκτίμηση της διάρκειας εκτέλεσης της κάθε μίας αναφορικά με την αντίστοιχη απαιτούμενη ποσότητα πόρων για την εκτέλεσή της, τις χρονικές προθεσμίες και τυχόν εσωτερικές απαραίτητες χρονικές καθυστερήσεις κατά τα στάδια εκτέλεσης των δραστηριοτήτων του έργου λόγω χρονισμού με εξωτερικά συμβάντα (π.χ. χρόνος παραλαβής πρώτων υλών). Αβίαστα, η υψηλή πολυπλοκότητα μοντελοποίησης της πολύπλευρης πραγματικότητας και επίλυσης των προβλημάτων προγραμματισμού έργων, σκιαγραφεί την επιτακτική ανάγκη ύπαρξης υπολογιστικών διαδικασιών, όπου με την ελάχιστη απαιτούμενη προσπάθεια από τη σκοπιά του διαχειριστή του έργου θα επιφέρουν το επιθυμητό αποτέλεσμα. Οι συνέπειες ενός μη σωστά σχεδιασμένου χρονοπρογράμματος μπορούν να θέσουν σε κίνδυνο την επιτυχημένη εκτέλεση και ολοκλήρωση του έργου.

Στην παρούσα διπλωματική εργασία προσδιορίστηκε ένα νέο μαθηματικό μοντέλο, το οποίο διαχειρίζεται προβλήματα προγραμματισμού έργων υπό περιορισμένους ανανεώσιμους πόρους με ή χωρίς γενικευμένες σχέσεις προτεραιότητας με ελάχιστες και μέγιστες χρονικές καθυστερήσεις, μέσα από την παραγωγή ευέλικτων προφίλ πόρων που ικανοποιούν την προκαθορισμένη απαιτούμενη προσπάθεια κάθε δραστηριότητας του έργου (*Effort Driven Resource Constrained Project Scheduling with Flexible Resource Profiles and Generilized Precedence Relations with minimal and maximal Time Lags*). Η κεντρική ιδέα συνοψίζεται στη δυνατότητα υπολογισμού της κατανομής των πόρων που ελαχιστοποιεί τη συνολική διάρκεια του έργου, με δυνατότητα παραγωγής ευέλικτων προφίλ εκτέλεσης και ταυτόχρονο σεβασμό στις απαιτήσεις και στους περιορισμούς που αφορούν τις προτεραιότητες εκτέλεσης των δραστηριοτήτων, το σύνολο των διαθέσιμων πόρων και την αναμενόμη απαιτούμενη προσπάθεια ανά δραστηριότητα. Εν συνεχεία, αναπτύχθηκε ένας εξελικτικός αλγόριθμος, ο οποίος αποτελεί μια μεταευρετική υπολογιστική διαδικασία βελτιστοποίησης, που βασίζεται στη συνδυαστική εξελικτική δυνατότητα δύο εμφωλευμένων γενετικών αλγορίθμων. Οι

εκτενείς πειραματικές διαδικασίες που πραγματοποιήθηκαν, απέδειξαν την αποδοτικότητα και την αποτελεσμάτικοτητά του προτεινόμενου αλγορίθμου επίλυσης, παρότι σημειώθηκε η ανάγκη επιπρόσθετων προσπαθειών με στόχο την εύρεση των τιμών των παραμέτρων που βελτιστοποιούν την απόδοτικότητά του. Σημειώνεται ότι μέσα από τα πειραματικά αποτελέσματα σκιαγραφείται η δυναμική και οι προοπτικές που διακατέχουν το προτεινόμενο μοντέλο, το οποίο επιλύει αποτελεσματικά όλα τα προβλήματα προγραμματισμού έργων που δύναται να διαχειριστεί, με δυνατότητα επίλυσης ακόμα και άλυτων, ως τώρα, προβλημάτων.

Η πρόσφατη είσοδος του αντικειμένου των ευέλικτων προφίλ πόρων στο επιστημονικό και ερευνητικό χώρο του προγραμματισμού έργων, αιτιολογεί το περιορισμένο πλήθος επιστημονικών δημοσιεύσεων και πειραματικών αποτελεσμάτων επίλυσης προβλημάτων προγραμματισμού έργων με περιορισμένους πόρους μέσω ευέλικτων προφίλ εκτέλεσης. Η καινοτομική σκοπιά διαχείρισης των ευέλικτων προφίλ από το προτεινόμενο μοντέλο, δίδει τη δυνατότητα ευελιξίας όχι μόνο στα προφίλ των πόρων, αλλά και στους αντικειμενικούς στόχους του έργου, μέσω του καθορισμού των τιμών των παραμέτρων που το χαρακτηρίζουν. Επιπρόσθετα, απαλλάσσει το διαχειριστή από την επίπονη και επισφαλή διαδικασία εκτίμησης της διάρκειας εκτέλεσης κάθε δραστηριότητας, αναφορικά με την αντίστοιχη απαιτούμενη κατανομή πόρων για την εκτέλεσή της, που ελλοχεύει κινδύνους που ενδέχεται να επηρεάσουν ακόμα και την αποτελεσματική ολοκλήρωση του έργου. Η διαδικασία επίλυσης, τέλος, η οποία είναι χρονικά σύντομη, επιτρέπει επανεκτελέσεις της για διάφορα πιθανά σενάρια, δίδοντας τη δυνατότητα στο διαχειριστή έργου να έχει στη διάθεσή του γρήγορα και αποτελεσματικά ένα εφικτό, αν όχι το βέλτιστο, χρονοπρόγραμμα για το σύνολο των προς εκτέλεση δραστηριοτήτων, σε συνδυασμό με τα προφίλ εκτέλεσης κάθε μιας εξ' αυτών.

Το επόμενο βήμα της παρούσας ερευνητικής προσπάθειας έγκειται τόσο στη βελτιστοποίηση του προτεινόμενου αλγορίθμου, όσο και στην ενίσχυσή του με άλλων ειδών μετα-ευρετικούς αλγορίθμους με απώτερο στόχο τη δημιουργία υβριδικών αλγορίθμων επίλυσης μέσω ευέλικτων προφίλ πόρων. Κάθε είδος εξελικτικού αλγορίθμου έχει διαφορετικά πλεονεκτήματα και μειονεκτήματα και ο συνδυασμός τους πιθανά να δώσει ακόμη καλύτερα αποτελέσματα, πιο σύντομα και αποδοτικά. Επίσης, απαιτούνται πρόσθετες πειραματικές διαδικασίες για την ανάλυση του τρόπου παραγωγής προφίλ εκτέλεσης πόρων, που θα βελτιστοποιήσουν τη διαδικασία επίλυσης. Χαρακτηριστικά αναφέρεται ότι η εύρεση και ο συνδυασμός των «καλών» προφίλ εκτέλεσης ανά δραστηριότητα κατά την υλοποίηση της εξελικτικής διαδικασίας του προτεινόμενου αλγορίθμου, ενδεχομένως να αποτελούν ένα κρίσιμο παράγοντα που να οδηγεί στη βελτιστοποίηση του προτεινόμενου μοντέλου και της διαδικασίας επίλυσης αυτού.

Τέλος, σημειώνεται η επιτακτική ανάγκη μελέτης συνδυασμού των διαφόρων μεθόδων και διαδικασιών επίλυσης προβλημάτων προγραμματισμού έργων, με απώτερο στόχο τη δημιουργία ολιστικών μεθόδων ικανών να αντιμετωπίσουν ρεαλιστικά προβλήματα προγραμματισμού έργων, ανεξαρτήτου φύσης και χαρακτηριστικών, μέσα από διαδικασίες ανάδρασης και βελτιστοποίησης. Τέτοιες ενέργειες έχουν αρχίσει να πραγματοποιούνται, με εφαρμογή ολιστικών μαθηματικών μοντέλων πολυστοχικού προγραμματισμού και χρήση υβριδικών αλγορίθμων επίλυσης (Rokou, 2013). Πάραυτα, υπάρχουν πολλά αντικείμενα που χρήζουν έρευνας, ώστε να επιτευχθεί μια αποτελεσματική σύζευξη μεταξύ των ποικίλων μοντέλων και μεθόδων επίλυσης που παρέχονται από την επιστημονική κοινότητα και των πρακτικών προβλημάτων που αντιμετωπίζουν οι διαχειριστές έργων, χωρίς συμβιβασμούς, υποθέσεις και απλοποιήσεις.

Βιβλιογραφία

AGIN, N. 1966. Optimum seeking with branch and bound. *Management Science,* 13**,** B-176-B-185.

ALLEN, T. J., LEE, D. & TUSHMAN, M. L. 1980. R&D performance as a function of internal communication, project management, and the nature of the work. *Engineering Management, IEEE Transactions on***,** 2-12.

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.

ALVAREZ-VALDES, R. & TAMARIT, J. M. 1993. The project scheduling polyhedron: dimension, facets and lifting theorems. *European Journal of Operational Research,* 67**,** 204-220.

ATKINSON, R. 1999. Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management,* 17**,** 337-342.

BALLESTÍN, F., BARRIOS, A. & VALLS, V. 2011. An evolutionary algorithm for the resource-constrained project scheduling problem with minimum and maximum time lags. *Journal of Scheduling,* 14**,** 391-406.

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.

BEY, R. B., DOERSCH, R. H. & PATTERSON, J. H. 1981. *The net present value criterion: its impact on project scheduling*.

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

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

BOCTOR, F. F. 1996. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research,* 34**,** 2335-2351.

BÖTTCHER, J., DREXL, A., KOLISCH, R. & SALEWSKI, F. 1999. Project scheduling under partially renewable resource constraints. *Management Science,* 45**,** 543-559.

BOUFFARD, V. & FERLAND, J. A. 2007. Improving simulated annealing with variable neighborhood search to solve the resource-constrained scheduling problem. *Journal of Scheduling,* 10**,** 375-386.

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. Complex scheduling problems. Zeitschrift Oper. Res, 1999. Citeseer.

BRUCKER, P., DREXL, A., MOHRING, R., NEUMANN, K. & PERSCH, E. 1998. Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. *European Journal of Operations Research*.

BRUCKER, P. & KNUST, S. 2012. Resource-Constrained Project Scheduling. *Complex Scheduling.* Springer Berlin Heidelberg.

CHO, J. H. & KIM, Y. D. 1997. A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society***,** 736-744.

CHRISTOFIDES, N., ALVAREZ-VALDÉS, 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., POLAKOV, W. N. & TRABOLD, F. W. 1922. *The Gantt chart: A working tool of management*, Ronald Press Company.

COOKE-DAVIES, T. 2002. The "real" success factors on projects. *International Journal of Project Management,* 20**,** 185-190.

DAVIS, E. W. 1973. Project scheduling under resource constraints—historical review and categorization of procedures. *AIIE 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.

DE REYCK, B. & HERROELEN, W. 1999. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research,* 119**,** 538-556.

DE VILDER, D. 2004. *Project management T-kit*, Council of Europe.

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. 2000. The discrete time/resource trade-off problem in project networks: a branch-and-bound approach. *IIE transactions,* 32**,** 1059-1069.

DEMEULEMEESTER, E. & HERROELEN, W. 2002. Project scheduling-A research handbook. Vol. 49 of International Series in Operations Research & Management Science. Kluwer Academic Publishers, Boston.

DU, D.-Z. & KO, K.-I. 2011. *Theory of computational complexity*, John Wiley & Sons.

EGLESE, R. W. 1990. Simulated annealing: a tool for operational research. *European journal of operational research,* 46**,** 271-281.

EISENHARDT, K. M. & TABRIZI, B. N. 1995. Accelerating adaptive processes: Product innovation in the global computer industry. *Administrative science quarterly***,** 84-110.

ELMAGHRABY, S. E. 1964. An algebra for the analysis of generalized activity networks. *Management Science,* 10**,** 494-514.

F¼NDELING, C. U. & TRAUTMANN, N. 2010. A priority-rule method for project scheduling with work-content constraints. *European Journal of Operational Research,* 203**,** 568-574.

GAREY, M. R. & JOHNSON, D. S. 1979. Computers and Intractability. *A Guide to the Theory of NP-Completeness*.

GLOVER, F. 1989. Tabu search-part I. *ORSA Journal on computing,* 1**,** 190-206.

GLOVER, F. 1990. Tabu search—part II. *ORSA Journal on computing,* 2**,** 4-32.

GLOVER, F. & LAGUNA, M. 1997. Tabu search, 1997. *Kluwer Academic Publishers*.

GOLDBERG, D. E. 2002. *The design of innovation: Lessons from and for competent genetic algorithms*, Kluwer Academic Publishers.

GOLDRATT, E. M. 1997. Critical chain. North River Press: Great Barrington, MA.

GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K. & KAN, A. H. G. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete Mathematics,* 5**,** 287-326.

GUTIÉRREZ, E., LA TORRE, F. & MEJÍA, G. 2007. A genetic algorithm for the resource constrained project scheduling problem (RCPSP). Universidad Privada Boliviana.

HANS, E. W., HERROELEN, W., LEUS, R. & WULLINK, G. 2007. A hierarchical approach to multi-project planning under uncertainty. *Omega,* 35**,** 563-577.

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.

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. & LEUS, R. 2005. Project scheduling under uncertainty: Survey and research potentials. *European journal of operational research,* 165**,** 289-306.

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.

HOLLAND, J. H. 1975. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, U Michigan Press.

ICMELI-TUKEL, O. & ROM, W. O. 1997. Ensuring quality in resource constrained project scheduling. *European Journal of Operational Research,* 103**,** 483-496.

ICMELI, O. & ERENGUC, S. S. 1994. A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows. *Computers & Operations Research,* 21**,** 841-853.

ISO 2012. 21500:Guidance on project management. *ICS 03.100.40.* International Organization of Standardization.

