ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΒΙΟΜΗΧΑΝΙΚΗΣ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΕΠΙΧΕΙΡΗΣΙΑΚΗΣ ΕΡΕΥΝΑΣ

Διπλωματική Εργασία:

Αναδραστικός χρονοπρογραμματισμός έργων με την χρήση γενετικού αλγορίθμου

Πετρίδης Ιάσων

Επιβλέπων Καθηγητής:

Βρασίδας - Ιωάννης Λεόπουλος

Αθήνα, 2014

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

Πετρίδης Ιάσων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω αρχικά τον επιβλέποντα καθηγήτη Β.Λεώπουλο για την καθοδήγησή του και την βοήθειά του στα χρόνια της φοίτησής μου. Θα ήθελα και να ευχαριστήσω τον καθηγητή Κ.Κηρυττόπουλο για την αρχική ιδέα και όλη την υποστήριξη που μου πρόσφερε κατά την φοιτησή μου στην σχολή.

Είμαι ευγνώμων για την καθοδήγηση κατά την εκπόνηση της διπλωματικής στην Ε.Ρόκου.

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου, τον αδερφό μου και τους φίλους μου για τα χρόνια υπομονής και υποστήριξης που μου έχουν προσφέρει.

Περιληψη

Η αβεβαιότητα είναι εγγενής στη διαχείριση του έργου. Τα περισσότερα, αν όχι όλα, τα χρονοπρογράμματα έργων τείνουν να διαφοροποιηθούν και, επομένως, έχουν ανάγκη για επαναπρογραμματισμό, έτσι ώστε να συνεχίσει το έργο. Η χρήση των κανονικών μεθόδων χρονοπρογραμματισμού σε τέτοιες περιπτώσεις τείνει να δημιουργήσει νέα χρονοπρογράμματα που είναι είτε πάρα πολύ διαφορετικά από το αρχικό χρονοπρόγραμμα ή σχετικά ασταθή σε περαιτέρω διαφοροποιήσεις.

Ο στόχος της παρούσας διπλωματικής είναι να προταθεί μια αναδραστική μέθοδος χρονοπρογραμματισμού η οποία μπορεί να χρησιμοποιηθεί για την αναθεώρηση ή την εκ νέου βελτιστοποίηση ενός ήδη κατασκευασμένου αρχικού χρονοπρογράμματος όταν συμβαίνουν απροσδόκητα γεγονότα, λαμβάνοντας υπ 'όψιν το υπάρχον χρονοπρόγραμμα και τις δραστηριότητες που έχουν ήδη γίνει ή βρίσκονται σε εξέλιξη. Η προτεινόμενη προσέγγιση βασίζεται σε ένα υβριδικό αλγόριθμο που συνδυάζει τον γενετικό αλγόριθμο με το σειριακό σύστημα παραγωγής χρονοπρογράμματος και στοχεύει στην παροχή γρήγορης ανταπόκρισης και πολλαπλών σενάρια επίλυσης στον διαχειριστή του έργου, έτσι ώστε να χειρίζεται αποτελεσματικά τις πραγματοποιημένες ή μελλούμενες διαφοροποιήσεις.

Abstract

Uncertainty is inherent in project management. Most, if not all, project schedules tend to get disrupted and are therefore in need of rescheduling in order to continue. The use of normal scheduling methods in such cases tends to create new schedules that are either too different from the baseline schedule or relatively unstable to further disruptions.

The objective of this thesis is to propose a reactive scheduling process that may be used to revise or re-optimize a previously developed baseline schedule when unexpected events occur, taking in consideration the existing schedule and the work already done or currently in progress. The proposed approach is based on a hybrid algorithm that combines genetic algorithms with serial schedule generation scheme and aims at providing quick response and multiple solution scenarios to the project manager so as to efficiently handle the experienced or forecasted disruptions.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Αλγορίθμων

1 Εισαγωγή

Η συντριπτική πλειοψηφία των ερευνητικών προσπαθειών στον προγραμματισμό των έργων κατά τα τελευταία αρκετά χρόνια έχει επικεντρωθεί στην ανάπτυξη αναλυτικών και υποβέλτιστων διαδικασίων για τη δημιουργία ενός αρχικού χρονοπρογράμματος υποθέτοντας πλήρη στοιχεία και ένα ντετερμινιστικό περιβάλλον (Herroelen and Leus, 2004). Στην πράξη, κατά τη διάρκεια της εκτέλεσης, τα έργα αποτελούν συχνά αντικείμενο σημαντικής αβεβαιότητας, η οποία μπορεί να οδηγήσει σε πολυάριθμες μικρές ή μεγάλες διαφοροποιήσεις στο χρονοπρόγραμμα.

Στον προγραμματισμό των έργων, η αβεβαιότητα μπορεί να πάρει πολλές διαφορετικές μορφές. Οι διάρκειες των δραστηριοτήτων μπορεί να μην εκτιμηθούν σωστά, οι πόροι μπορούν να γίνουν μη-διαθέσιμοι (π.χ. βλάβες μηχανών, ασθένειες εργατών, κλπ.), το έργο μπορεί να διακοπεί λόγω κακών καιρικών συνθηκών, νέες απρόβλεπτες δραστηριότητες μπορεί να προστεθούν λόγω αλλαγών στο σκοπό του έργου, κλπ. Όλες αυτές οι μορφές αβεβαιότητας μπορούν να οδηγήσουν στην παραβίαση του αρχικού χρονοδιαγράμματος του έργου. Σε γενικές γραμμές, η διαχείριση του έργου θέλει να αποφύγει αυτές τις αποκλίσεις από το χρονοδιάγραμμα. Αυτό μπορεί να επιτευχθεί με τη δημιουργία ενός αρχικού χρονοπρογράμματος με προληπτικό χαρακτήρα, προσπαθώντας να προβλέψουν ορισμένους τύπους διαταραχών, έτσι ώστε να ελαχιστοποιηθούν οι επιπτώσεις τους, εφόσον προκύψουν. Εάν το χρονοπρόγραμμα διαφοροποιηθεί παρά αυτές τις προληπτικές προσπάθειες σχεδιασμού, θα χρειαστεί μια αναδραστική πολιτική προγραμματισμού για να επισκευάσει το ανέφικτο χρονοπρόγραμμα (Deblaere et al., 2011).

Ο στόχος της παρούσας εργασίας είναι να προτείνει μια αναδραστική μέθοδο προγραμματισμού που μπορεί να χρησιμοποιηθεί για την αναθεώρηση ή την εκ νέου βελτιστοποίηση ενός αρχικού χρονοπρογράμματος όταν συμβαίνουν σε αυτό απροσδόκητα γεγονότα. Πιο συγκεκριμένα, έχοντας ένα αρχικό χρονοπρόγραμμα που έχει ακολουθηθεί και ενημερωθεί εγκαίρως με πραγματικά δεδομένα από τον διαχειριστή του έργου, σε μια συγκεκριμένη χρονική στιγμή, μία ή περισσότερες διαφοροποιήσεις στο πρόγραμμα έχουν παρατηρηθεί ή προβλεφθεί λόγω νέων δεδομένων ή / και αλλαγών σε μία ή περισσότερες περιβαλλοντικές μεταβλητές. Για παράδειγμα, αναμένεται έλλειψη των υλικών λόγω των επερχόμενων απεργιών. Στην περίπτωση αυτή, το αρχικό χρονοπρόγραμμα που είχε αναπτυχθεί καθίσταται ανέφικτο κατά τη διάρκεια εκτέλεσης του έργου, λόγω της εμφάνισης διαφοροποιήσεων ενός ή περισσότερων πόρων ή διαρκειών δραστηριοτήτων. Ως εκ τούτου, χρειαζόμαστε μια αναδραστική πολιτική που υπαγορεύει πώς να επανέλθει το χρονοπρόγραμμα σε ένα εφικτό που να αποκλίνει όσο το δυνατόν λιγότερο από το αρχικό και να επιλύει τους ανέφικτους πλέον περιορισμούς πόρων που προκλήθηκαν από τις διαφοροποιήσεις που εμφανίστηκαν κατά την εκτέλεση του έργου.