KARP, R. 1975. On the computational complexity. *Networks,* 5**,** 45-68.

KELLEY JR, J. E. 1963. The critical path method: Resource planning and scheduling, Muth JF, Thompson GL, Industrial Scheduling, 1963, 347-365. Prentice-Hall, Englewood Cliffs, NJ.

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.

KIRKPATRICK, S., GELATT, C. D. & VECCHI, M. P. 1983. Optimization by simmulated annealing. *science,* 220**,** 671-680.

KIS, T. 2005. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. *Mathematical programming,* 103**,** 515-539.

KLEIN, R. 2000. Project scheduling with time-varying resource constraints. *International Journal of Production Research,* 38**,** 3937-3952.

KOLISCH, R. 1995. *Project scheduling under resource constraints: efficient heuristics for several problem classes*, Springer Verlag.

KOLISCH, R. 1996a. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management,* 14**,** 179-192.

KOLISCH, R. 1996b. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research,* 90**,** 320-333.

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.

KOLISCH, R., MEYER, K., MOHR, R., SCHWINDT, C. & URMANN, M. 2003. Ablaufplanung fur die Leitstrukturoptimierung in der Pharmaforschung. *Zeitschrift fur Betriebswirtschaft,* 73**,** 825-848.

KOLISCH, R. & PADMAN, R. 1997. An integrated perspective of project scheduling. *Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel,* 463.

KOLISCH, R. & PADMAN, R. 2001. An integrated survey of deterministic project scheduling. *Omega,* 29**,** 249-272.

KOLISCH, R. & SPRECHER, A. 1997. PSPLIB - A project scheduling problem library. *European Journal of Operational Research,* 96**,** 205-216.

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. 1995. Project planning. *Scheduling & Control: A hands-On guide to bringing projects in on time and on budget, McGrawHill*.

LUMSDEN, P. 1968. *The line-of-balance method*, Pergamon Press Limited.

MAYLOR, H. 2003. *Project Management,* London, Pitman.

MENDES, J. J. D. M., GONÇALVES, J. F. & RESENDE, M. G. C. 2009. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research,* 36**,** 92-109.

MIKA, M., WALIGORA, G. & WĘGLARZ, J. 2008. Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research,* 187**,** 1238-1250.

MINGOZZI, A., MANIEZZO, V., RICCIARDELLI, S. & BIANCO, L. 1998. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science,* 44**,** 714-729.

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.

MUNNS, A. K. & BJEIRMI, B. F. 1996. The role of project management in achieving project success. *International Journal of Project Management,* 14**,** 81-87.

N¼BEL, H. & SCHWINDT, C. 1997. *A classification of shifts, schedules, and objective functions in project scheduling*, Inst. für Wirtschaftstheorie und Operations-Research.

NABER, A. & KOLISCH, R. 2014a. A Continuous-Time Model for the Resource-Constrained Project Scheduling with Flexible Profiles. *Proceedings of the 14th International Conference on Project Management and Scheduling.* TUM School of Management.

NABER, A. & KOLISCH, R. 2014b. MIP Models for Resource-Constrained Project Scheduling with Flexible Resource Profiles. *European Journal of Operational Research*.

NEUMANN-BRAUN, K., SCHWINDT, C. & ZIMMERMANN, J. 2003. *Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions*, Springer.

NEUMANN, K. & SCHWINDT, C. 1997. Activity-on-node networks with minimal and maximal time lags and their application to make-to-order production. *Operations-Research-Spektrum,* 19**,** 205-217.

NONOBE, K. & IBARAKI, T. 2002. Formulation and tabu search algorithm for the resource constrained project scheduling problem. *Essays and surveys in metaheuristics.* Springer.

NONOBE, K. & IBARAKI, T. A tabu search algorithm for a generalized resource constrained project scheduling problem. MIC2003: The fifth metaheuristics international conference, 2003. 55-1.

PATTERSON, J. H., SLOWINSKI, R., TALBOT, F. B. & WEGLARZ, J. 1989. An algorithm for a general class of precedence and resource constrained scheduling problems. *Advances in project scheduling,* 187**,** 3-28.

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 2013. *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.

RANJBAR, M. & KIANFAR, F. Resource-Constrained Project Scheduling Problem with Flexible Work Pro les: A Genetic Algorithm Approach.

ROKOU, E. 2013. *Decision Making in Project Management: Multi Objective Extended Resource Constrained Project Scheduling.* National Technical University of Athens.

SAMPSON, S. E. & WEISS, E. N. 1993. Local search techniques for the generalized resource constrained project scheduling problem. *Naval Research Logistics (NRL),* 40**,** 665-675.

SHAFFER, L. R., RITTER, J. B. & MEYER, W. L. 1965. *The critical-path method*, McGraw-Hill.

SHMOYS, D. B. & TARDOS, É. 1993. An approximation algorithm for the generalized assignment problem. *Mathematical Programming,* 62**,** 461-474.

SPRECHER, A., HARTMANN, S. & DREXL, A. 1997. An exact algorithm for project scheduling with multiple modes. *Operations-Research-Spektrum,* 19**,** 195-203.

SPRECHER, A., KOLISCH, R. & DREXL, A. 1995. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research,* 80**,** 94-102.

STINSON, J. P., DAVIS, E. W. & KHUMAWALA, B. M. 1978. Multiple resource–constrained scheduling using branch and bound. *AIIE Transactions,* 10**,** 252-259.

SURI, R. 1998. Quick Response Manufacturing: A Company Wide Approach to Reducing Lead Times. *Portland, Productivity Press.*

TALBOT, F. B. 1982. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science,* 28**,** 1197-1210.

TALBOT, F. B. & PATTERSON, J. H. 1978. An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science,* 24**,** 1163-1174.

TRITSCHLER, M., NABER, A. & KOLISCH, R. 2014. A Genetic Algorithm for the Resource-Constrained Project Scheduling Problem with Flexible Resource Profiles. *Proceedings of the 14th International Conference on Project Management and Scheduling.* TUM School of Management.

WANG, H., LI, T. & LIN, D. 2010. Efficient genetic algorithm for resource-constrained project scheduling problem. *Transactions of Tianjin University,* 16**,** 376-382.

WIEST, J. D. 1963. *The scheduling of large projects with limited resources.* Ph.D. Thesis, Carnegie Institute of Technology.

Παραρτήματα

Α. Αποτελέσματα Εκτέλεσης Προτεινόμενου Αλγόριθμου

Πίνακας 16. Πειραματικά αποτελέσματα προβλημάτων RCPSP των συνόλων J30&60

Όνομα αρχείου	Βέλτιστη τιμή	Ελάχιστη Διάρκεια	Μέγιστη Διάρκεια	Μέση Διάρκεια	Μέση απόκλιση από βέλτιστη τιμή
J301_1	43	43	51	46	6.98%
J301_2	47	49	58	53	12.77%
J301_3	47	45	51	47	0.00%
J301_4	62	58	66	63	1.61%
J301_5	39	37	42	39	0.00%
J301_6	48	45	52	47	-2.08%
J301_7	60	64	75	69	15.00%
J301_8	53	54	58	56	5.66%
J301_9	49	49	55	53	8.16%
J301_10	45	43	49	46	2.22%
J302_1	38	39	43	41	7.89%
J302_2	51	42	50	46	-9.80%
J302_3	43	42	49	45	4.65%
J302_4	43	36	42	40	-6.98%
J302_5	51	47	52	49	-3.92%
J302_6	47	41	53	46	-2.13%
J302_7	47	40	47	43	-8.51%
J302_8	54	48	54	51	-5.56%
J302_9	54	50	61	56	3.70%
J302_10	43	43	49	46	6.98%
J303_1	72	63	75	68	-5.56%
J303_2	40	38	42	40	0.00%
J303_3	57	47	51	48	-15.79%
J303_4	98	103	118	111	13.27%
J303_5	53	49	55	50	-5.66%
J303_6	54	48	51	49	-9.26%
J303_7	48	34	39	36	-25.00%
J303_8	54	45	52	49	-9.26%
J303_9	65	51	54	52	-20.00%
J303_10	59	56	66	61	3.39%
J601_1	77	80	87	84	9.09%
J601_2	68	75	89	82	20.59%
J601_3	68	71	85	78	14.71%
J601_4	91	91	104	96	5.49%
J601_5	73	82	88	85	16.44%

J601_6	66	68	77	72	9.09%
J601_7	72	74	81	78	8.33%
J601_8	75	82	95	89	18.67%
J601_9	85	90	99	95	11.76%
J601_10	80	87	94	90	12.50%
J602_1	65	65	72	68	4.62%
J602_2	82	73	87	82	0.00%
J602_3	78	81	97	89	14.10%
J602_4	78	79	97	86	10.26%
J602_5	54	59	67	62	14.81%
J602_10	69	73	82	76	10.14%

Όνομα αρχείου	**$RS_{1,opt}$**	**$RS_{2,opt}$**	**$RS_{3,opt}$**	**$RS_{4,opt}$**	**$RS'_{1,avg}$**	**$RS'_{2,avg}$**	**$RS'_{3,avg}$**	**$RS'_{4,avg}$**
J301_1	0.18	0.20	0.00	0.21	0.05	0.03	1.14	0.07
J301_2	0.22	0.20	0.14	0.22	0.04	0.05	0.06	0.14
J301_3	0.33	0.00	0.19	0.22	0.05	0.06	0.08	0.11
J301_4	0.18	0.20	0.00	0.21	1.20	0.05	0.07	0.03
J301_5	0.19	0.20	0.18	0.20	0.14	0.02	0.11	0.11
J301_6	0.25	0.17	0.00	0.18	0.11	0.12	0.17	0.04
J301_7	0.00	0.21	0.25	0.25	0.12	0.04	0.06	0.07
J301_8	0.25	0.22	0.17	0.20	0.14	0.05	0.05	0.10
J301_9	0.21	0.18	0.20	0.21	0.04	0.10	0.14	0.09
J301_10	0.20	0.20	0.25	0.00	0.04	0.13	0.07	0.06
J302_1	1.00	0.50	0.50	0.55	0.09	0.05	0.09	0.05
J302_2	0.50	0.50	0.67	0.53	0.03	0.08	0.11	0.14
J302_3	0.57	0.56	0.57	0.50	0.15	0.05	0.08	0.14
J302_4	0.53	0.00	0.55	0.57	0.09	0.74	0.07	0.08
J302_5	0.50	0.60	0.53	0.60	0.06	0.13	0.06	0.11
J302_6	0.54	0.00	0.50	0.52	0.06	0.09	0.18	0.06
J302_7	0.50	0.50	0.50	0.50	0.13	0.19	0.11	0.04
J302_8	0.50	0.50	0.57	0.50	0.06	0.06	0.09	0.04
J302_9	0.57	0.50	0.50	0.54	0.44	0.05	0.06	0.04
J302_10	0.50	0.53	0.50	0.50	0.08	0.09	0.13	0.06
J303_1	0.75	0.67	0.71	0.67	0.24	0.13	0.04	0.04
J303_2	0.75	0.75	0.67	0.67	0.66	0.04	0.10	0.02
J303_3	0.71	0.67	0.75	0.73	0.18	0.11	0.06	0.06
J303_4	1.00	0.67	0.00	0.50	0.08	0.07	0.04	0.02
J303_5	0.67	0.80	0.75	0.71	0.05	0.07	0.06	0.08
J303_6	0.71	0.75	0.67	0.73	0.07	0.21	0.08	0.11
J303_7	0.00	0.80	0.73	0.70	0.17	0.07	0.16	0.13
J303_8	0.70	1.00	0.67	0.67	0.08	0.04	0.07	0.07
J303_9	0.75	0.73	0.67	0.75	0.08	0.26	0.04	0.03
J303_10	0.67	0.50	0.75	0.69	0.20	0.13	0.06	0.07
J601_1	0.18	0.2	0.19	0.21	0.03	0.03	0.03	0.04
J601_2	0.21	0.20	0.20	0.20	0.01	0.02	0.02	0.03
J601_3	0.21	0.21	0.22	0.17	0.06	0.02	0.03	0.04
J601_4	0.19	0.20	0.23	0.22	0.03	0.04	0.04	0.02
J601_5	0.18	0.20	0.21	0.22	0.05	0.04	0.03	0.02

J601_6	0.22	0.25	0.18	0.22	0.04	0.04	0.06	0.01
J601_7	0.23	0.17	0.21	0.21	0.02	0.02	0.04	0.03
J601_8	0.20	0.25	0.22	0.25	0.04	0.08	0.03	0.02
J601_9	0.23	0.20	0.22	0.20	0.03	0.02	0.07	0.02
J601_10	0.20	0.20	0.25	0.14	0.03	0.02	0.03	0.02
J602_1	0.50	0.50	0.56	0.52	0.08	0.02	0.04	0.03
J602_2	0.53	0.51	0.50	0.50	0.02	0.03	0.05	0.05
J602_3	0.50	0.50	0.50	0.56	0.07	0.05	0.01	0.02
J602_4	0.50	0.57	0.52	0.50	0.02	0.03	0.06	0.04
J602_5	0.50	0.50	0.52	0.52	0.03	0.03	0.09	0.04
J602_10	0.50	0.50	0.50	0.54	0.04	0.05	0.02	0.05