Σε αυτή την εργασία θα περιγράφθει για την αναδραστική μέθοδο προγραμματισμού έργων μια νέα μεταευρετική μέθοδος επίλυσης που μπορεί να χρησιμοποιηθεί για την «επιδιόρθωση» ενός αρχικού χρονοπρογράμματος έργου με περιορισμένους πόρους που υπέστη διαφοροποιήσεις στις απαιτήσεις των πόρων ή/και στις διάρκειες των δραστηριοτήτων. Ο στόχος της είναι να δημιουργήσει ένα νέο χρονοπρόγραμμα που να είναι εφικτό με βάση τα νέα δεδομένα από τις διαφοροποιήσεις αλλά και να μην αποκλίνει πολύ από το αρχικό χρονοπρόγραμμα. Η προτεινόμενη προσέγγιση βασίζεται σε έναν

γενετικό αλγόριθμο που παράγει και ελέγχει πολλαπλά νέα χρονοπρογράμματα και βρίσκει το κοντινότερο στο αρχικό χρονοπρόγραμμα από αυτά, ενώ ταυτόχρονα δεν αλλάζει πολύ το χρόνο τέλους του έργου.

Συγκεκριμένα στο κεφάλαιο 2 παρουσιάζεται η βιβλιογραφική επισκόπηση της θεωρίας των έργων καθώς και το θεωρητικό υπόβαθρο που σχετίζεται με τους γενετικούς αλγορίθμους. Στην συνέχεια στο κεφάλαιο 3 περιγράφεται το προς επίλυση πρόβλημα και στο κεφάλαιο 4 η μοντελοποίηση του προβλήματος και η προτεινόμενη διαδικασία επίλυσης αυτού. Στο κεφάλαιο 5 περιγράφεται αναλυτικά η υλοποίηση της προτεινόμενης λύσης, εστιάζοντας στον προτεινόμενο γενετικό αλγόριθμο.

Τα αποτελέσματα της επίλυσης παρουσιάζονται στο κεφάλαιο 6, ενώ στο κεφάλαιο 7 επιδεικνύεται η υλοποιηθείσα διεπαφή που παρέχεται στον χρήστη ως επιπρόσθετο στο Ms Project, για την εισαγωγή και παραμετροποίηση των δεδομένων εισόδου και την οπτικοποίηση των ευρισκόμενων λύσεων.

Τέλος, στο κεφάλαιο 8 παρουσιάζονται τα συμπεράσματα που προέκυψαν και προτείνονται προεκτάσεις της παρούσας προσπάθειας για μελλοντική έρευνα.

2 Βιβλιογραφική Επισκόπηση

2.1 Έργο (Project)

Ως έργο ορίζεται, «μια μοναδική δέσμη συντονισμένων και ελεγχόμενων δραστηριοτήτων με διακριτό σημείο έναρξης και λήξης, που λαμβάνουν χωρά 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). Τα δύο σημαντικότερα στοιχεία που πρέπει να έχει ένα οποιαδήποτε εγχείρημα για να θεωρηθεί ως έργο είναι η προσωρινότητα και η μοναδικότητά του (Project Management Institute., 2013).

2.1.1 Προσωρινότητα

Το στοιχείο της προσωρινότητας υποδηλώνει ότι κάθε έργο έχει καθορισμένη αρχή και τέλος. Το τέλος ενός έργου επιτυγχάνεται είτε όταν οι στόχοι του έργου έχουν επιτευχθεί είτε όταν το έργο πρέπει να σταματήσει επειδή οι στόχοι του δεν μπορούν να επιτευχθούν είτε όταν δεν υπάρχει πια ανάγκη για το έργο. Η προσωρινότητα του έργου δεν σημαίνει φυσικά ότι το έργο είναι μικρής διάρκειας. Ένα έργο μπορεί να διαρκέσει από μερικές μέρες μέχρι αρκετά χρόνια.

Το σίγουρο πάντως είναι ότι η διάρκεια ενός έργου είναι καθορισμένη, όση και αν είναι αυτή η διάρκεια. Η διάρκεια ενός έργου υπολογίζεται από τις επιμέρους δραστηριότητες (activities) που το αποτελούν. Δραστηριότητες είναι συγκεκριμένες εργασίες οι οποίες χρειάζονται να γίνουν για να επιτευχθούν οι στόχοι του έργου.

Οι δραστηριότητες έχουν τρία κύρια χαρακτηριστικά: την διάρκεια τους, την απαίτηση τους σε πόρους (resources) και τις αλληλεξαρτήσεις τους. Οι σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων ουσιαστικά προσδιορίζουν την σειρά εκτέλεσης τους και τον χρονισμό τους, τέτοιου είδους συσχετίσεις μεταξύ δραστηριοτήτων είναι, το τέλος – αρχή (finish – start), όπου για να ξεκινήσει μια δραστηριότητα πρέπει να έχουν πρώτα ολοκληρωθεί όλες οι δραστηριότητες που συνδέονται με αυτήν, η αρχή – αρχή (start – start), όπου για να ξεκινήσει μια δραστηριότητα πρέπει πρώτα να έχουν ξεκινήσει οι δραστηριότητες που συνδέονται με αυτήν, τέλος – τέλος (finish – finish), όπου μια δραστηριότητα για να ολοκληρωθεί πρέπει να ολοκληρωθούν και οι δραστηριότητες που συνδέονται με αυτήν, και αρχή – τέλος (start – finish), όπου μια δραστηριότητα για να ολοκληρωθεί την χρονική στιγμή πρέπει να ξεκινήσουν οι δραστηριότητες που συνδέονται με αυτήν.

2.1.2 Μοναδικότητα

Το στοιχείο της μοναδικότητας υποδηλώνει ότι το αποτέλεσμα κάθε έργου είναι μοναδικό και διαφοροποιείται από το αποτέλεσμα οποιουδήποτε άλλου έργου. Ομοιότητες μεταξύ αποτελεσμάτων διαφορετικών έργων μπορεί να υπάρχουν, αλλά αυτό δεν αναιρεί την μοναδικότητα κάθε έργου. Για παράδειγμα, δύο κτήρια μπορεί να έχουν ίδια αρχιτεκτονική

και να έχουν κατασκευαστεί από τα ίδια υλικά, αλλά βρίσκονται σε διαφορετικές περιοχές ή είχαν διαφορετικό εργατικό δυναμικό.

2.2 Διαχείριση Έργων (Project Management)

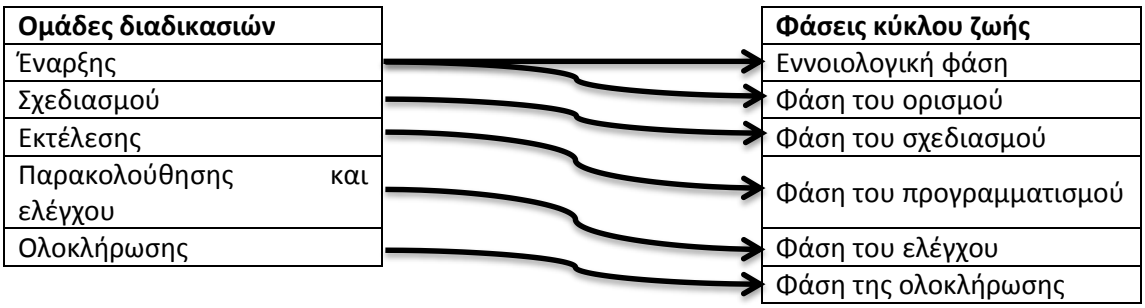
Η διαχείριση έργων είναι η εφαρμογή γνώσεων, ικανοτήτων, εργαλείων και τεχνικών στις δραστηριότητες ενός έργου για να καλύψει τις απαιτήσεις του έργου. Αυτό επιτυγχάνεται μέσω της κατάλληλης εφαρμογής και ολοκλήρωσης πέντε ομάδων διαδικασιών (Project Management Institute., 2013):

- Έναρξης
- Σχεδιασμού
- Εκτέλεσης
- Παρακολούθησης και ελέγχου
- Ολοκλήρωσης

Αυτές οι ομάδες διαδικασιών εμφανίζονται καθ'όλη τη διάρκεια του έργου και παραλληλίζονται με τις φάσεις του κύκλου ζωής του. Έχουμε συνεπώς τις εξής φάσεις (Demeulemeester and Herroelen, 2002):

- Εννοιολογική φάση
- Φάση του ορισμού
- Φάση του σχεδιασμού
- Φάση του προγραμματισμού
- Φάση του ελέγχου
- Φάση της ολοκλήρωσης

Ο παραλληλισμός των φάσεων με τις ομάδες διαδικασιών φαίνεται στο Σχήμα 1.



Σχήμα 1: Παραλληλισμός ομάδων διαδικασιών και φάσεων κύκλου ζωής (Demeulemeester and Herroelen, 2002, Project Management Institute., 2013).

2.2.1 Φάσεις κύκλου ζωής ενός έργου

2.2.1.1 Εννοιολογική φάση

Στην πρώτη φάση του κύκλου ζωής ενός έργου προσδιορίζεται από τον άνθρωπο, εταιρεία ή οργανισμό, που θα διαθέσει το κεφάλαιο, μία ανάγκη που πρέπει να ικανοποιηθεί. Αυτή η ανάγκη αποτελεί τον στόχο που θα πρέπει το έργο να ικανοποιήσει. Φυσικά αυτή η ανάγκη είναι ακόμη σχετικά απροσδιόριστη στη αρχή της εννοιολογικής φάσης. Γνωρίζουμε ότι χρειαζόμαστε ένα έργο με συγκεκριμένο στόχο για μια συγκεκριμένη ανάγκη αλλά δεν

γνωρίζουμε το συγκεκριμένο έργο που θα καλύψει αυτά τα στοιχεία. Για παράδειγμα, μια εταιρεία θέλει να επεκτείνει τις δουλειές τις σε μια ξένη χώρα. Αυτόν το στόχο μπορεί να τον πετύχει μέσω πολλών τρόπων: είτε ένα εργοστάσιο που θα παράγει τα προϊόντα της στην ξένη χώρα, είτε μια αποθήκη μέσω της οποία θα διανέμει τα προϊόντα που φτιάχνει τώρα στην ξένη χώρα, είτε απλώς να αγοράσει μια ήδη έτοιμη εταιρεία στην ξένη χώρα και να χρησιμοποιήσει αυτήν. Πρέπει λοιπόν να επιλεχθεί ο καταλληλότερος τρόπος με τον οποίο θα επιτευχθεί ο στόχος του έργου μας ή ακόμη και να αλλάξει εντελώς ο στόχος, π.χ. να μην επεκταθεί σε ξένη χώρα αλλά να ενισχύσει την θέση της στην χώρα της αν αυτό είναι πιο συμφέρον (Demeulemeester and Herroelen, 2002).

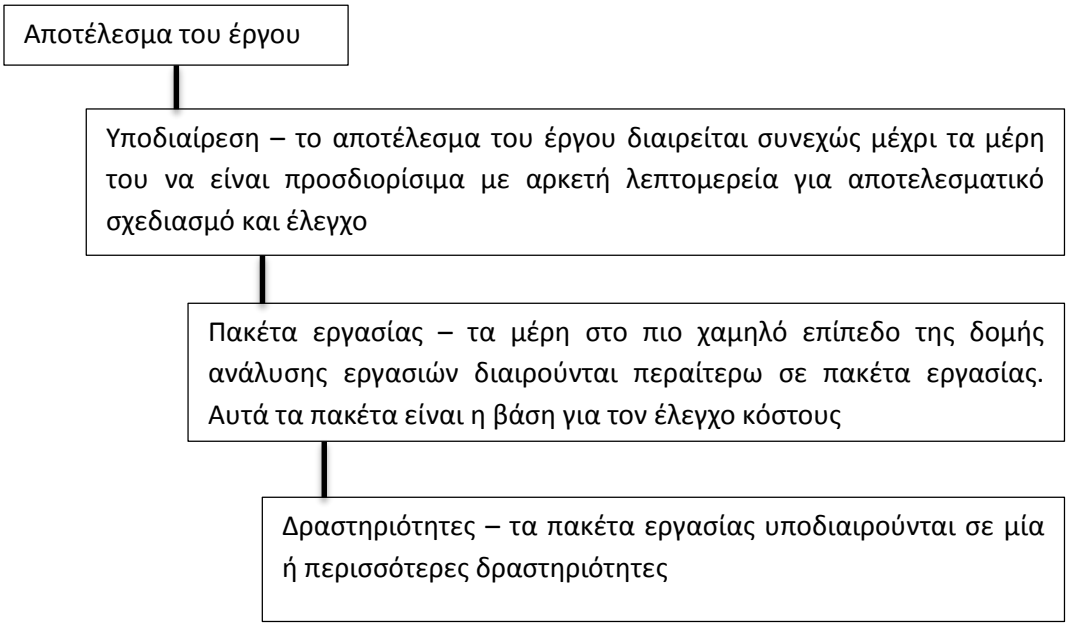
2.2.1.2 Φάση ορισμού

Κατά την φάση του ορισμού, υπολογίζονται με μεγαλύτερη σαφήνεια οι στόχοι, η έκταση και η στρατηγική του έργου. Οι στόχοι του έργου θα πρέπει να είναι σαφής και μετρήσιμοι. Η έκταση του έργου προσδιορίζει με σαφήνεια το περιεχόμενο της εργασίας καθώς και το αποτέλεσμα του έργου. Με αυτό το τρόπο ο πελάτης γνωρίζει ακριβώς τι να περιμένει ως αποτέλεσμα όταν ολοκληρωθεί το έργο. Τέλος, επιλέγεται μία στρατηγική με την οποία θα εκτελεστεί το έργο, η οποία έχει προέλθει από ανάλυση των οικονομικών, τεχνολογικών, νομικών, γεωγραφικών και κοινωνικών παραγόντων που μπορεί να επηρεάσουν το έργο(Demeulemeester and Herroelen, 2002).

2.2.1.3 Φάση σχεδιασμού

Στην φάση του σχεδιασμού προσδιορίζονται οι δραστηριότητες του έργου, εκτιμούνται οι διάρκειες και οι ανάγκες σε πόρους, προσδιορίζεται η διαδοχικότητα των δραστηριοτήτων και προσδιορίζονται οι περιορισμοί του προγράμματος.

Λόγω της πολυπλοκότητας που μπορεί να έχει ένα έργο, είναι χρήσιμο για τον σωστό προσδιορισμό των δραστηριοτήτων να δημιουργηθεί μια δομή ανάλυσης εργασιών (Work Breakdown Structure – WBS). Σκοπός του WBS είναι να διαιρέσει το έργο σε μείζονα κομμάτια, που λέγονται πακέτα εργασίας (work packages), και να προσδιορίσει σε καθένα από αυτά τα πακέτα τις δραστηριότητες που πρέπει να γίνουν για να επιτευχθούν οι στόχοι του έργου (Demeulemeester and Herroelen, 2002).



Σχήμα 2: Περίληψη της διαδικασίας δημιουργίας μιας δομής ανάλυσης εργασιών (Demeulemeester and Herroelen, 2002).