Πίνακας 17. Πειραματικά αποτελέσματα προβλημάτων RCPSP/max των συνόλων UBO10&20

Όνομα αρχείου	**Ελάχιστη Διάρκεια**	**Μέγιστη Διάρκεια**	**Μέση Διάρκεια**	**Κάτω όριο Διάρκειας LB**	**Άνω όριο Διάρκειας UB**	**Μέση απόκλιση από LB**	**Μέση απόκλιση από UB**
UBO10_01	inf	inf	inf	inf	inf	0.00%	0.00%
UBO10_02	43	46	44	45	45	-0.22%	0.00%
UBO10_03	39	44	42	41	41	3.66%	3.66%
UBO10_04	60	69	65	57	57	14.21%	14.21%
UBO10_05	43	48	45	43	43	6.05%	6.05%
UBO10_06	50	54	51	inf	inf	-	0.00%
UBO10_07	55	60	57	58	58	-1.38%	-1.38%
UBO10_08	inf	inf	inf	inf	inf	0.00%	0.00%
UBO10_09	39	42	41	35	37	17.43%	11.08%
UBO10_10	inf	inf	inf	inf	inf	0.00%	0.00%
UBO10_11	27	30	28	26	26	10.00%	10.00%
UBO10_12	45	52	48	45	45	8.44%	8.44%
UBO10_13	39	44	42	45	45	-6.22%	-6.22%
UBO10_14	41	46	43	41	41	6.59%	6.59%
UBO10_15	40	41	40	inf	inf	-	-
UBO10_16	26	34	31	25	28	26.00%	12.50%
UBO10_17	61	68	65	68	68	-4.26%	-4.26%
UBO10_18	42	45	43	42	45	2.62%	-4.22%
UBO10_19	31	35	33	38	38	-12.11%	-12.11%
UBO10_20	63	66	64	68	68	-5.15%	-5.15%
UBO10_21	52	58	55	51	51	8.43%	8.43%
UBO10_22	30	33	31	31	31	1.29%	1.29%
UBO10_23	32	37	35	31	32	14.19%	10.63%
UBO10_24	42	48	44	40	40	12.00%	12.00%
UBO10_25	54	55	54	55	55	-1.45%	-1.45%
UBO10_26	34	40	37	33	34	12.42%	9.12%
UBO10_27	inf	inf	inf	inf	inf	-	-
UBO10_28	46	51	48	47	47	2.77%	2.77%
UBO10_29	36	42	38	32	33	21.56%	17.88%
UBO10_30	37	42	39	39	39	1.79%	1.79%

UBO10_31	50	52	51	51	51	0.00%	0.00%
UBO10_32	41	47	43	42	42	2.86%	2.86%
UBO10_33	46	52	49	50	50	-1.40%	-1.40%
UBO10_34	48	50	48	41	50	19.02%	-2.40%
UBO10_35	91	99	94	inf	inf	-	-
UBO10_36	45	50	47	58	58	-17.41%	-17.41%
UBO10_37	47	53	50	inf	inf	-	-
UBO10_38	59	66	63	52	57	21.35%	10.70%
UBO10_39	86	90	88	96	96	-8.33%	-8.33%
UBO10_40	46	46	46	47	47	-2.13%	-2.13%
UBO10_41	34	38	35	38	39	-5.53%	-7.95%
UBO10_42	52	59	56	59	59	-4.75%	-4.75%
UBO10_43	39	44	40	39	40	4.87%	2.25%
UBO10_44	37	40	39	38	38	2.89%	2.89%
UBO10_45	71	78	73	73	73	0.96%	0.96%
UBO10_46	42	46	44	48	48	-7.92%	-7.92%
UBO10_47	24	26	25	26	27	-3.08%	-6.67%
UBO10_48	24	31	27	27	27	2.59%	2.59%
UBO10_49	49	58	55	50	59	11.00%	-5.93%
UBO10_50	inf	inf	inf	inf	inf	0.00%	0.00%
UBO20_01	94	107	100	103	103	-2.43%	-2.43%
UBO20_02	92	100	95	inf	inf	-	-
UBO20_03	116	122	118	inf	inf	-	-
UBO20_04	98	108	104	83	98	25.45%	6.25%
UBO20_05	78	84	81	inf	inf	-	-
UBO20_07	inf	inf	inf	inf	inf	0.00%	0.00%
UBO20_08	100	109	104	87	93	20.26%	12.50%
UBO20_09	inf	inf	inf	inf	inf	0.00%	0.00%
UBO20_10	92	107	99	85	106	17.25%	-5.97%
UBO20_11	102	107	105	100	100	5.67%	5.67%
UBO20_12	111	115	113	126	126	-10.12%	-10.12%
UBO20_13	95	97	96	91	92	5.91%	4.76%
UBO20_14	80	85	83	90	90	-7.78%	-7.78%
UBO20_16	77	78	77	77	77	0.49%	0.49%
UBO20_17	74	82	78	66	69	18.43%	13.29%
UBO20_18	59	62	60	66	66	-8.90%	-8.90%
UBO20_19	67	72	68	60	60	14.58%	14.58%
UBO20_20	65	73	68	57	66	20.18%	3.79%

Β. Μορφή Αρχείων Εισόδου υπό μελέτη Προβλήματος

n	ρ	q_{max}						
0	*1*	s_0	j_1^0	…	$j_{s_0}^0$	$[\delta_{0,j_1^0}]$	…	$[\delta_{0,j_{s_0}^0}]$
1	*1*	s_1	j_1^1	…	$j_{s_1}^1$	$[\delta_{1,j_1^1}]$	…	$[\delta_{1,j_{s_1}^1}]$
…								
n	*1*	s_n	j_1^n	…	$j_{s_n}^n$	$[\delta_{n,j_1^n}]$	…	$[\delta_{n,j_{s_n}^n}]$
n+1	*1*	*0*						
0	*1*	*0*	…	*0*	*0*			
1	*1*	$E_{1,1}$	…	$E_{1,\rho}$	$t_{alloc,min}^1$			
…								
n	*1*	$E_{n,1}$	…	$E_{n,\rho}$	$t_{alloc,min}^n$			
n+1	*1*	*0*	…	*0*	*0*			
R_1	…	R_ρ						
r_1^{min}	…	r_ρ^{min}						
r_1^{max}	…	r_ρ^{max}						
din_1	…	din_ρ						

Συμβολισμός

n	Αριθμός πραγματικών δραστηριοτήτων
ρ	Αριθμός ανανεώσιμων πόρων
q_{max}	Μέγιστος επιτρεπόμενος αριθμός διαμερίσεων
s_i	Αριθμός άμεσων διαδόχων του κόμβου *i* στο δίγραφο έργου
j_s^i	*s*-ος διάδοχος του κόμβου *i* στο δίγραφο έργου
δ_{i,j_s^i}	"Βάρος τόξου" *(i, j_s^i)*. Χρονική καθυστέρηση εκφρασμένη σε ποσοστό της μελλοντικά υπολογιζόμενης διάρκειας της δραστηριότητας *i* , δεκαδικής τιμή (π.χ. 0.4)
$E_{i,\rho}$	Απαιτούμενη προσπάθεια για την ολοκλήρωση της δραστηριότητας *i* από τον πόρο ρ
R_k	Διαθέσιμη ποσότητα πόρου k
$t_{alloc,min}^i$	Ελάχιστο χρονικό διάστημα σταθερής χρησιμοποιούμενης ποσότητας των πόρων από τη δραστηριότητα *i*
r_k^{min}	Ελάχιστη ανά περίοδο χρησιμοποιούμενη ποσότητα πόρου k
r_k^{max}	Μέγιστη ανά περίοδο χρησιμοποιούμενη ποσότητα πόρου k
din_k	Παραγωγική δυναμικότητα πόρου k