Αφού βρούμε τις δραστηριότητες με την χρήση του WBS, βρίσκουμε και ορίζουμε τις διάρκειες και ανάγκες τους σε πόρους, καθώς φυσικά και τις ακριβείς σχέσεις προτεραιότητάς τους. Στον Πίνακας 1 παρουσιάζεται ένα έργο 10 δραστηριοτήτων που έχει μόνο έναν πόρο. Παρατηρούμε ότι η πρώτη και η τελευταία δραστηριότητα έχουν μηδενική διάρκεια καθώς και μηδενικές ανάγκες σε πόρο. Αυτές οι δύο δραστηριότητες είναι πλασματικές και δημιουργούνται για τον καλύτερο χρονοπρογραμματισμό του έργου. Πρακτικά ο έργο του Πίνακας 1 αποτελείται μόνο από 8 δραστηριότητες και οι δύο πλασματικές δραστηριότητες αναπαριστούν την έναρξη και λήξη του έργου. Φυσικά αυτό δεν σημαίνει ότι δεν μπορεί να έχει μια δραστηριότητα μηδενική διάρκεια. Σε γενικές γραμμές οι δραστηριότητες με μηδενική διάρκεια υπάρχουν για να αναπαριστούν συγκεκριμένα γεγονότα κατά την εκτέλεση του έργου, και για αυτό τον λόγο ονομάζονται γεγονότα ή ορόσημα (milestones).

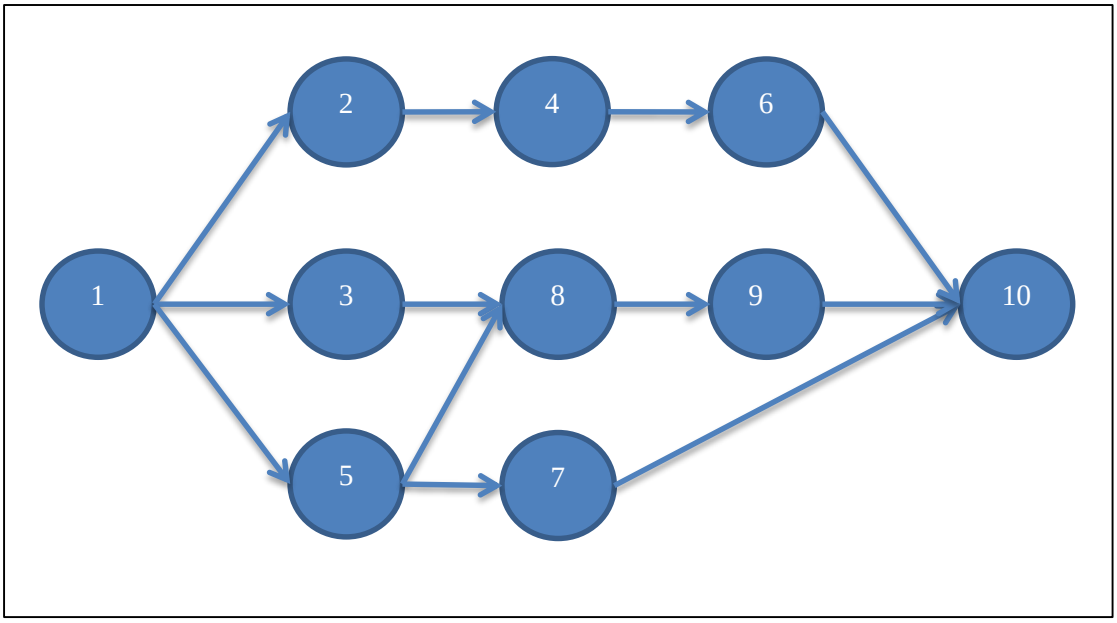
Αριθμός δραστηριότητας	Διάρκεια	Ανάγκες σε πόρους	Διάδοχες δραστηριότητες
1	0	0	2, 3, 5
2	2	2	4
3	7	3	8
4	3	4	6
5	4	4	7, 8
6	8	3	10
7	6	2	10
8	4	3	9
9	2	4	10
10	0	0	-

Πίνακας 1: Ανάλυση δεδομένων ενός έργου 10 δραστηριοτήτων.

Αφού έχουμε ορίσει όλες τις μεμονωμένες δραστηριότητες που θα περιέχει το έργο μας, μπορούμε να τις παραστήσουμε γραφικά στη μορφή ενός δικτύου του έργου. Σε αυτό το

δίκτυο φαίνονται οι μεταξύ σχέσεις αλληλουχίας των δραστηριοτήτων, οι εκτιμώμενες διάρκειες τους ή/και οι ανάγκες τους σε πόρους. Οι πιο συνηθισμένες μορφές δικτύων έργου έχουν την μορφή $G = (N, A)$, αποτελούνται δηλαδή από N κόμβους και A τόξα, και είναι οι εξής δύο (Demeulemeester and Herroelen, 2002):

- Η αναπαράσταση δραστηριότητας σε τόξο, όπου οι δραστηριότητες αναπαριστούνται σαν ένα σύνολο από A τόξα και έχουμε ένα σύνολο N κόμβων που αναπαριστά γεγονότα. Η αναπαράσταση αυτή δίνει καλύτερη έποψη του χρονισμού των συμβάντων.
- Η αναπαράσταση δραστηριότητας σε κόμβο (Σχήμα 3), όπου οι δραστηριότητες αναπαριστούνται σαν ένα σύνολο από N κόμβους και οι μεταξύ τους σχέσεις σαν ένα σύνολο A τόξων. Αυτή η αναπαράσταση είναι η πιο συχνά συναντόμενη και εστιάζει στις ίδιες τις δραστηριότητες και τις αλληλεξαρτήσεις μεταξύ αυτών.



Σχήμα 3: Αναπαράσταση δραστηριότητας σε κόμβο του έργου του Πίνακας 1.

2.2.1.4 Φάση προγραμματισμού

Σε αυτή τη φάση κατασκευάζεται ένα αρχικό χρονοπρόγραμμα το οποίο καθορίζει για κάθε δραστηριότητα χρονικές στιγμές έναρξης και λήξης. Ένα χρονοπρόγραμμα καλείται εφικτό όταν οι δραστηριότητες έχουν προγραμματιστεί να εκτελεστούν με τέτοιο τρόπο, ώστε να μην απαιτούνται καμία χρονική στιγμή περισσότεροι πόροι από τους διαθέσιμους και η σειρά εκτέλεσης τους να είναι σύμφωνη με τις σχέσεις προτεραιότητας που είχαν οριστεί αρχικά. Σε μεγαλύτερο βαθμό αναλύεται αυτή η φάση στην ενότητα του Προγραμματισμού Έργων (Demeulemeester and Herroelen, 2002).

2.2.1.5 Φάση ελέγχου

Αφού έχει δημιουργηθεί το αρχικό χρονοπρόγραμμα, εφαρμόζεται. Αυτό περιλαμβάνει τόσο την εκτέλεση των δραστηριοτήτων έτσι όπως έχουν οριστεί όσο και την παρακολούθησή τους για να βεβαιωθεί ότι η εκτέλεση του έργου, πραγματοποιείται εντός του προϋπολογισμού, στον προσυμφωνηθέντα χρόνο και ποιότητα.

2.2.1.6 Φάση ολοκλήρωσης

Κατά την φάση της ολοκλήρωσης παραδίδεται το αποτέλεσμα του έργου στον πελάτη (Demeulemeester and Herroelen, 2002).