C. Πηγαίος Κώδικας

C.1. Εξωτερικός Γενετικός Αλγόριθμος

```
public class GA_Final {
     public static void main(String[] args) throws IOException,
CloneNotSupportedException {
         //Give type of solvable problem Code=1:rcpsp/max Code=2 MyInput
Code=3: rcpsp
         int code=0;
             do {
                 System.out.println("Give type of solvable problem
Code=1:rcpsp/max Code=2 MyInputFRCPSP/max Code=3: rcpsp");
                 InputStreamReader t10 = new InputStreamReader(System.in);
                 BufferedReader t20 = new BufferedReader(t10);
                 String t30= t20.readLine();
                 boolean positive=false;
                 try{
                     Integer.parseInt(t30);
                     positive =true;
                 }
                 catch(Exception e1){
                     System.out.println("Error while reading" +
e1.getMessage());
                     positive=false;
                 }
                 if (positive == true) {
                     code=Integer.parseInt(t30);
                 }
             }
             while (code!=1 && code !=2&& code!=3);
         System.out.println("Give file i.e. <rcp.txt>");
         //Read file according to problem type and initialize variables
rmax, rmin, tallocmin, qmax, din…

         //G.A.external
         //set POP size of chromosomes
         int POPchroms;
         boolean posEven;
         do {
             POPchroms=-1;
             System.out.println("Give POP size");
             InputStreamReader s = new InputStreamReader(System.in);
             BufferedReader size = new BufferedReader(s);
             String psize= size.readLine();
             boolean posNumber=false;
             posEven=false;
             try{
                 Integer.parseInt(psize);
                 posNumber =true;
             }
            catch(Exception e){
                System.out.println("Error while reading" + e.getMessage());
                posNumber=false;
             }
            if (posNumber == true) {
                 POPchroms=Integer.parseInt(psize);
             }
            if(POPchroms%2==0)
                posEven =true;
```

```
            else System.out.println("POP prepei na einai even");
        } while (POPchroms < 0 || posEven==false);
        //list with all POPchroms chromosomes, with ActList random for
now, resNum, jobBum
        ArrayList<Chromosome> chromosomes = new ArrayList();
        ArrayList<Activity> randomList;
        Activity newAct;
        for(int c=0;c<POPchroms;c++){
            // NEED DEEP CLONE OF list projectActivities
            randomList = new ArrayList();
            for(i=0;i<myProject.getJobNum();i++){
                newAct = new Activity(i);
                newAct.effort=myProject.projectActivities.get(i).effort;
                newAct.succLags=myProject.projectActivities.get(i)
.succLags;
                newAct.succLagsPercent=myProject.projectActivities.get(i)
.succLagsPercent;
                randomList.add(newAct);
            }
            Chromosome xromosoma = new
Chromosome(randomList,myProject.getResourceNum(),myProject.getJobNum());
            chromosomes.add(xromosoma);
        }
        //put random values in the partition table of each chrom--
>partition[][] with 0<= Qik <=qMax
        for(int c=0;c<POPchroms;c++){
            for(i=0;i<myProject.getJobNum();i++){
                for(int j=0;j<myProject.getResourceNum();j++){
                    Random rand = new Random();
                    int value = rand.nextInt(myProject.
getMaxNumPartition()+1) ;
                    chromosomes.get(c).partition[j][i]=value;
                }
            }
        }
        //Genes
        int evolutionStepsExternal;
        do {
            evolutionStepsExternal=-1;
            System.out.println("Give total number of genes of
GAexternal");
            InputStreamReader s1 = new InputStreamReader(System.in);
            BufferedReader steps = new BufferedReader(s1);
            String tsteps= steps.readLine();
            boolean positive=false;
            try{
                Integer.parseInt(tsteps);
                positive =true;

            }
           catch(Exception e1){
               System.out.println("Error while reading" +
e1.getMessage());
               positive=false;
            }
           if (positive == true) {
                evolutionStepsExternal=Integer.parseInt(tsteps);
            }
        } while (evolutionStepsExternal < 0);

        System.out.println("Computing Initial population and profile...");
        //INITIAL POPULATION compute profiles
```

```
        List<List<Integer>> prof ;
        List<Integer> poroiPerPeriod;
        int ginomena[][];
        boolean okProducts;
        int effort;
        int check=0;
        int minEff;
        double checkInt=0;
        for(int c=0;c<POPchroms;c++){
            //Set zero profile for i=0
            prof=new ArrayList();
            for(int j=0;j<myProject.getResourceNum();j++){
                poroiPerPeriod=new ArrayList();
                poroiPerPeriod.add(0);
                prof.add(poroiPerPeriod);
            }
            chromosomes.get(c).profiles.add(0, prof);
            //Compute profiles for i=1,...,n
            for(i=1;i<myProject.getJobNum()-1;i++){ //for each act
                //For i=0,n+1  zero profile
                prof =new ArrayList();
                for(int j=0;j<myProject.getResourceNum();j++){  //gia
kathe poro
                    effort=chromosomes.get(c).actsList.get(i).effort
.get(j);
                    //if an act doesn't use a resource, put zero profile
                    if(effort==0){
                        poroiPerPeriod=new ArrayList();
                        for(int u=0;u<chromosomes.get(c).partition[j][i]+1
;u++){
                            poroiPerPeriod.add(0);
                        }
                        prof.add(poroiPerPeriod);
                    }
                    else{//act i uses resource j-->compute profil
                        //if effort/resProductivity!=integer-->Stop wrong
inputs
                        checkInt=(double) effort/myProject.resProductivity
.get(j);
                        if((checkInt == Math.floor(checkInt))&&
!Double.isInfinite(checkInt)){
                            checkInt=0;
                        }
                        else{
                            System.out.println("Unfeasible due to wrong
inputs of resProductivity,effort of act "+i+" in resource "+j+" where
effort is "+effort+ " and resource Productivity is " +myProject.
resProductivity.get(j));
                            System.exit(0);
                        }
                        //check: if (Σ(rmin*tmin)|for each
partition)*resProductivity >effort there is no countable profile so stop
the programe!
                        minEff=0;
                        for(int y=0;y<chromosomes.get(c).
partition[j][i]+1;y++){
                            minEff+=myProject.minResUsage.get(j)
*chromosomes.get(c).actsList.get(i).getTallocMin();
                        }
                        if(minEff*myProject.resProductivity.get(j)>
effort){
                            //STOP PROGRAM
```

```
                                System.out.println("Unfeasible due to wrong
inputs of rmin,tmin,effort of act "+i+" in resource "+j+" where effort is
"+effort+" and min resource usage is "+myProject.minResUsage.get(j)+" with
talloc min ="+chromosomes.get(c).actsList.get(i).getTallocMin());
                                System.exit(0);
                            }
                            else {
                                //compute profile
                                poroiPerPeriod=new ArrayList();
                                ginomena = new int [2][];
                                //Generate profile from method computeProfile
                                ginomena=extraMethods.computeProfile2(effort
/myProject.resProductivity.get(j), myProject.resProductivity.get(j),
chromosomes.get(c).partition[j][i], myProject.minResUsage.get(j),
myProject.maxResUsage.get(j),chromosomes.get(c).actsList.get(i).getTallocMi
n(),myProject.getTimeHorizon());
                                //Put each quantity usage of resource in a per
time period usage schedule for each partition (per time period not for
total time of using resource in quantity x)
                                for(int z=0;z<chromosomes.get(c).partition [j]
[i]+1;z++){
                                    for(int d=0;d<ginomena[1][z];d++){
                                        poroiPerPeriod.add(ginomena[0][z]);
                                    }
                                }
                                prof.add(poroiPerPeriod);
                            }
                        }
                    }
                    chromosomes.get(c).profiles.add(i, prof);
                }
                //Set zero profile for i=n+1
                prof=new ArrayList();
                for(int j=0;j<myProject.getResourceNum();j++){
                    poroiPerPeriod=new ArrayList();
                    poroiPerPeriod.add(0);
                    prof.add(poroiPerPeriod);
                }
                chromosomes.get(c).profiles.add(myProject.getJobNum()-1,
prof);
            }

            //Set probabilities of Pmutation,Chromosome and Pmutation,resource
            double PmutationChrom;
            double PmutationChromRes;
            double PmutationInternal=-10;
            do {
                PmutationChrom=-1;
                System.out.println("Give probability of chromosome mutation");
                InputStreamReader p1 = new InputStreamReader(System.in);
                BufferedReader pm = new BufferedReader(p1);
                String pmut= pm.readLine();
                boolean okNumber=false;
                try{
                    Double.parseDouble(pmut);
                    okNumber =true;
                }
                catch(Exception e2){
                    System.out.println("Error while reading" +
e2.getMessage());
                    okNumber=false;
                }
```

```
            if (okNumber == true)
                PmutationChrom= Double.parseDouble(pmut);
        } while (PmutationChrom <0 || PmutationChrom>1);
        do {
            PmutationChromRes=-1;
            System.out.println("Give probability of resource mutation");
            InputStreamReader p2 = new InputStreamReader(System.in);
            BufferedReader pm2 = new BufferedReader(p2);
            String pmut2= pm2.readLine();
            boolean okNumber=false;
            try{
                Double.parseDouble(pmut2);
                okNumber =true;
            }
            catch(Exception e3){
                System.out.println("Error while reading" +
e3.getMessage());
                okNumber=false;
            }
            if (okNumber == true)
                PmutationChromRes= Double.parseDouble(pmut2);
        } while (PmutationChromRes <0 || PmutationChromRes>1);
        //Set POP, genes, Pmutation of GAinternal
        int evolutionStepsInternal=0;
        int popInternal=0;
        boolean posEven2;
        do {
            popInternal=-1;
            System.out.println("Give POP internal GA size");
            InputStreamReader s = new InputStreamReader(System.in);
            BufferedReader size = new BufferedReader(s);
            String psize= size.readLine();
            boolean posNumber=false;
            posEven2=false;
            try{
                Integer.parseInt(psize);
                posNumber =true;
            }
            catch(Exception e3){
                System.out.println("Error while reading" +
e3.getMessage());
                posNumber=false;
            }
            if (posNumber == true) {
                popInternal=Integer.parseInt(psize);
            }
            if(popInternal%2==0)
                posEven2 =true;
            else
                System.out.println("POP prepei na einai even");
        }
        while (popInternal < 0 || posEven2==false);
        //genes GAinternal
        do {
            evolutionStepsInternal=-1;
            System.out.println("Give total number of genes of
GAinternal");
            InputStreamReader s1 = new InputStreamReader(System.in);
            BufferedReader steps = new BufferedReader(s1);
            String tsteps= steps.readLine();
            boolean positive3=false;
            try{
```

```
                Integer.parseInt(tsteps);
                positive3 =true;
            }
            catch(Exception e4){
                System.out.println("Error while reading" +
e4.getMessage());
                positive3=false;
            }
            if (positive3 == true) {
                evolutionStepsInternal=Integer.parseInt(tsteps);
            }
        }
        while (evolutionStepsInternal < 0);
        //Pmutation of chromosomes GAinternal
        do {
            PmutationInternal=-1;
            System.out.println("Give probability of mutation");
            InputStreamReader p1 = new InputStreamReader(System.in);
            BufferedReader pm = new BufferedReader(p1);
            String pmut= pm.readLine();
            boolean okNumber3=false;
            try{
                Double.parseDouble(pmut);
                okNumber3 =true;
            }
            catch(Exception e5){
                System.out.println("Error while reading" +
e5.getMessage());
                okNumber3=false;
            }
            if (okNumber3 == true)
                PmutationInternal= Double.parseDouble(pmut);
        } while (PmutationInternal <0 || PmutationInternal>1);

        int e=0;


        //for each gene of GAexternal:
        do{
            System.out.println("Running GA external's gene: "+ (e+1));

            //CROSSOVER based on the partition table
            //1.random pairs of chromosomes of POP
            List<List<Chromosome>> parentsPairsCross = new ArrayList ();
            List<Chromosome> randomPairs ;
            //Shuffle the chromosomes list :Collections.shuffle();
            Collections.shuffle(chromosomes);
            //pairs (j,j+1) in the random  list of chromosomes
            for(int j=0;j<POPchroms;j=j+2){
                randomPairs = new ArrayList();
                randomPairs.add(chromosomes.get(j));
                randomPairs.add(chromosomes.get(j+1));
                parentsPairsCross.add(randomPairs);
            }
            //2.1point-crossover: random ineger q1, 1<=q1<=J
            //DAUGHTER: positions:1-q1: from Mother, q1+1-J: from Father
            //SON :                          Father,              Mother
            //From parentsPairsCross: 1st=mother, 2nd=father
            ArrayList <Chromosome> children = new ArrayList ();
            Chromosome daughter ;
            Chromosome son;
            ArrayList<Activity> randomList1;
```

```
            ArrayList<Activity> randomList2;
            for(i=0;i<parentsPairsCross.size();i++) {
                //DEEP CLONE of projectActivities
                randomList1 = new ArrayList();
                randomList2 = new ArrayList();
                for(int w=0;w<myProject.getJobNum();w++){
                    newAct = new Activity(w);
                    newAct.effort=myProject.projectActivities.get(w).
effort;
                    newAct.succLags=myProject.projectActivities.get(w).
succLags;
                    newAct.succLagsPercent=myProject.projectActivities.
get(w).succLagsPercent;
                    randomList1.add(newAct);
                }
                for(int q=0;q<myProject.getJobNum();q++){
                    newAct = new Activity(q);
                    newAct.effort=myProject.projectActivities.get(q).
effort;
                    newAct.succLags=myProject.projectActivities.get(q).
succLags;
                    newAct.succLagsPercent=myProject.projectActivities.
get(q).succLagsPercent;
                    randomList2.add(newAct);
                }
                //New chromosomes son+daughter
                daughter = new Chromosome(randomList1,
myProject.getResourceNum(),myProject.getJobNum());
                son = new Chromosome(randomList2,
myProject.getResourceNum(),myProject.getJobNum());
                //take random q1
                Random random1 = new Random();
                int q1 = random1.nextInt(myProject.getJobNum()-1);
                //[0,q1]
                for(int d=0;d<q1;d++){
                    for(int j=0;j<myProject.getResourceNum();j++){
                        daughter.partition[j][d]=
parentsPairsCross.get(i).get(0).partition[j][d];//from mother get(0)
                        son.partition[j][d]=
parentsPairsCross.get(i).get(1).partition[j][d]; //apo father get(1)
                    }
                }
                //[q1+1-JobNum]
                for(int d=q1;d<myProject.getJobNum();d++){
                    for(int j=0;j<myProject.getResourceNum();j++){
                        daughter.partition[j][d]=
parentsPairsCross.get(i).get(1).partition[j][d]; //rest table from mother
                        son.partition[j][d]=
parentsPairsCross.get(i).get(0).partition[j][d]; //rest table from mother
                    }
                }
                children.add(son);
                children.add(daughter);
            }

            //MUTATION based on partition table to CHILDREN
            for(int c=0;c<children.size();c++){
                int a1 = 0;
                int a2 = 0;
                int selact1;
                int selact2;
                //Give random probability to each cromosome of children
```

```
                Random randomP = new Random();
                double prob = randomP.nextDouble();
                if(prob<PmutationChrom){ //if prob<PmutationChrom-MUTATION
                    //Give random probability to each resource
                    for(int j=0;j<myProject.getResourceNum();j++){
                        Random randomPr = new Random();
                        double probChromRes = randomPr.nextDouble();
                        if(probChromRes<PmutationChromRes){
                            //get random 2 actNumbers i to exchange
positions to the corresponding value of the partition[j][i]
                            Random r1 = new Random();
                            a1 = r1.nextInt(myProject.getJobNum()-1);
                            do {
                                Random random2 = new Random();
                                a2 = random2.
nextInt(myProject.getJobNum()-1);
                            } while (a1== a2);
                            //exchange positions to partition[j][a1] and
partition[j][a2]
                            selact1 = children.get(c).partition[j][a1];
                            selact2 = children.get(c).partition[j][a2];
                            children.get(c).partition[j][a1]=selact2;
                            children.get(c).partition[j][a2]=selact1;
                        }
                    }
                }
            }

            //Compute children's profile
            for(int c=0;c<children.size();c++){ //for each chromosome
                //Set zero profile for i=0
                prof=new ArrayList();
                for(int j=0;j<myProject.getResourceNum();j++){
                    poroiPerPeriod=new ArrayList();
                    poroiPerPeriod.add(0);
                    prof.add(poroiPerPeriod);
                }
                children.get(c).profiles.add(0, prof);
                //Compute profiles for i=1,...,n
                for(i=1;i<myProject.getJobNum()-1;i++){ //for each act
                    //Set zero profile to dummies
                    prof =new ArrayList();
                    for(int j=0;j<myProject.getResourceNum();j++){//for
each resource
                        effort=children.get(c).actsList.get(i)
.effort.get(j);
                        //if act i doesn't use a resource set zero profile
                        if(effort==0){
                            poroiPerPeriod=new ArrayList();
                            for(int u=0;u<children.get(c).partition[j][i]
+1;u++){
                                poroiPerPeriod.add(0);
                            }
                        prof.add(poroiPerPeriod);
                        }
                        else{//act i uses resource j-->compute profil
                        //check: if (Σ(rmin*tmin)|for each partition)
>effort there is no countable profile so stop the programe!
                        //i.e: 5*t1+4*t2=8, tmin=2 rmin=4 rmax=6: no
countable profile.Stop the programe!
                            minEff=0;
```

```
                                for(int y=0;y<children.get(c).partition[j][i]
+1;y++){
                                    minEff+=myProject.minResUsage.get(j)*
children.get(c).actsList.get(i).getTallocMin();
                                }
                                if(minEff*myProject.resProductivity.get(j)>
effort){
                                    //STOP PROGRAM
                                    System.out.println("Unfeasible due to
wrong inputs of rmin,tmin,effort of act "+i+" in resource "+j+" where
effort is "+effort+" and min resource usage is
"+myProject.minResUsage.get(j)+" with talloc min
="+children.get(c).actsList.get(i).getTallocMin());
                                    System.exit(0);
                                }
                                //compute profile
                                else {
                                    poroiPerPeriod=new ArrayList();
                                    ginomena = new int [2][];
                                    //Generate profile from method
computeProfile
                                    ginomena=extraMethods.computeProfile2(
effort/myProject.resProductivity.get(j),
myProject.resProductivity.get(j),children.get(c).partition[j][i],
myProject.minResUsage.get(j),myProject.maxResUsage.get(j),children.get(c).a
ctsList.get(i).getTallocMin(),myProject.getTimeHorizon());
                                    //Put each quantity usage of resource in a
per time period usage schedule for each partition (per time period not for
total time of using resource in quantity x)
                                    for(int z=0;z<children.get(c).partition
[j][i]+1;z++){
                                        for(int d=0;d<ginomena[1][z];d++){
                                            poroiPerPeriod.add(ginomena[0][z])
;
                                        }
                                    }
                                    prof.add(poroiPerPeriod);
                                }
                            }
                        }
                        children.get(c).profiles.add(i, prof);

                    }
                    //Set zero profile for i=n+1
                    prof=new ArrayList();
                    for(int j=0;j<myProject.getResourceNum();j++){
                        poroiPerPeriod=new ArrayList();
                        poroiPerPeriod.add(0);
                        prof.add(poroiPerPeriod);
                    }
                    chromosomes.get(c).profiles.add(myProject.getJobNum()-1,
prof);
                }

                //2POP chromosomes + children
                ArrayList <Chromosome> chromosomes2POP = new ArrayList();
                chromosomes2POP.addAll(chromosomes);
                chromosomes2POP.addAll(children);

                extraMethods methods = new extraMethods();
                int maxDuration;
```

```
            //for each act of actList of each chrom compute duration from
profile(=max for all resources)
            for(int c=0;c<chromosomes2POP.size();c++){
                if(chromosomes2POP.get(c).fitnessValue==0){
                    chromosomes2POP.get(c).actsList.get(0).
setDurationTotal(0);
                    chromosomes2POP.get(c).actsList.
get(myProject.getJobNum()-1).setDurationTotal(0);
                    for(int j=0;j<myProject.getResourceNum();j++){
                        chromosomes2POP.get(c).actsList.get(0).
durations.add(0);
                        chromosomes2POP.get(c).actsList.
get(myProject.getJobNum()-1).durations.add(0);
                    }
                    for(i=1;i<myProject.getJobNum()-1;i++){
                        maxDuration=0;
                        for(int j=0;j<myProject.getResourceNum();j++){
                            chromosomes2POP.get(c).actsList.get(i).
durations.add(j,chromosomes2POP.get(c).profiles.get(i).get(j).size());
                            if(chromosomes2POP.get(c).actsList.get(i).
durations.get(j)>maxDuration)
                                maxDuration=
chromosomes2POP.get(c).actsList.get(i).durations.get(j);
                        }
                        chromosomes2POP.get(c).actsList.get(i).
setDurationTotal(maxDuration);
                    }
                }
            }
            //Put lags and successors in succLagID according to problem
type
            if (code==2){
                //Based on total duration compute int lags as : percent of
lag * total duration
                for(int c60=0;c60<chromosomes2POP.size();c60++){
                    for(int i60=0;i60<myProject.getJobNum();i60++){
                        Set mapSet = (Set) chromosomes2POP.get(c60).
actsList.get(i60).succLagsPercent.entrySet();
                        //Create iterator on Set
                        Iterator mapIterator = mapSet.iterator();
                        int i80=0;
                        while (mapIterator.hasNext() &&
i80<chromosomes2POP.get(c60).actsList.get(i60).succLagsPercent.size()) {
                            Map.Entry mapEntry = (Map.Entry) mapIterator.
next();
                            // getKey Method of HashMap access a key of
map
                            int curSucID60 = (int) mapEntry.getKey();
                            int lag;
                            if(chromosomes2POP.get(c60).actsList.get(i60).
succLagsPercent.get(curSucID60)<0)
                                lag = (int)
(chromosomes2POP.get(c60).actsList.get(i60).succLagsPercent.get(curSucID60)
*chromosomes2POP.get(c60).actsList.get(i60).getDurationTotal()-0.5);
                            else
                                lag = (int)
(chromosomes2POP.get(c60).actsList.get(i60).succLagsPercent.get(curSucID60)
*chromosomes2POP.get(c60).actsList.get(i60).getDurationTotal()+0.5);
                            chromosomes2POP.get(c60).actsList.get(i60).
succLagsID.put(curSucID60, lag);
                            i80++;
                        }
```

```
                    }
                }
            }
            if(code==1){
                for(int c60=0;c60<chromosomes2POP.size();c60++){
                    for(int i60=0;i60<myProject.getJobNum();i60++){
                        Set mapSet = (Set) chromosomes2POP.get(c60).
actsList.get(i60).succLags.entrySet();
                        //Create iterator on Set
                        Iterator mapIterator = mapSet.iterator();
                        int i80=0;
                        while (mapIterator.hasNext() &&
i80<chromosomes2POP.get(c60).actsList.get(i60).succLags.size()) {
                            Map.Entry mapEntry = (Map.Entry) mapIterator.
next();
                            // getKey Method of HashMap access a key of
map
                            int curSucID60 = (int) mapEntry.getKey();
                            int lag =
chromosomes2POP.get(c60).actsList.get(i60).succLags.get(curSucID60);
                            chromosomes2POP.get(c60).actsList.get(i60).
succLagsID.put(curSucID60, lag);
                            i80++;
                        }
                    }
                }
            }
            if(code==3){
                for(int c60=0;c60<chromosomes2POP.size();c60++){
                    for(int i60=0;i60<myProject.getJobNum();i60++){
                        Set mapSet = (Set) chromosomes2POP.get(c60).
actsList.get(i60).succLags.entrySet();
                        //Create iterator on Set
                        Iterator mapIterator = mapSet.iterator();
                        int i80=0;
                        while (mapIterator.hasNext() &&
i80<chromosomes2POP.get(c60).actsList.get(i60).succLags.size()) {
                            Map.Entry mapEntry = (Map.Entry) mapIterator.
next();
                            // getKey Method of HashMap access a key of
map
                            int curSucID60 = (int) mapEntry.getKey();
                            int lag = chromosomes2POP.get(c60).
actsList.get(i60).getDurationTotal();
                            chromosomes2POP.get(c60).actsList.get(i60).
succLagsID.put(curSucID60, lag);
                            i80++;
                        }
                    }
                }
            }

            //Compute FW,predecessors for each chromosome
            for(int u=0;u<chromosomes2POP.size();u++){
                if(chromosomes2POP.get(u).fitnessValue==0){
                    chromosomes2POP.get(u).fw =
extraMethods.calcFW(chromosomes2POP.get(u), myProject.getJobNum());
                    methods.calcPred(chromosomes2POP.get(u),
myProject.getJobNum());
                }
            }
```

```
            //GA_internal for each cromosome
            //create a hashmap with key each chromosome and value the
corresponding fv
            Map<Chromosome,Integer> popFV2 = new HashMap();
            int fv=1000000000;
            int u=0;
            System.out.println("Running GA internal...");
            for(int z4=0;z4<chromosomes2POP.size();z4++){
                //For chromosomes with no schedules (fv=0)
                if(chromosomes2POP.get(z4).fitnessValue==0){
                    //call GAinternal
                    fv= extraMethods.GAinternal(popInternal,
evolutionStepsInternal, PmutationInternal,chromosomes2POP.get(z4),
myProject.getJobNum(),myProject.getResourceNum(),myProject.projectResources
,myProject.getTimeHorizon());
                    popFV2.put(chromosomes2POP.get(z4), fv);
                }
                else{
                    //For chroms-parents with already defined fv don't
compute again cause nothing has changed
                    popFV2.put(chromosomes2POP.get(z4),
chromosomes2POP.get(z4).fitnessValue);
                }
            }
            //CHECK: if all chromosomes have fv=1000000000(which means
that no feasible schedule found during internalGA and sSGSU) STOP!
            int stop=0;
            for(int z45=0;z45<chromosomes2POP.size();z45++){
                if(chromosomes2POP.get(z45).fitnessValue==1000000000)
                    stop++;
            }
            if(stop==chromosomes2POP.size()){
                System.out.println("Unfeasible schedule");
                System.exit(10);
            }

            //ELITISM
            //For each chrom with fv==1000000000 remove form popFV2 and if
popFV2.size<POPchroms (too many dublicates) tote elitism-->new chroms
            do{
                if(stop>0 || popFV2.size()<POPchroms){
                    System.out.println("Elitism...");
                    for(int z65=0;z65<chromosomes2POP.size();z65++){
                        if(popFV2.containsKey(chromosomes2POP.get(z65))){
                            if(popFV2.get(chromosomes2POP.get(z65))
==1000000000){
                                //remove chrom from popFV and generate new
random chrom
                                popFV2.remove(chromosomes2POP.get(z65));
                                //new chromosome
                                // NEED DEEP CLONE OF list
projectActivities
                                randomList = new ArrayList();
                                for(i=0;i<myProject.getJobNum();i++){
                                    newAct = new Activity(i);
                                    newAct.effort=myProject.
projectActivities.get(i).effort;
                                    newAct.succLags
=myProject.projectActivities.get(i).succLags;
                                    newAct.succLagsPercent=
myProject.projectActivities.get(i).succLagsPercent;
                                    randomList.add(newAct);
```

```
                                    }
                                    Chromosome newCrhomosomeElitisim = new
Chromosome(randomList,myProject.getResourceNum(),myProject.getJobNum());
                                    //Random values in partition table--
>partition[][] me 0<= Qik <=qMax
                                    for(i=0;i<myProject.getJobNum();i++){
                                        for(int j=0;
j<myProject.getResourceNum();j++){
                                            Random rand = new Random();
                                            int value = rand.nextInt(
myProject.getMaxNumPartition()+1) ;
                                            newCrhomosomeElitisim.
partition[j][i]=value;
                                        }
                                    }
                                    prof=new ArrayList();
                                    for(int j=0;
j<myProject.getResourceNum();j++){
                                        poroiPerPeriod=new ArrayList();
                                        poroiPerPeriod.add(0);
                                        prof.add(poroiPerPeriod);
                                    }
                                    newCrhomosomeElitisim.profiles.add(0,
prof);
                                    //Compute profiles for i=1,...,n
                                    for(i=1;i<myProject.getJobNum()-1;i++){
                                        prof =new ArrayList();
                                        for(int j=0;
j<myProject.getResourceNum();j++){
                                            effort=
newCrhomosomeElitisim.actsList.get(i).effort.get(j);
                                            if(effort==0){
                                                poroiPerPeriod=new
ArrayList();
                                                for(int u90=0
;u90<newCrhomosomeElitisim.partition[j][i]+1;u90++){
                                                    poroiPerPeriod.add(0);
                                                }
                                               prof.add(poroiPerPeriod);
                                            }
                                            else{
                                                //compute profile
                                                poroiPerPeriod=new
ArrayList();
                                                ginomena = new int [2][];
                                                //Generate profile from method
computeProfile
                                                ginomena=
extraMethods.computeProfile2(effort/myProject.resProductivity.get(j),
myProject.resProductivity.get(j),newCrhomosomeElitisim.partition[j][i],
myProject.minResUsage.get(j), myProject.maxResUsage.get(j),
newCrhomosomeElitisim.actsList.get(i).getTallocMin(),myProject.getTimeHoriz
on());
                                                for(int z=0;
z<newCrhomosomeElitisim.partition[j][i]+1;z++){
                                                    for(int d=0;
d<ginomena[1][z];d++){
                                                        poroiPerPeriod.
add(ginomena[0][z]);
                                                    }
                                                }
                                                prof.add(poroiPerPeriod);
```

```
                                        }
                                    }
                                    newCrhomosomeElitisim.profiles.add(i,
prof);
                                }
                                prof=new ArrayList();
                                    for(int j=0;
j<myProject.getResourceNum();j++){
                                        poroiPerPeriod=new ArrayList();
                                        poroiPerPeriod.add(0);
                                        prof.add(poroiPerPeriod);
                                }
                                newCrhomosomeElitisim.profiles.
add(myProject.getJobNum()-1, prof);
                                newCrhomosomeElitisim.actsList.
get(0).setDurationTotal(0);
                                newCrhomosomeElitisim.actsList
.get(myProject.getJobNum()-1).setDurationTotal(0);
                                for(int j=0;
j<myProject.getResourceNum();j++){
                                    newCrhomosomeElitisim.actsList.
get(0).durations.add(0);
                                    newCrhomosomeElitisim.actsList.
get(myProject.getJobNum()-1).durations.add(0);
                                }
                                for(i=1;i<myProject.getJobNum()-1;i++){
                                    maxDuration=0;
                                    for(int j=0;
j<myProject.getResourceNum();j++){
                                        newCrhomosomeElitisim.actsList
.get(i).durations.add(j,newCrhomosomeElitisim.profiles.get(i).get(j).size()
);
                                        if(newCrhomosomeElitisim.actsList.
get(i).durations.get(j)>maxDuration)
                                            maxDuration=
newCrhomosomeElitisim.actsList.get(i).durations.get(j);
                                    }
                                    newCrhomosomeElitisim.actsLis
t.get(i).setDurationTotal(maxDuration);
                                }
                                if (code==2){
                                //Based on total duration compute int lags
as : percent of lag * total duration
                                    for(int i60=0
;i60<myProject.getJobNum();i60++){
                                        Set mapSet = (Set)
newCrhomosomeElitisim.actsList.get(i60).succLagsPercent.entrySet();
                                        Iterator mapIterator =
mapSet.iterator();
                                        int i80=0;
                                        while (mapIterator.hasNext() &&
i80<newCrhomosomeElitisim.actsList.get(i60).succLagsPercent.size()) {
                                            Map.Entry mapEntry =
(Map.Entry) mapIterator.next();
                                            int curSucID60 = (int)
mapEntry.getKey();
                                            int lag;
                                            if(newCrhomosomeElitisim
.actsList.get(i60).succLagsPercent.get(curSucID60)<0)
                                                lag = (int)
(newCrhomosomeElitisim.actsList.get(i60).succLagsPercent.get(curSucID60)*ne
wCrhomosomeElitisim.actsList.get(i60).getDurationTotal()-0.5);
```

```
                                                else
                                                    lag = (int)
(newCrhomosomeElitisim.actsList.get(i60).succLagsPercent.get(curSucID60)*ne
wCrhomosomeElitisim.actsList.get(i60).getDurationTotal()+0.5);
                                                newCrhomosomeElitisim
.actsList.get(i60).succLagsID.put(curSucID60, lag);
                                                i80++;
                                            }
                                        }
                                    }
                                    if(code==1){
                                        for(int i60=0;
i60<myProject.getJobNum();i60++){
                                            Set mapSet = (Set)
newCrhomosomeElitisim.actsList.get(i60).succLags.entrySet();
                                            Iterator mapIterator =
mapSet.iterator();
                                            int i80=0;
                                            while (mapIterator.hasNext() &&
i80<newCrhomosomeElitisim.actsList.get(i60).succLags.size()) {
                                                Map.Entry mapEntry =
(Map.Entry) mapIterator.next();
                                                int curSucID60 = (int)
mapEntry.getKey();
                                                int lag =
newCrhomosomeElitisim.actsList.get(i60).succLags.get(curSucID60);
                                                newCrhomosomeElitisim.
actsList.get(i60).succLagsID.put(curSucID60, lag);
                                                i80++;
                                            }
                                        }
                                    }
                                    if(code==3){
                                        for(int i60=0;
i60<myProject.getJobNum();i60++){
                                            Set mapSet = (Set)
newCrhomosomeElitisim.actsList.get(i60).succLags.entrySet();
                                            Iterator mapIterator =
mapSet.iterator();
                                            int i80=0;
                                            while (mapIterator.hasNext() &&
i80<newCrhomosomeElitisim.actsList.get(i60).succLags.size()) {
                                                Map.Entry mapEntry =
(Map.Entry) mapIterator.next();
                                                int curSucID60 = (int)
mapEntry.getKey();
                                                int lag =
newCrhomosomeElitisim.actsList.get(i60).getDurationTotal();
                                                newCrhomosomeElitisim.
actsList.get(i60).succLagsID.put(curSucID60, lag);
                                                i80++;
                                            }
                                        }
                                    }
                                    newCrhomosomeElitisim.fw =
extraMethods.calcFW(newCrhomosomeElitisim, myProject.getJobNum());
                                    methods.calcPred(newCrhomosomeElitisim,
myProject.getJobNum());
                                    //GA internal and fitness value
                                    fv= extraMethods.GAinternal(popInternal,
evolutionStepsInternal, PmutationInternal,newCrhomosomeElitisim,
```

```
myProject.getJobNum(),myProject.getResourceNum(),myProject.projectResources
,myProject.getTimeHorizon());
                                popFV2.put(newCrhomosomeElitisim, fv);
                            }
                        }
                    }
                    stop--;
                }
            }
            while(stop>0 && popFV2.size()<POPchroms);

            //sort hashmap popFV2 with increasing order based on fv and
add to: sortedpopFV
            List list4 = new LinkedList(popFV2.entrySet());
            Collections.sort(list4, new Comparator() {  // Defined Custom
Comparator here
                public int compare(Object o1, Object o2) {
                return ((Comparable) ((Map.Entry) (o1)).getValue())
                       .compareTo(((Map.Entry) (o2)).getValue());
                }
            }
            );
            HashMap sortedpopFV2 = new LinkedHashMap(); // Here I am
copying the sorted list in HashMap using LinkedHashMap to preserve the
insertion order
            for (Iterator it2 = list4.iterator(); it2.hasNext();) {
                Map.Entry entry = (Map.Entry) it2.next();
                sortedpopFV2.put(entry.getKey(), entry.getValue());
            }
            //SELECTION from 2pop to POP with those with min fv from
sortedpopFV!
            chromosomes.clear();
            Set mapSet = (Set) sortedpopFV2.entrySet();
            //Create iterator on Set
            Iterator mapIterator = mapSet.iterator();
            int i50=0;
            while (mapIterator.hasNext() && i50<POPchroms) {
                Map.Entry mapEntry = (Map.Entry) mapIterator.next();
                // getKey Method of HashMap access a key of map
                Chromosome key = (Chromosome) mapEntry.getKey();
                chromosomes.add(key);
                i50++;
            }

             System.out.println("In gene: "+ (e+1) +" best chromosome's
makespan : "+chromosomes.get(0).fitnessValue);
             e++;
         } while(e<evolutionStepsExternal);

         Chromosome bestFinalChromosome=chromosomes.get(0);//first of last

         //Outputs: minMakespan, ST of act,duartions and Profil with time
period + quantity
         System.out.println("Total makespan of project:
"+chromosomes.get(0).fitnessValue+" with activities profiles and start
times by execution order as following:");
         for(i=0;i<myProject.getJobNum();i++){
             System.out.println(i+1 +". Activity : "
+bestFinalChromosome.actsList.get(i).getID() + " with ST: "
+bestFinalChromosome.actsList.get(i).getST()
                     +" and total duration: "+
bestFinalChromosome.actsList.get(i).getDurationTotal());
```

```
        }
        System.out.println("Profile of each activity per resource type:
");
        for(i=0;i<myProject.getJobNum();i++){
            for(int j=0;j<myProject.getResourceNum();j++){
                for(int k=0
;k<bestFinalChromosome.profiles.get(bestFinalChromosome.actsList.get(i).get
ID()).get(j).size();k++){
                    System.out.println("Activity with id : "
+bestFinalChromosome.actsList.get(i).getID() + " uses resource "+ j+
" in time period "+ (bestFinalChromosome.actsList.get(i).getST()+k)+" with
" + bestFinalChromosome.profiles.get(bestFinalChromosome.actsList.get(i).
getID()).get(j).get(k) + " units");
                }
            }
        }
    }
```