2.3 Προγραμματισμός έργων (Project Scheduling)

Όπως αναφέρθηκε στην ενότητα της διαχείρισης έργων, ο προγραμματισμός έργων έχει σαν στόχο την δημιουργία ενός αρχικού χρονοπρογράμματος, του οποίου σκοπός είναι να καθοδηγήσει την εκτέλεση του έργου. Το αρχικό (baseline) χρονοπρόγραμμα πρέπει να τηρεί τους περιορισμούς της διαδοχικότητας και των πόρων, αλλά πρέπει να είναι και στιβαρό (robust). Στιβαρό σημαίνει ότι τυχόν αλλαγές που θα παρουσιαστούν σε αυτό κατά την φάση του ελέγχου, αφού έχει ήδη ξεκινήσει να εκτελείται, δεν θα προκαλέσουν μεγάλη αλλαγή στην έκταση του έργου. Μία από τις συχνότερες μεθόδους κατασκευής ενός αρχικού χρονοπρογράμματος είναι η μέθοδος του κρίσιμου δρόμου (critical path method – CPM) (Demeulemeester and Herroelen, 2002).

2.3.1 Μέθοδος του κρίσιμου δρόμου

Στόχος του CPM είναι η εύρεση μίας σειράς – δρόμου από δραστηριότητες, οι οποίες δεν μπορούν να αλλάξουν οι χρονικές στιγμές έναρξης και λήξης χωρίς να αλλάξει η συνολική διάρκεια του έργου. Η μέθοδος έχει δύο μέρη: τους προς τα μπροστά και τους προς τα πίσω υπολογισμούς (Demeulemeester and Herroelen, 2002).

Στο πρώτο μέρος της μεθόδου υπολογίζονται οι νωρίτερα δυνατοί χρόνοι έναρξης και λήξης των δραστηριοτήτων του έργου. Ξεκινώντας από την πρώτη δραστηριότητα, η οποία είναι η πλασματική δραστηριότητα της αρχής, και ορίζει τους νωρίτερα δυνατούς χρόνους έναρξης και λήξης αυτής ίσους με μηδέν.

Κινώντας προς τα κάτω, προς την τελευταίαν δραστηριότητα, την πλασματική δραστηριότητα του λήξης, για κάθε δραστηριότητα ελέγχεται αν οι προαπαιτούμενες τις δραστηριότητες έχουν προγραμματιστεί, έχουν δηλαδή χρόνο έναρξης και λήξης. Επιλέγοντας από τις προαπαιτούμενες δραστηριότητες τον μέγιστο νωρίτερα δυνατό χρόνο λήξης τους και ορίζοντας αυτόν σαν νωρίτερα δυνατό χρόνο έναρξης της δραστηριότητάς μας. Στην συνέχεια ορίζοντας τον νωρίτερα δυνατό χρόνο λήξης ίσο με το άθροισμα του νωρίτερα δυνατού χρόνου εκκίνησής της με την διάρκειά της. Η διαδικασία τελειώνει όταν βρεθεί ο νωρίτερα δυνατός χρόνος έναρξης της τελευταίας δραστηριότητας (Demeulemeester and Herroelen, 2002).

Στο δεύτερο μέρος της μεθόδου υπολογίζονται οι αργύτεροι δυνατοί χρόνοι έναρξης και λήξης των δραστηριοτήτων του έργου. Ξεκινώντας από την τελευταία δραστηριότητα ορίζονται οι αργύτεροι δυνατοί χρόνοι έναρξης και λήξης της δραστηριότητας τέλους ίσοι μεταξύ τους.

Στην συνέχεια κινώντας προς τα επάνω προς την πρώτη δραστηριότητα και για κάθε δραστηριότητα βρίσκεται ο μικρότερος από τους αργύτερους δυνατούς χρόνους λήξης των διάδοχων δραστηριοτήτων της και ορίζεται αυτός σαν τον αργότερο δυνατό χρόνο λήξης της. Για να υπολογιστεί ο αργότερος δυνατός χρόνος έναρξης της, αφαιρείται από τον αργότερο δυνατό χρόνο λήξης, η διάρκεια της. Η διαδικασία τελειώνει όταν βρεθεί ο

αργότερος δυνατός χρόνος έναρξης της πρώτης δραστηριότητας (Demeulemeester and Herroelen, 2002).

Έχοντας λοιπόν τους νωρίτερο και αργότερο δυνατούς χρόνους έναρξης όλων των δραστηριοτήτων μπορούμε να βρούμε το ελεύθερο περιθώριο που έχει κάθε δραστηριότητα. Οι δραστηριότητες που έχουν μηδενικό περιθώριο αποτελούν τμήμα του κρίσιμου δρόμου. Όπως είναι λογικό ο κρίσιμος δρόμος ξεκινάει και τελειώνει με την πρώτη και την τελευταία δραστηριότητα του έργου αντίστοιχα. Είναι πιθανόν ένα έργο να έχει παραπάνω από έναν κρίσιμο δρόμο.

Σημαντικό μειονέκτημα του CPM είναι ότι δεν λαμβάνει υπ'όψιν τους πόρους και συνεπώς και τους περιορισμούς που υπάρχουν σε αυτούς. Λύση σε αυτό το μειονέκτημα έρχεται να δώσει το πρόβλημα του προγραμματισμού έργων με περιορισμένους πόρους (Resource – Constrained Project Scheduling Problem – RCPSP).

2.4 RCPSP

Οι δραστηριότητες έχουν τρία κύρια χαρακτηριστικά: την διάρκεια τους, την απαίτηση τους σε πόρους και τις σχέσεις προτεραιότητας τους. Με το CPM είναι δυνατός ο χρονοπρογραμματισμός ενός έργου με βάση τη διάρκεια και την διαδοχικότητα των δραστηριοτήτων του. Στην περίπτωση όμως που οι διαθέσιμοι πόροι είναι περιορισμένοι και λιγότεροι από αυτούς που απαιτούνται για την εκτέλεση των δραστηριοτήτων στους χρόνους που προσδιορίζονται από το CPM, τότε απαιτείται διαφορετική προσέγγιση για τον εφικτό προγραμματισμό των δραστηριοτήτων του έργου (Demeulemeester and Herroelen, 2002).

2.4.1 Πόροι

Οι πόροι κατατάσσονται σε τρεις κύριες κατηγορίες: ανανεώσιμοι, μη ανανεώσιμοι και διπλά περιορισμένοι(Demeulemeester and Herroelen, 2002). Οι **ανανεώσιμοι πόροι** (renewable resources), είναι διαθέσιμοι για κάθε χρονική περίοδο, με τον περιορισμό να αφορά τη συνολική διαθεσιμότητα κάθε πόρου ανά χρονική περίοδο. Οι ανανεώσιμοι πόροι δεν καταναλώνονται, απλά δεσμεύονται για συγκεκριμένες χρονικές περιόδους και στη συνέχεια είναι ξανά διαθέσιμοι προς χρήση. Οι **μη ανανεώσιμοι πόροι** (non renewable resources είναι διαθέσιμοι για όλη τη διάρκεια του έργου με ένα περιορισμένο διαθέσιμο συνολικό ποσό. Οι πόροι αυτής της κατηγορίας πρακτικά καταναλώνονται, καθώς η χρησιμοποιούμενη ποσότητα αφαιρείται μόνιμα από το διαθέσιμο συνολικό μέγεθος κατανάλωσης. Οι **διπλά περιορισμένοι πόροι** (doubly-constrained resources), είναι περιορισμένοι τόσο για κάθε χρονική περίοδο, όσο και για τη συνολική διάρκεια του έργου.

2.4.2 Εννοιολογική διατύπωση

Το RCPSP μπορεί να μοντελοποιηθεί εννοιολογικά με τον ακόλουθο τρόπο (Demeulemeester and Herroelen, 2002):

$$\min f_n \qquad (2\text{-}1)$$

Υπόκεινται σε

$$f_i \le f_j - d_j \qquad \text{for all} (i,j) \in A \qquad (2\text{-}2)$$

$$f_1 = 0 \qquad (2\text{-}3)$$

$$\sum_{i \in S_t} r_{ik} \le a_k \qquad \text{for } k = 1, \dots, m \text{ and } t = 1, \dots, f_n \qquad (2\text{-}4)$$

Αυτή την εννοιολογική διατύπωση έχει συνολικώς οκτώ μεταβλητές:

- ***A*** είναι το σύνολο των δραστηριοτήτων του έργου, στο οποίο περιέχονται και οι πλασματικές δραστηριότητες που ορίζουν την αρχή και το τέλος του έργου,
- ***n*** είναι η πλασματική δραστηριότητα του λήξης,
- ***m*** είναι το σύνολο των ανανεώσιμων πόρων του έργου,
- f_i είναι ο χρόνος λήξης της δραστηριότητας ***i***,
- d_i είναι η διάρκεια της δραστηριότητας ***i***,
- a_k είναι η διαθεσιμότητα του ανανεώσιμου πόρου ***k***,
- r_{ik} είναι η απαίτηση της δραστηριότητας ***i*** στον πόρο ***k***,
- S_t είναι το σύνολο των δραστηριοτήτων που είναι σε εξέλιξη την χρονική στιγμή ***t***.

Το προγραμματιστικό μοντέλο αποτελείται από τέσσερις εξισώσεις: (2-1) μέχρι (2-4). Η εξίσωση (2-1) αποτελεί την αντικειμενική συνάρτηση του μοντέλου, η οποία ελαχιστοποιεί το χρόνο λήξης της πλασματικής συνάρτησης λήξης. Η εξίσωση (2-2) εκφράζει τις σχέσεις διαδοχικότητας των δραστηριοτήτων, ενώ η εξίσωση (2-3) αναγκάζει την πλασματική δραστηριότητα αρχής να τελειώνει στην χρονική στιγμή 0. Τέλος, η εξίσωση (2-4) εκφράζει το γεγονός ότι σε καμία χρονική στιγμή δεν πρέπει να παραβιάζεται η διαθεσιμότητα των πόρων του έργου.

Αυτό το προγραμματιστικό μοντέλο δεν μπορεί να επιλυθεί απευθείας, διότι δεν υπάρχει εύκολος τρόπος να μεταφράσει το σύλονο S_t σε μια γραμμική προγραμματιστική διατύπωση και ως εκ τούτου δεν αποτελεί μοντέλο γραμμικού προγραμματισμού. Χρειάζονται άλλες γραμμικές προγραμματιστικές διατυπώσεις για να διατυπωθούν οι περιορισμοί στους πόρους σε μία σωστή και επιλύσιμη μορφή (Demeulemeester and Herroelen, 2002).

Μεταβλητή	**Έννοια**
A	Σύνολο δραστηριοτήτων. Περιέχει και τις πλασματικές δραστηριότητες που ορίζουν την αρχή και το τέλος.
n	Πλασματική δραστηριότητα λήξης.
m	Σύνολο ανανεώσιμων πόρων.
i	Μια τυχαία δραστηριότητα, που ανήκει στο ***A***.
f_i	Χρόνος λήξης δραστηριότητας ***i.***
d_i	Διάρκεια της δραστηριότητας ***i.***
k	Ένας τυχαίος πόρος, που ανήκει στο ***m***.
α_k	Διαθεσιμότητα του ανανεώσιμου πόρου ***k.***
r_{ik}	Απαίτηση της δραστηριότητας ***i*** στον πόρο ***k.***
t	Μια τυχαία χρονική στιγμή ***t***, κατά την εκτέλεση του έργου.
S_t	Σύνολο των δραστηριοτήτων που είναι σε εξέλιξη την χρονική στιγμή ***t.***

Πίνακας 2: Συνοπτική παρουσίαση μεταβλητών RCPSP.

2.4.3 Προβλήματα

Όπως φάνηκε στη προηγούμενη ενότητα, η αβεβαιότητα είναι κύριο χαρακτηριστικό του χρονοπρογραμματισμού έργων. Όσο λεπτομερής και προσεγμένος και να είναι ο σχεδιασμός ενός έργου, υπάρχουν εκατοντάδες παράγοντες για τους οποίους δεν μπορείς να προετοιμαστεί ο διαχειριστής του έργου επαρκώς από την αρχή, αλλά εμφανίζονται κατά την διάρκεια του έργου. Βίαια φυσικά φαινόμενα, εργατικά ατυχήματα,

κοινωνιοπολιτικά προβλήματα είναι κάποια από τα γεγονότα που μπορεί να συμβούν κατά την διάρκεια του έργου και να προκαλέσουν σε μικρό ή μεγάλο βαθμό κάποια διαφοροποίηση στο χρονοπρόγραμμα.

Για την αποφυγή μεγάλων διαφοροποιήσεων στο χρονοπρόγραμμα ενός έργου υπάρχουν δυο γενικότερες τάσεις στην επιστημονική κοινότητα αλλά και σε ένα βαθμό αντίστοιχα στις χρησιμοποιούμενες πρακτικές, αφενός η προληπτική (proactive), όπου ουσιαστικά προσπαθείται η ενσωμάτωση των αβεβαιοτήτων στις διάρκειες των δραστηριοτήτων κατά την κατασκευή του αρχικού χρονοπρογράμματος και η αναδραστική (reactive), όπου ο προγραμματιστής του έργου καλείται, μετά την διαπίστωση των διαφοροποιήσεων από το αρχικό χρονοπρόγραμμα, να επαναπρογραμματίσει το έργο κατάλληλα (Demeulemeester and Herroelen, 2002).

2.4.3.1 Προληπτική μέθοδος

Η προληπτική μέθοδος έχει ως στόχο τη δημιουργία ενός στιβαρού αρχικού χρονοπρογράμματος (Lambrechts et al., 2008). Υπάρχουν διάφορες στρατηγικές που μπορούν να χρησιμοποιηθούν στην προληπτική μέθοδο από μεθόδους όπως η PERT (Demeulemeester and Herroelen, 2002) εως και ευρετικές και μετα-ευρετικές προσεγγίσεις (γενετικοί αλγόριθμοι, προσομοιωμένης ανώπτυσης κλπ.) στις οποίες λαμβάνεται, με διάφορους τρόπους και σε διαφορετικό κάθε φορά βαθμό, υπόψη η αβεβαιότητα ολοκλήρωσης στον προβλεπόμενο χρόνο. Τακτικές αυτής της κατηγορίες περιλαμβάνουν την εισαγωγή σημείων –δραστηριοτήτων χρονισμού ή την βελτιστοποίηση του χρονοπρογράμματος με στόχο τη μεγιστοποίηση των ελεύθερων περιθωρίων (Demeulemeester and Herroelen, 2002, Van de Vonder et al., 2008).

2.4.3.2 Αναδραστική μέθοδος

Παρόλο που με την χρήση της προληπτικής μεθόδου μπορεί να έχουμε δημιουργήσει ένα αρκετά στιβαρό αρχικό χρονοπρόγραμμα οι αλλαγές που προκύπτουν κατά τη διάρκεια εκτέλεσης τόσο σε διάρκειες όσο και σε διαθεσιμότητες πόρων μπορεί να προκαλέσουν διαφοροποιήσειστέτοιες διαφοροποιήσεις στα δεδομένα του έργου, ώστε το αρχικό χρονοπρόγραμμα να μην μπορεί να χρησιμοποιηθεί μιας και είναι δυνατόν να έχουν προκύψει ανέφικτοι περιορισμοί τόσο στους πόρους όσο και στις συσχετίσεις των δραστηριοτήτων. Σε αυτό το νέο πρόβλημα που προέκυψε κατά την εκτέλεση του έργου έρχεται να δώσει λύση η αναδραστική μέθοδος (Smith, 1995, Demeulemeester and Herroelen, 2002).