C.2. Υπολογισμός Προφίλ Πόρων

```
public static int [][] computeProfile2(int eff,int resProductivity, int
partitionNum, int rmin, int rmax, int tmin, int T){
        int ginomena[][]= new int [2][partitionNum+1];
        int xronoi[]=new int [partitionNum+1];
        int time[]= new int [partitionNum+1];
        boolean again=false;
        int minEff;
        boolean check;
        int o=0;
        double xronos=0;
        double integ=0;
        do{
            again=false;
            do{
                minEff=0;
                check=false;
                integ=0;
                //Random quantities randomNum = rand.nextInt((max - min) +
1) + min;
                for(int z=0;z<partitionNum+1;z++){
                    Random rand1 = new Random();
                    int poroi = rand1.nextInt(rmax-rmin)+ rmin;
                    ginomena[0][z]=poroi;
                }
                //check if all quants the same then if eff/posotita!=int
take new quants
                for(int z=0;z<partitionNum+1;z++){
                    for(int k=0;k<partitionNum+1;k++){
                        if(ginomena[0][z]==ginomena[0][k]){
                            o++;
                        }
                    }
                }
                if(o==(partitionNum+1)*(partitionNum+1)){
                    integ=(double)eff/ginomena[0][0];
                    if((integ == Math.floor(integ)) &&
!Double.isInfinite(integ)){
                        check=false;
                    }
                    else
                        check=true;
```

```
                }
                //check if posotites*tmin >eff take new quantities
                for(int z=0;z<partitionNum+1;z++){
                     minEff+=ginomena[0][z]*tmin;
                }
                if(minEff>eff)
                    check=true;
            }while(check==true);

            //Compute times
            if(partitionNum==0){
                //if only one partition
                xronos=(double)eff/ginomena[0][0];
                //check if the result is int, if not take new quantities
                if ((xronos == Math.floor(xronos)) &&
!Double.isInfinite(xronos)){
                    if(xronos<T&&xronos*ginomena[0][0]==eff)
                        ginomena[1][0]= (int) xronos;
                    else
                        again=true;
                }
                else
                    again=true;
            }
            else {
                //initialise times xronoi =tmin
                for(int z=0;z<partitionNum+1;z++){
                    time[z]=tmin;
                    xronoi[z]=-1;
                }
                //Compute times for each partition
                xronoi=
extraMethods.computeTimes(time,xronoi,partitionNum,partitionNum,ginomena,
tmin, T, eff, resProductivity);
                //if no times could be computed to result in effort then
take new quantities
                if(xronoi[0]==-1)
                    again=true;
                else{
                    //put xronoi[] in table ginomena[1][]
                    for(int k=0;k<partitionNum+1;k++){
                        ginomena[1][k]=xronoi[k];
                    }
                }
            }
        }while (again==true);
        return ginomena;
    }

 private static int [] computeTimes(int t[], int []xronoi,int z,int
partitionNum,int ginomena [][],int tmin, int T,int eff,int
resProductivity){
        int compEffort;
        int o;
        boolean okProducts=false;

        z=0;
        do{
            compEffort=0;
            for(int i=0;i<partitionNum+1;i++){
                compEffort+=ginomena[0][i]*t[i];
            }
```

```
            if(eff==compEffort){
                okProducts=true;
                for(int i=0;i<partitionNum+1;i++){
                    xronoi[i]=t[i];
                }
            }
            o=0;
            if(t[partitionNum]!=T){
                for(int i=0;i<partitionNum;i++){
                    if(t[i]>=T){
                        o++;
                        t[i+1]++;
                        t[i]=tmin;
                        if(o==1)
                            t[z]--; //to take the right value
                    }
                }
            }
            t[z]++;
        } while(t[partitionNum]!=T&&okProducts==false);
        return xronoi;
    }
```

C.3. Εσωτερικός Γενετικός Αλγόριθμος

```
 public static int GAinternal(int population,int steps, double probability,
Chromosome chrom, int jobNum, int resNum, ArrayList<Integer>
ProjectResources, int TimeHorizon) throws IOException{
        //G.A.
        int POPint=population;
        int evolutionSteps=steps;
        double PmutationInt=probability;
        //Initial population
        List<List<Activity>> individuals = new ArrayList ();
        ArrayList <Activity> schedActs ;
        ArrayList<Activity> actsWithPredInSched;
        for(int i=0;individuals.size() < POPint; i++){
            ArrayList<Activity> lista = new ArrayList();
            Activity nea;
            //copy actList of chrom!!
            for(int y=0;y<jobNum;y++){
                nea = new Activity(y);
                nea.effort=chrom.actsList.get(y).effort;
                nea.durations=chrom.actsList.get(y).durations;
                nea.succLagsID=chrom.actsList.get(y).succLagsID;
                nea.setDurationTotal(chrom.actsList.get(y)
.getDurationTotal());
                nea.predecessors=chrom.actsList.get(y).predecessors;
                lista.add(nea);
            }
            schedActs = new ArrayList();
            schedActs.add(lista.get(0)); //put dummy

            do {
                actsWithPredInSched = new ArrayList();
                for(int j=0;j<jobNum;j++) {
                    int r=0;
                    for(int k=0; k<lista.get(j).predecessors.size();k++){
                        for(int o=0;o<schedActs.size();o++){
```

```
                                if (schedActs.get(o).getID() == lista.get(j).
predecessors.get(k).getID())
                                    r++;
                            }
                        }
                        boolean r1=false;
                        //check if act already scheduled
                        for (Activity act: schedActs) {
                        if(act.getID()==lista.get(j).getID())
                            r1= true;
                        }
                        //check if pred already scheduled
                        if (r== lista.get(j).predecessors.size() && r1==false)
                            actsWithPredInSched.add(lista.get(j));
                    }
                    //Random from actsWithPredInSched choose which act to put
                Random randomGenerator1 = new Random();
                int randomIndex1 = randomGenerator1.nextInt
(actsWithPredInSched.size());
                Activity randomAct1 = actsWithPredInSched.get(randomIndex1);
                schedActs.add(randomAct1);
                }
                while (schedActs.size() != lista.size());

                individuals.add(schedActs);
            }
            int e=0;
            int minMakespan=1000000000;
            int numIndivWithMinMakespan;
        do{
            //Dublicate the list of individuals for chrossover so not to have
problems with acts in children and parents later on counting ssgs and in
STandFT. NEED HARD COPY
            List<List<Activity>> individualsCOPY = new ArrayList ();
            List<Activity> individ = null;
            Activity newactivity;
            for(int y200=0;y200<individuals.size();y200++){
                individ= new ArrayList();
                for(int y300=0;y300<jobNum;y300++){
                    newactivity = new Activity(individuals.get(y200).
get(y300).getID());
                    newactivity.effort=individuals.get(y200).get(y300).effort;
                    newactivity.succLagsID=
individuals.get(y200).get(y300).succLagsID;
                    newactivity.durations=
individuals.get(y200).get(y300).durations;
                    newactivity.predecessors=
individuals.get(y200).get(y300).predecessors;
                    newactivity.setDurationTotal(individuals.get(y200).
get(y300).getDurationTotal());
                    individ.add(newactivity);
                }
                individualsCOPY.add(individ);
            }

            //CROSSOVER
            List<List<List<Activity>>> parentsPairsCross = new ArrayList ();
            List<List<Activity>> randomPairs ;
            //shuffle the individual list
            Collections.shuffle(individualsCOPY);
            //do pairs (j,j+1) in the random list of individuals!
            for(int j=0;j<POPint;j=j+2){
```

```
            randomPairs = new ArrayList();
            randomPairs.add(individualsCOPY.get(j));
            randomPairs.add(individualsCOPY.get(j+1));
            parentsPairsCross.add(randomPairs);
        }
        //2.u have pairs and go for 2point-crossover: random inegers q1,q2
1<=q1<=q2<=J
        //DAUGHTER: positions:1-q1: from Mother, q1+1-q2: from Father, q2+
1-J:from Mother
        //SON :                          Father,              Mother,
Father
        //from parentsPairsCross : 1st=mother kai to 2nd=father
        List<List<Activity>> childrenInt = new ArrayList ();
        List<Activity> daughter ;
        List<Activity> son;
        for(int i=0;i<parentsPairsCross.size();i++) {
            daughter = new ArrayList();
            son = new ArrayList();
            //take random q1,q2
            Random random1 = new Random();
            int q1 = random1.nextInt(jobNum);
            Random random2 = new Random();
            int q2 = random2.nextInt((jobNum-q1))+q1; //q2>q1
            //[0,q1]
            for(int d1=0;d1<q1;d1++){
                daughter.add(parentsPairsCross.get(i).get(0).get(d1));
//from mother get(0)
                son.add(parentsPairsCross.get(i).get(1).get(d1)); //from
father get(1)
            }
            //[q1+1,q2]
            int r=0;
            for(int d=0;q2-q1 != r ;d++){
                    if(extraMethods.isThisIDinTheList
(daughter,parentsPairsCross.get(i).get(1).get(d).getID())==false){
                        daughter.add(parentsPairsCross.get(i).
get(1).get(d)); //from father
                        r++;
                    }
            }
            int r1=0;
            for(int d=0;q2-q1 != r1 ;d++){
                    if(extraMethods.isThisIDinTheList(son,
parentsPairsCross.get(i).get(0).get(d).getID())==false){
                        son.add(parentsPairsCross.get(i).get(0).get(d));
//from mother
                        r1++;
                    }
            }
            //[q2+1,J]
            int r2=0;
            for(int d=0;jobNum-q2 != r2 ;d++){
                    if(extraMethods.isThisIDinTheList(daughter,
parentsPairsCross.get(i).get(0).get(d).getID())==false){
                        daughter.add(parentsPairsCross.get(i).
get(0).get(d)); //from mother
                        r2++;
                    }
            }
            int r3=0;
            for(int d=0;jobNum-q2 != r3 ;d++){
```

```
                    if(extraMethods.isThisIDinTheList
(son,parentsPairsCross.get(i).get(1).get(d).getID())==false){
                        son.add(parentsPairsCross.get(i).get(1).get(d));
//from father
                        r3++;
                    }
            }
            //check if precedence relations are violated
                    int ok1;
                    int ok2=0;
                    for(int p=0;p<jobNum;p++){
                        ok1=0;
                        for(int q=0;q<p;q++){
                            for(r=0;r<son.get(p).predecessors.size();r++)
{                               if(son.get(q).getID()==
son.get(p).predecessors.get(r).getID())
                                    ok1++;
                            }
                        }
                    }
                    ok2=0;
                    for(int p=0;p<jobNum;p++){
                        ok1=0;
                        for(int q=0;q<p;q++){
                            for(r=0;r<daughter.get(p)
.predecessors.size();r++) {
                                if(daughter.get(q).getID()==daughter.
get(p).predecessors.get(r).getID())
                                    ok1++;
                            }
                        }
                    }
            childrenInt.add(son);
            childrenInt.add(daughter);
        }
        //MUTATION
        //Give random probability to each individual of children
        for(int i=0;i<childrenInt.size();i++){
            int a1 = 0;
            int a2 = 0;
            Activity selact1;
            Activity selact2;
            boolean ok=false;
            int ok1;
            int ok2;
            Random randomPr = new Random();
            double prob = randomPr.nextDouble();
            if(prob<PmutationInt){
                //if prob<Pumtation -->Mutate individual
                do{
                    ok1=0;
                    ok2=0;
                    ok=false;
                    //get random 2 acts to exchange positions
                    Random r4 = new Random();
                    a1 = r4.nextInt(jobNum-1);
                    do {
                        Random random3 = new Random();
                        a2 = random3.nextInt(jobNum-1);
                    } while (a1== a2);
                    //exchange positions
                    selact1 = childrenInt.get(i).get(a1);
```

```
                    selact2 = childrenInt.get(i).get(a2);
                    childrenInt.get(i).remove(a1);
                    childrenInt.get(i).add(a1, selact2);
                    childrenInt.get(i).remove(a2);
                    childrenInt.get(i).add(a2, selact1);
                    //check if preced relations are violated (2) for all
acts of inidividual
                    for(int p=0;p<jobNum;p++){
                        ok1=0;
                        for(int q=0;q<p;q++){
                            for(int r=0;r<childrenInt.get(i).get(p).
predecessors.size();r++) {
                                if(childrenInt.get(i).get(q).getID()==
childrenInt.get(i).get(p).predecessors.get(r).getID())
                                ok1++;
                            }
                        }
                        if(ok1== childrenInt.get(i).get(p).predecessors
.size())
                            ok2++;
                    }
                    if(ok2==jobNum)
                        ok=true;
                    else {
                        //put individual in the first condition
                        childrenInt.get(i).remove(a1);
                        childrenInt.get(i).add(a1, selact1);
                        childrenInt.get(i).remove(a2);
                        childrenInt.get(i).add(a2, selact2);
                        }
                } while(ok==false);
            }
        }
        //2pop population= individuals+children
        List<List<Activity>> DoublePOP = new ArrayList();
        DoublePOP.addAll(individuals);
        DoublePOP.addAll(childrenInt);
        //popFV= hashmap with key the individual and value the fitness
value
        Map<List<Activity>,Integer> popFV = new HashMap();
        //compute fitness value by ssgs
        int fv=1000000000;
        int u=0;
        boolean idio=false;
        for(int b10=0;b10<DoublePOP.size();b10++){
            fv=calcSSGSU(DoublePOP.get(b10),chrom, resNum,jobNum,
ProjectResources);
            if(minMakespan>fv)
                minMakespan=fv;
            if(popFV.containsKey(DoublePOP.get(b10)))
                u++;
            popFV.put(DoublePOP.get(b10), fv);

        }
       //ELITISM
        if(popFV.size()<POPint) {

            ArrayList <Activity> schedActs2 ;
            ArrayList<Activity> actsWithPredInSched2;
            for(int i3=0;popFV.size() < POPint; i3++){
                ArrayList<Activity> lista2 = new ArrayList();
                Activity nea2;
```

```
                //Kanw copy to actList tu chrom!!
                for(int y=0;y<jobNum;y++){
                    nea2 = new Activity(y);
                    nea2.effort=chrom.actsList.get(y).effort;
                    nea2.durations=chrom.actsList.get(y).durations;
                    nea2.succLagsID=chrom.actsList.get(y).succLagsID;

nea2.setDurationTotal(chrom.actsList.get(y).getDurationTotal());
                    nea2.predecessors=chrom.actsList.get(y).predecessors;
                    lista2.add(nea2);
                }
                schedActs2 = new ArrayList();
                schedActs2.add(lista2.get(0));
                do {
                    actsWithPredInSched2 = new ArrayList();
                    for(int j=0;j<jobNum;j++) {
                        int r=0;
                        for(int k=0; k<lista2.get(j).
predecessors.size();k++){
                            for(int o=0;o<schedActs2.size();o++){
                                if (schedActs2.get(o).getID() ==
lista2.get(j).predecessors.get(k).getID())
                                    r++;
                            }
                        }
                        boolean r1=false;
                        for (Activity act: schedActs2) {
                            if(act.getID()==lista2.get(j).getID())
                                r1= true;
                        }
                        if (r== lista2.get(j).predecessors.size() &&
r1==false)
                            actsWithPredInSched2.add(lista2.get(j));
                    }
                    Random randomGenerator = new Random();
                    int randomIndex = randomGenerator.nextInt
(actsWithPredInSched2.size());
                    Activity randomAct =
actsWithPredInSched2.get(randomIndex);
                    schedActs2.add(randomAct);
                }
                while (schedActs2.size() != lista2.size());
                fv=calcSSGSU(schedActs2, chrom,resNum,jobNum,
ProjectResources);
                popFV.put(schedActs2, fv);
                if(minMakespan>fv)
                    minMakespan=fv;
            }
        }
        //increasing sort hashmap popFV based on fv and add in map:
sortedpopFV
        List list = new LinkedList(popFV.entrySet());
        Collections.sort(list, new Comparator() {  // Defined Custom
Comparator here
            public int compare(Object o1, Object o2) {
               return ((Comparable) ((Map.Entry) (o1)).getValue())
                  .compareTo(((Map.Entry) (o2)).getValue());
            }
            }
        );
```

```
        HashMap sortedpopFV = new LinkedHashMap(); // Here I am copying
the sorted list in HashMap using LinkedHashMap to preserve the insertion
order
        for (Iterator it = list.iterator(); it.hasNext();) {
              Map.Entry entry = (Map.Entry) it.next();
              sortedpopFV.put(entry.getKey(), entry.getValue());
        }
        //SELECTION from 2pop to POP with min fv from sortedpopFV!
        individuals.clear();
        Set mapSet = (Set) sortedpopFV.entrySet();
        //Create iterator on Set
        Iterator mapIterator = mapSet.iterator();
        int i=0;
        while (mapIterator.hasNext() && i<POPint) {
            Map.Entry mapEntry = (Map.Entry) mapIterator.next();
            // getKey Method of HashMap access a key of map
            List<Activity> key = (List<Activity>) mapEntry.getKey();
            individuals.add(key);
            i++;
        }
        //End of GA
        e++;
    } while(e<evolutionSteps);
    if(minMakespan<1000000000) {
        //put to actList of chromosomeExternal the best individualInternal
and as fv=minMakespan;
        for(int i10=0;i10<jobNum;i10++){
            for(int j100=0;j100<jobNum;j100++){
                if(chrom.actsList.get(i10).getID()==individuals.get(0).
get(j100).getID()){
                    chrom.actsList.get(i10).setST(individuals.get(0).
get(j100).getST());
                    chrom.actsList.get(i10).setFT(individuals.get(0).
get(j100).getFT());
                }
            }
        }
    }
    chrom.fitnessValue=minMakespan;
    return minMakespan;
    }
```