Γενικώς, ο αναδραστικός χρονοπρογραμματισμός αναφέρεται στις μετατροπές του χρονοπρογράμματος που χρειάζονται να γίνουν κατά την διάρκεια της εκτέλεσης του έργου λόγω απρόβλεπτων συμβάντων που τροποποίησαν τις επικρατούσες συνθήκες – αρχικά δεδομένα (Herroelen and Leus, 2004). Από την στιγμή που η μέθοδος «αντιδράει» σε διαφοροποιήσεις και αλλαγές του χρονοπρογράμματος κατά την εκτέλεση του έργου, χρειάζεται να εκτελείται σε πραγματικό χρόνο και να δίνει πολύ γρήγορα αποτελέσματα. Υπάρχουν δύο διαφορετικές αναδραστικές μέθοδοι, η καθαρά αναδραστική μέθοδος (purely reactive project scheduling) και η απλή μέθοδος (reactive scheduling) (Lambrechts et al., 2008).

Στην καθαρά αναδραστική μέθοδο δεν έχουμε ένα αρχικό χρονοπρόγραμμα αλλά το έργο προγραμματίζεται σε πραγματικό χρόνο. Κάθε δραστηριότητα προγραμματίζεται σε τυχαία σημεία αποφάσεως *t* τα οποία συμβαίνουν σειριακά μέσα στον χρόνο. Αυτά τα τυχαία σημεία αποφάσεως συμπίπτουν με τους χρόνους λήξης των δραστηριοτήτων και η απόφαση για την εκκίνηση δραστηριοτήτων που έχουν εφικτές διαδοχικότητες και πόρους μπορεί να βασιστεί μόνο σε πληροφορίες που είναι γνωστές μέχρι εκείνη τη στιγμή (Lambrechts et al., 2008).

Η δεύτερη μέθοδος συναντάται και σαν προληπτική – αναδραστική μέθοδος (proactive – reactive scheduling) διότι βασίζεται στη δημιουργία ενός στιβαρού αρχικού χρονοπρογράμματος και εν συνεχεία στην προσαρμογή στις νέες συνθήκες που προκύπτουν με τις ελάχιστες δυνατές αλλαγές. Αυτή η μέθοδος θα χρησιμοποιήθει στην εργασία και θα αναλυθεί εκτενέστερα στην επόμενη ενότητα.

2.5 Προληπτική – αναδραστική μέθοδος

Η ύπαρξη και ο τρόπος χρήσης του αρχικού χρονοπρογράμματος σε αυτή τη μέθοδος είναι τα χαρακτηριστικά που την κάνουν να ξεχωρίζει. Στόχος της δεν είναι μόνο ο επαναχρονοπρογραμματισμός του έργου αλλά και το να γίνει με τέτοιο τρόπο έτσι ώστε να έχει την μικρότερη διαφορά από το αρχικό. Με αυτήν τη μέθοδο συνεπώς οι δραστηριότητες δεν μετακινούνται πάρα πολύ από τις αρχικές θέσεις τους με αποτέλεσμα το χρονοπρόγραμμα απλώς να έχει μετακινηθεί προς μία μετέπειτα ημερομηνία χωρίς να έχει αλλοιωθεί σημαντικά η αρχική μορφή του (Lambrechts et al., 2008).

2.5.1 Εννοιολογική διατύπωση

Έστω ένα έργο με αρχικό χρονοπρόγραμμα S^0 και αρχικούς χρόνους έναρξης s_j^0, όπου j μία τυχαία δραστηριότητα. Κάθε χρονική στιγμή t το προβλεπόμενο χρονοπρόγραμμα S^t προβλέπει το πώς θα εξελικτεί η εκτέλεση του έργου με βάση τις πληροφορίες που γνωρίζει μέχρι εκείνη τη χρονική στιγμή.

Αν τυχόν προκύψουν διαφοροποιήσεις στο χρονοπρόγραμμα και δεν ολοκληρωθούν οι δραστηριότητες που έπρεπε να είχαν ολοκληρωθεί την χρονική στιγμή t τυχόν να χρειάζεται να γίνουν τροποποιήσεις στο χρονοπρόγραμμα για να ανταπεξέλθει στις διαφοροποιήσεις. Συνεπώς τώρα υπάρχει ένα πραγματικό χρονοπρόγραμμα S^T το οποίο θα ολοκληρωθεί στη νέα χρονική στιγμή T.

Για να βρεθεί το πόσο καλό είναι το νέο πραγματικό χρονοπρόγραμμα πρέπει να υπολογιστεί πόσο μικρή είναι η διαφορά $\Delta(S^0, S^T) = \sum_j w_j * |s_j^T - s_j^0|$, όπου w_j είναι το βάρος της δραστηριότητας j σε διαφοροποιήσεις και s_j^T ο χρόνος έναρξης της δραστηριότητας j του πραγματικού χρονοπρογράμματος S^T (Van de Vonder et al., 2007). Η διαφορά $\Delta(S^0, S^T)$ μας δείχνει πόσο αποκλίνει το νέο πραγματικό χρονοπρόγραμμα από το αρχικό που έχουμε. Η διαφορά αυτή είναι ζητούμενο να είναι όσο μικρότερη γίνεται.

Μεταβλητή	Έννοια
S^0	Αρχικό χρονοπρόγραμμα του έργου.
j	Μια τυχαία δραστηριότητα του έργου.
s_j^0	Αρχικός χρόνος έναρξης της δραστηριότητας j.
t	Τυχαία χρονική στιγμή κατά την εκτέλεση του έργου.
S^t	Προβλεπόμενο χρονοπρόγραμμα τη χρονική στιγμή t.
T	Προβλεπόμενος χρόνος λήξης του έργου.
S^T	Προβλεπόμενο χρονοπρόγραμμα για το χρόνο λήξης T.
s_j^T	Χρόνος έναρξης της δραστηριότητας j για το προβλεπόμενο χρονοπρόγραμμα.
w_j	Βάρος της δραστηριότητας j σε διαφοροποιήσεις.

Πίνακας 3: Συνοπτική παρουσίαση μεταβλητών προληπτικής – αναδραστικής μεθόδου.

2.6 Μέθοδοι επίλυσης

Οι αναδραστικές μέθοδοι, όπως φάνηκε στις προηγούμενες ενότητες, έχουν σαν στόχο την εύρεση ενός νέου χρονοπρογράμματος για ένα έργο που υπέστει διαφοροποιήσεις κατά την εκτέλεσή του. Ακόμη και η προληπτική – αναδραστική μέθοδος χρειάζεται ένα νέο χρονοπρόγραμμα για να το συγκρίνει με το αρχικό και να δει κατά πόσο διαφέρει από αυτό. Συνεπώς χρειάζεται να λυθεί ένα νέο πρόβλημα χρονοπρογραμματισμού με τα νέα δεδομένα για το έργο μετά την διαφοροποίηση. Προφανώς αν το αρχικό πρόβλημα ήταν τύπου RCPSP τότε και το νέο πρόβλημα θα είναι τύπου RCPSP.

Η επίλυση του νέου προβλήματος μπορεί να γίνει με οποιαδήποτε μέθοδο χρησιμοποιείται κατά την επίλυση ενός RCPSP, γενικότερα και δεν απαιτείται ιδιαίτερη διαδικασία επίλυσης. Στην παγκόσμια βιβλιογραφία προτείνεται πληθώρα μεθόδων που αφορούν τον αναδραστικό χρονοπρογραμματισμό (Deblaere et al., 2008, Demeulemeester and Herroelen, 2002, Herroelen and Leus, 2004, Van de Vonder et al., 2007, Lambrechts et al., 2008, Deblaere et al., 2011). Οι μέθοδοι αυτές μπορούν να ταξινομηθούν σε τρεις μεγάλες κατηγορίες: τις αναλυτικές(Korf, 1985, Demeulemeester and Herroelen, 1992, Demeulemeester and Herroelen, 2000), τις ευρετικές(Bartusch et al., 1988, Kolisch, 1996b, Kolisch, 1996a, Sprecher et al., 1997, Hartmann and Drexl, 1998, Kyriakidis et al., 2012, Wiesemann et al., 2012) και τις μεταευρετικές μεθόδους(Glover, 1989, Glover, 1990, Hartmann, 1998, Lourenço et al., 2001, Ballestín and Trautmann, 2008, Montoya-Torres et al., 2010). Στη συνέχεια όπου θα αναφέρεται αναδραστική μέθοδο ή αναδραστικός χρονοπρογραμματισμός, θα εννοείται η προληπτική – αναδραστική μέθοδο.