C.4. Σειριακή Μέθοδος Παραγωγής Χρονοπρογραμμάτων (s-SGS)

```
public static int calcSSGSU(List<Activity> actList, Chromosome chromos,int
                  resNum, int jobNum, ArrayList<Integer> projectResources)
{
        int r1=0;
        boolean r;
        int TimeHorizon = 1000;
        Activity selectedAc = null;
        ArrayList <StatusSSGSu> statuses = new ArrayList();
        ArrayList<Activity> unscheduledSet =new ArrayList();
        ArrayList<Activity> scheduledSet =new ArrayList();

        //calc ES, LS with FloydWarshall
        for (int a=0;a<jobNum;a++){
            actList.get(a).setESfw(chromos.fw[0][actList.get(a).getID()]);
            actList.get(a).setLSfw(-
chromos.fw[actList.get(a).getID()][0]);
        }
```

```
          //all acts to unscheduleSet
          unscheduledSet.addAll(actList);
          //dummy act 0: ST=FT=0
          actList.get(0).setFT(0);
          actList.get(0).setST(0);
          scheduledSet.add(actList.get(0));
          unscheduledSet.remove(actList.get(0));

          //Max unschedule steps
          int u=0;
          int maxu=30;
          int s=1;
          boolean stopES=false;
          boolean stopU=false;

          //Table of current available resources of first snapshot
          StatusSSGSu arxiko = new StatusSSGSu(0, resNum, TimeHorizon);
          for (int l=0; l<resNum ;l++) {
              for(int c=0;c<TimeHorizon;c++) {
                 arxiko.curAvailableResources[l][c] =
projectResources.get(l);
              }
          }
          statuses.add(arxiko);

          //Till schedule set has all acts except of last dummy
          while(scheduledSet.size()!=jobNum-1 && u<=maxu && stopES==false &&
stopU==false) {

              //New snapshot for each repetiton
              StatusSSGSu newStatus = new StatusSSGSu(s, resNum ,
TimeHorizon);
              newStatus.scheduledAct.addAll(scheduledSet);

              //calculate eligible acts
              for (int i=1; i<jobNum-1 ;i++) {
                  r=false;
                  r1=0;
                  //1.check if act already scheduled
                    for (Activity act: newStatus.scheduledAct) {
                    if(act.getID()==actList.get(i).getID()) r= true;
                    }

                  //2.check if predec of act already scheduled
                  for (int j=0;j<actList.get(i).predecessors.size();j++) {
                      for (Activity act1: newStatus.scheduledAct) {
                        if (act1.getID() ==
actList.get(i).predecessors.get(j).getID())
                          r1++;
                      }
                  }
                  //add to eligibleAct
                  if (r1==actList.get(i).predecessors.size() && r==false )
                   newStatus.eligibleActs.add(actList.get(i));

              }

              //select act from eligible

newStatus.setIDselectedAct(selectActivityGA(newStatus.eligibleActs,
actList));
              //selectAc=selected activity
```

```
            for(Activity act:actList)
                  {
                  if(act.getID()==newStatus.getIDselectedAct())
selectedAc=act;
            }
            //find a resource feasible time schedule of selectedAc
            int Ri;
            int Rtotal=0;
            boolean ok=false;
            int t;
            int x;
            int timestart=0;
            if(selectedAc.getESfw()<0)
                timestart=0;
            else
                timestart=selectedAc.getESfw();
            if(selectedAc.getESfw()>800){
                System.out.print("inf ES");
                stopU=true;
                break;
            }
            //t>=ESselectedAc
            for (t=timestart; ok==false;t++) {
            //check if there are available resources until completion of
selectedAc for each τ
                for(int p=0; p < resNum;p++){
                    //for each resource
                    Ri=0;
                    x=0;
                    for (int τ=t+1; τ <= t + selectedAc.durations.get(p);
τ++){
                        //if (required<available)for each resource -->OK
                        if
(chromos.profiles.get(selectedAc.getID()).get(p).get(x) <= statuses.get(s-
1).curAvailableResources[p][τ])
                           Ri++;
                        x++;
                    }
                    if (Ri==selectedAc.durations.get(p))
                        Rtotal++;
                }
                if (Rtotal ==resNum)
                    ok=true;
                else ok=false;
                Rtotal=0;
                if(t>TimeHorizon) ok=true;

            }
            t=t-1;  //to take the right value


            /*Check based on t with LS to see whether to SCHEDULE or to
UNSCHEDULE*/
            //if t<=LS of act then SCHEDULE:
            if(t<=selectedAc.getLSfw()) {
                scheduledSet.add(selectedAc); //add to scheduleSet
                unscheduledSet.remove(selectedAc);
                newStatus.setSelectedTime(t);
                selectedAc.setST(t);
                selectedAc.setFT((t+ selectedAc.getDurationTotal()));
                //update resources
```

```
                    newStatus.curAvailableResources = statuses.get(s-
1).curAvailableResources;
                    for (int l=0; l<resNum ;l++) {
                        x=0;
                        for(int c=t+1; c<=t+ selectedAc.durations.get(l);c++){
                            newStatus.curAvailableResources[l][c] -=
chromos.profiles.get(selectedAc.getID()).get(l).get(x);
                            x++;
                        }
                    }
                    //update ES kai LS of Unscheduled acts:
                    for(int w=0;w<unscheduledSet.size();w++){
                        if(unscheduledSet.get(w).getESfw() <
selectedAc.getST()+
chromos.fw[selectedAc.getID()][unscheduledSet.get(w).getID()])
                            unscheduledSet.get(w).setESfw(selectedAc.getST()+
chromos.fw[selectedAc.getID()][unscheduledSet.get(w).getID()]);
                        if(unscheduledSet.get(w).getLSfw() >
selectedAc.getST()-
chromos.fw[unscheduledSet.get(w).getID()][selectedAc.getID()])
                            unscheduledSet.get(w).setLSfw(selectedAc.getST()-
chromos.fw[unscheduledSet.get(w).getID()][selectedAc.getID()]);
                    }
                }
                else{//UNSCHEDULE STEP
                    u++;//unschedule steps
                    //Select acts to put in set U
                    for(int i=0;i<scheduledSet.size();i++){
                        if(selectedAc.getLSfw()==scheduledSet.get(i).getST()-
chromos.fw[selectedAc.getID()][scheduledSet.get(i).getID()])
                            newStatus.U.add(scheduledSet.get(i));
                    }
                    //if U empty
                    if(newStatus.U.isEmpty())
                        stopU=true;
                    else {
                        //for all iεU: right shift increase start time by t-
LS(selectedAct)and unschedule
                        for(int i=0;i<newStatus.U.size() &&
stopES==false;i++){
                            Activity k = newStatus.U.get(i);
                            int newES = k.getST()+ (t-selectedAc.getLSfw()) ;
                            if(newES<=-chromos.fw[k.getID()][0]) { //if ES<=LS
                               k.setESfw(newES);
                               //Unscedule Act
                               scheduledSet.remove(k); //remove ap scheduleSet
                               newStatus.unscheduledAct.add(k);
                               unscheduledSet.add(k); //add to unscheduledSet
                            }
                            else
                                stopES =true;
                        }
                      if(stopES==false) {
                        //minST of acts in U:
                        int minSTu = 100000000;
                        for(int i=0;i<newStatus.U.size();i++){
                            if(newStatus.U.get(i).getST()<minSTu)
                                minSTu=newStatus.U.get(i).getST();
                        }
                        //UNSCHEDULE all scheduled acts with ST>min{ST from U}
                        for(int i=0; i<scheduledSet.size();i++) {
                            Activity q= scheduledSet.get(i);
```

```
                        if (q.getST()>minSTu) {
                            scheduledSet.remove(q);
                            newStatus.unscheduledAct.add(q);
                            unscheduledSet.add(q);
                        }
                    }
                    //Update ES-LS of act V\C
                    for(int i=0; i<unscheduledSet.size();i++){
                        int maxi=0;
                        Activity z = unscheduledSet.get(i);
                        for(int j=0;j<newStatus.U.size();j++){
                            if(newStatus.U.get(j).getESfw()+
chromos.fw[newStatus.U.get(j).getID()][z.getID()]>maxi)
                                maxi= newStatus.U.get(j).getESfw()+
chromos.fw[newStatus.U.get(j).getID()][z.getID()];
                        }
                        if(maxi>z.getESfw())
                            z.setESfw(maxi);
                        z.setLSfw(-chromos.fw[z.getID()][0]);

                        for(int v=0;v<scheduledSet.size();v++){
                            //ES
                            if(z.getESfw()<scheduledSet.get(v).getST()+
chromos.fw[scheduledSet.get(v).getID()][z.getID()])
                                z.setESfw(scheduledSet.get(v).getST()+
chromos.fw[scheduledSet.get(v).getID()][z.getID()]);
                            //LS
                            if(z.getLSfw()>scheduledSet.get(v).getST()-
chromos.fw[z.getID()][scheduledSet.get(v).getID()])
                                z.setLSfw(scheduledSet.get(v).getST()-
chromos.fw[z.getID()][scheduledSet.get(v).getID()]);
                        }
                    }
                    //Update curAvailableRes
                    int x1;
                    newStatus.curAvailableResources = statuses.get(s-
1).curAvailableResources;
                    for (int l=0; l<resNum ;l++) {
                        for (int
a3=0;a3<newStatus.unscheduledAct.size();a3++){
                            x1=0;
                            Activity activ =
newStatus.unscheduledAct.get(a3);
                            int time = activ.getST();
                            for(int c=time+1; c<=time+
activ.durations.get(l);c++){
                                newStatus.curAvailableResources[l][c] +=
chromos.profiles.get(activ.getID()).get(l).get(x1);
                                x1++;
                            }
                        }
                    }
                  }
                }
             }
            statuses.add(newStatus);
            s++;
        }
        Activity lastDummy=unscheduledSet.get(0);
        if (u<=maxu && stopES==false && stopU==false) {
            //Last dummy act
            //ST=FT=maxFTpred
```

```
            int lastFT =0;
            for(int y150=0;y150<scheduledSet.size();y150++) {
                 if (scheduledSet.get(y150).getST()+
scheduledSet.get(y150).getDurationTotal()> lastFT)
                     lastFT = scheduledSet.get(y150).getST()+
scheduledSet.get(y150).getDurationTotal();
            }
            lastDummy.setFT(lastFT);
            lastDummy.setST(lastFT);
            scheduledSet.add(lastDummy);
            unscheduledSet.clear();
            return (lastFT);
        }
        else {
            return 1000000000;
        }
    }
```

C.5. Ορισμός Βασικών Οντοτήτων Αντικειμενοστραφούς Μοντέλου

```
public class Project {
    public ArrayList<Activity> projectActivities;
    private int jobNum;
    private int resourceNum;
    private int maxNumPartition; //qMax
    private int TimeHorizon;
    public ArrayList<Integer> projectResources; //availability
    public ArrayList <Integer> resProductivity;
    public ArrayList <Integer> minResUsage;
    public ArrayList <Integer> maxResUsage;
    public ArrayList <Activity> scheduledSet;
    public int [][] fw;
    public ArrayList <Activity> unscheduledSet;
    public List<Chromosome> bestChromosomes;
    public ArrayList<Double> resourceStrength;

    public Project(int jobs) {
        this.projectActivities=new ArrayList();
        this.resProductivity = new ArrayList();
        this.maxResUsage = new ArrayList();
        this.minResUsage = new ArrayList();
        this.projectResources = new ArrayList();
        this.scheduledSet = new ArrayList();
        this.fw = new int [jobs][jobs];
        this.unscheduledSet = new ArrayList();
        this.maxNumPartition=0;
        this.bestChromosomes= new ArrayList();
        this.resourceStrength=new ArrayList();
    }
    public void setJobNum(int jobNum) {
        this.jobNum = jobNum;
    }
    public void setResourceNum(int resourceNum) {
        this.resourceNum = resourceNum;
    }
    public int getJobNum() {
        return jobNum;
    }
    public int getResourceNum() {
        return resourceNum;
    }
    public int getMaxNumPartition() {
```

```
        return maxNumPartition;
    }
    public void setMaxNumPartition(int maxNumPartition) {
        this.maxNumPartition = maxNumPartition;
    }
    public void setTimeHorizon(int TimeHorizon) {
        this.TimeHorizon = TimeHorizon;
    }
    public int getTimeHorizon() {
        return TimeHorizon;
    }
}

public class Chromosome {
    public ArrayList<Activity> actsList; //activity list
    public int partition [][]; //partition per resource per act Qik
    //List1:i=1...n Lista:k=1...σ resources, Lista3:quantity per time
period (size=time)
    public List<List<List<Integer>>> profiles;
    public int [][] fw;
    public int fitnessValue;

    public Chromosome(ArrayList<Activity> actsList, int numRes, int
numJobs) {
        this.actsList = actsList;
        this.partition = new int [numRes][numJobs]; //stiles oi diamerisis
        this.profiles = new ArrayList();
        this.fw = new int [numJobs][numJobs];
    }
}

public class Activity implements Cloneable {
    private int ID;
    private int durationTotal;
    public ArrayList<Integer> durations;
    private int ESfw;
    private int LSfw;
    private int ST;
    private int FT;
    private int tallocMin;
    public ArrayList <Integer> effort;
    public ArrayList <Integer> lags;
    public ArrayList<Integer> resources;
    public ArrayList<Activity> predecessors;
    public Map <Integer,Integer> succLags;
    public ArrayList numPartitionTotal; //Qik,total
    public List<List<Integer>> profile;
    public Map <Integer,Integer> succLagsID;
    public Map <Integer,Double> succLagsPercent;

    public Activity(int ID) {
        this.ID = ID;
        this.effort = new ArrayList();
        this.durationTotal = 0;
        this.ESfw=0;
        this.LSfw =0;
        this.numPartitionTotal = new ArrayList();
        this.lags= new ArrayList ();
        this.resources= new ArrayList();
        this.predecessors=new ArrayList<Activity>();
        this.succLags = new HashMap();
        this.tallocMin = 0;
```

```
        this.profile = new ArrayList();
        this.durations=new ArrayList();
        this.succLagsID=new HashMap();
        this.succLagsPercent= new HashMap();
    }
    public void setDurationTotal(int durationTotal) {
        this.durationTotal = durationTotal;
    }
    public void addPred (Activity curAct)
    {
    this.predecessors.add(curAct);
    }
    public void addSuccLags (int id, int lag) {
        this.succLags.put(id, lag);
    }
    public void addSuccLagsID (int curID, int lag) {
        this.succLagsID.put(curID, lag);
    }
    public void addSuccLagsPercent (int curID, double lag) {
        this.succLagsPercent.put(curID, lag);
    }
    public int getID() {
        return ID;
    }
    public int getDurationTotal() {
        return durationTotal;
    }
    public void setESfw(int ESfw) {
        this.ESfw = ESfw;
    }
    public int getESfw() {
        return ESfw;
    }
    public void setLSfw(int LSfw) {
        this.LSfw = LSfw;
    }
    public int getLSfw() {
        return LSfw;
    }
    public void setST(int ST) {
        this.ST = ST;
    }
    public void setFT(int FT) {
        this.FT = FT;
    }
    public int getST() {
        return ST;
    }
    public int getFT() {
        return FT;
    }
    public void setTallocMin(int tallocMin) {
        this.tallocMin = tallocMin;
    }
    public int getTallocMin() {
        return tallocMin;
    }
    public int getKey(Activity act, Activity succ){
        int lag=-1000000000;
        Set mapSet = (Set) succLags.entrySet();
        Map.Entry mapEntry = (Map.Entry) succLags.entrySet();
        if (act.succLags.equals(succ))
```

```
            lag = (int) mapEntry.getKey();

        return lag;

    }
    public static List<Activity> cloneList(List<Activity> list) throws
CloneNotSupportedException {
        List<Activity> clone = new ArrayList<Activity>(list.size());
        for(Activity item: list) clone.add((Activity) item.clone());
        return clone;
    }
 }

 public class StatusSSGSu {
    public int [][] curAvailableResources;
    public ArrayList <Activity> eligibleActs;
    private int IDselectedAct;
    private int selectedTime;
    public ArrayList <Activity> scheduledAct;
    public ArrayList <Activity> U;
    public ArrayList <Activity> unscheduledAct;

    public StatusSSGSu(int stage, int numResources, int TimeHoriz) {

        this.eligibleActs = new ArrayList ();
        this.scheduledAct   = new ArrayList();
        this.curAvailableResources = new int [numResources][TimeHoriz];
        this.U = new ArrayList();
        this.unscheduledAct = new ArrayList();

    }
    public int getIDselectedAct() {
        return IDselectedAct;
    }
    public int getSelectedTime() {
        return selectedTime;
    }
    public void setIDselectedAct(int IDselectedAct) {
        this.IDselectedAct = IDselectedAct;
    }
    public void setSelectedTime(int selectedTime) {
        this.selectedTime = selectedTime;
    }
 }

public class extraMethods {
    public static int [][] calcFW(Chromosome chromosome, int JobNum){
        //initialize matrix FW : dij=0, if i=j  dij=lij(with lags), -inf
otherwise
        //Do:
        int floyd[][]=new int [JobNum][JobNum];
        for (int l=0; l<JobNum ;l++) {
            for(int c=0; c<JobNum ;c++){
                if(chromosome.actsList.get(c).getID()==chromosome.
actsList.get(l).getID()){
                    floyd[chromosome.actsList.get(l).getID()]
[chromosome.actsList.get(c).getID()] = 0 ;
                }
                else if (chromosome.actsList.get(l).succLagsID.
containsKey(chromosome.actsList.get(c).getID())==true){
```

```
                    floyd[chromosome.actsList.get(l).getID()]
[chromosome.actsList.get(c).getID()] = chromosome.actsList.get(l).
succLagsID.get(chromosome.actsList.get(c).getID());
                }
                else{
                    floyd[chromosome.actsList.get(l).getID()]
[chromosome.actsList.get(c).getID()]=-1000000000;
                }
            }
        }
        for(int k=0;k<JobNum;k++){
            for(int i=0;i<JobNum;i++){
                for(int j=0;j<JobNum;j++){
                    if(floyd[i][k]+floyd[k][j]>floyd[i][j])
                        floyd[i][j] = floyd[i][k]+floyd[k][j];
                }
            }
        }
        return floyd;
    }

    public void calcPred(Chromosome chromosome, int jobNum){
        for (int i=0;i<jobNum;i++){
            for(int j=0;j<jobNum;j++){
                if ((chromosome.actsList.get(i).getID()!
=chromosome.actsList.get(j).getID() && chromosome.
fw[chromosome.actsList.get(j).getID()][chromosome.actsList.get(i).getID()]
>= 0)|| (chromosome.fw[chromosome.actsList.get(j).getID()]
[chromosome.actsList.get(i).getID()] == 0 && chromosome.
fw[chromosome.actsList.get(i).getID()][chromosome.actsList.get(j).getID()]
< 0 && chromosome.fw[chromosome.actsList.get(i).getID()]
[chromosome.actsList.get(j).getID()] >= -100000000))
                    chromosome.actsList.get(i).addPred
(chromosome.actsList.get(j));
            }
        }
    }
}
```