2.6.1 Αναλυτικές μέθοδοι

Οι βασικές αναλυτικές μέθοδοι που χρησιμοποιούνται στον αναδραστικό χρονοπρογραμματισμό είναι η μέθοδος του συστήματος διακλάδωσης (branching scheme) και η μέθοδος του κάτω φράγματος (lower bound) (Deblaere et al., 2011). Αυτές οι μέθοδοι χρησιμοποιούνται κυρίως για προβλήματα με πολλαπλούς τρόπους εκτέλεσης των δραστηριοτήτων (multi-mode problem). Η διαφορά αυτού του προβλήματος από το απλό πρόβλημα που έχουμε παρουσιάσει μέχρι στιγμής είναι το γεγονός ότι κάθε δραστηριότητα έχει πολλαπλούς διαφορετικούς τρόπους με τους οποίους μπορεί να εκτελεστεί, κάθε τρόπος προσδιορίζει συγκεκριμένη διάρκεια για τη δραστηριότητα και διαφορετικό συνδιασμό χρησιμοποιούμενων τύπων πόρων και πλήθους, ενώ οι συσχετίσεις μεταξύ των

δραστηριοτήτων και οι διαθέσιμες ποσότητες ανα τύπο πόρων παραμένουν σταθερές για όλους τους διαφορετικούς τρόπους εκτέλεσης.

2.6.1.1 Σύστημα διακλάδωσης

Στη μέθοδο της διακλάδωσης συνδυάζονται δύο διαφορετικοί αλγόριθμοι: ο πρώτος είναι ο αλγόριθμος του αναδραστικού χρονοπρογραμματισμού και δεύτερος ο αλγόριθμος του συστήματος διακλάδωσης.

Μεταβλητή	Έννοια
t^*	Χρονική στιγμή που συμβαίνει μια τυχαία διαφοροποίηση.
PS_l	Μερικώς εφικτό χρονοπρόγραμμα.
t_l	Χρονική στιγμή αποφάσεως του PS_l.
S_l	Σειρά προγραμματισμένων δραστηριοτήτων από το PS_l.
i	Τυχαία δραστηριότητα από την S_l.
s_i'	Σειρά χρόνων έναρξης της δραστηριότητας i στο νέο χρονοπρόγραμμα.
m_i'	Σειρά τρόπων εκτέλεσης της δραστηριότητας i.
A_l	Σειρά ενεργών δραστηριοτήτων από το PS_l.
P_l	Σειρά εκκρεμών δραστηριοτήτων από το PS_l.
ε_l	Σειρά επιλέξιμων δραστηριοτήτων από το PS_l. Υποσύνολο του P_l.
ε_l'	Σειρά δραστηριοτήτων που δεν έχουν τρόπο εκτέλεσης από το PS_l. Υποσύνολο του ε_l.

Πίνακας 4: Μεταβλητές μεθόδου συστήματος διακλάδωσης.

Ο πρώτος αλγόριθμος αναλόγως με το είδος της διαφοροποίησης που συμβαίνει ορίζει τις νέες μεταβλητές που προέκυψαν, όπως απαίτηση σε πόρους ή διάρκεια, και τρέχει τον δεύτερο αλγόριθμο με $l=1$.

Ο δεύτερος αλγόριθμος λειτουργεί σε εφτά βήματα:

1. Αν η δραστηριότητα που ελέγχεται βρίσκεται στο A_l ο αλγόριθμος μετακινείται στο βήμα 7, αλλιώς υπολογίζει το t_l με βάση τους χρόνους εκκίνησης των δραστηριοτήτων.
2. Υπολογίζονται οι δραστηριότητες του ε_l. Αν δεν εντοπισθεί καμία τότε ο αλγόριθμος μετακινείται στο βήμα 1. Αλλιώς προγραμματίζει τις δραστηριότητες του ε_l.
3. Υπολογίζονται οι διαφορετικοί τρόποι εκτέλεσης της δραστηριότητας.
4. Αν δεν παραμένει κάποιος τρόπος εκτέλεσης προς εξέταση ο αλγόριθμος μετακινείται στο βήμα 7. Διαφορετικά εκτελεί την δραστηριότητα με έναν τρόπο εκτέλεσής της. Αν για κάθε πόρο το σύνολο των αναγκών των ενεργών δραστηριοτήτων είναι μεγαλύτερο από την διαθεσιμότητά του μετακινείται στο βήμα 5, αλλιώς στο βήμα 1 με το ίδιο χρονοπρόγραμμα και με $l=l+1$.
5. Υπολογίζονται οι μικρότερες εναλλακτικές καθυστέρησης.
6. Επιλέγεται μια εναλλακτική καθυστέρησης και ο αλγόριθμος μετακινείται στο βήμα 1 με ένα νέο χρονοπρόγραμμα που έχει την εναλλακτική καθυστέρησης και με $l=l+1$. Αν δεν υπάρχουν εναλλακτικές καθυστέρησης να επιλεχθούν μετακινείται στο βήμα 4.
7. Θετεί το $l=l-1$ και αν $l=0$ σταματάει την μέθοδο, αλλιώς επιστρέφει στο βήμα 6.

2.6.1.2 Κάτω φράγμα

Λαμβάνει ώς είσοδο ένα μερικώς εφικτό χρονοπρόγραμμα. Αν μπορεί να βρεθεί τρόπος εκτέλεσης για κάθε δραστηριότητα έτσι ώστε να διατηρούνται οι περιορισμοί στους πόρους για την δραστηριότητα θέτουμε το νέο μερικώς εφικτό χρονοπρόγραμμα σαν κάτω φράγμα και επαναλαμβάνουμε την διαδικασία. Αν δεν μπορεί να βρεθεί εφικτός τρόπος εκτέλεσης τότε θεωρείται το κάτω φράγμα σαν άπειρο και ολοκληρώνεται η διαδικασία έχοντας υπολογίσει το κατώτερο κάτω φράγμα.

2.6.2 Ευρετικές μέθοδοι

Κύρια ευρετική μέθοδος που χρησιμοποιείται στον αναδραστικό χρονοπρογραμματισμό είναι το σειριακό σύστημα παραγωγής χρονοπρογράμματος (serial – schedule generation scheme – s-SGS) (Kolisch, 1996b, Kolisch, 1996a, Montoya-Torres et al., 2010). Αυτή η μέθοδος χρησιμοποιεί έναν κανόνα προτεραιότητας (priority rule) για να ορίσει την σειρά με την οποία θα προγραμματιστούν οι δραστηριότητες με αποτέλεσμα να οδηγεί σε μοναδική λύση, την οποία καλείται ο διαχειριστής έργου να συγκρίνει με το αρχικό χρονοπρόγραμμα και να αποφασίσει για την χρήση ή μη του νέου χρονοπρογράμματος. Συνηθίζεται να συνδέεται με άλλους αλγορίθμους που τρέχουν αυτή τη μέθοδο πολλαπλές φορές με διαφορετικούς κανονές προτεραιότητας για να βρουν την καλύτερη λύση (Van de Vonder et al., 2007, Montoya-Torres et al., 2010). Τέτοιες μέθοδοι θα παρουσιαστούν στο κεφάλαιο των μεταευρετικών μεθόδων.

2.6.2.1 Σειριακό σύστημα παραγωγής χρονοπρογράμματος

Το σειριακό σύστημα παραγωγής χρονοπρογράμματος ή s-SGS, αποτελείται από τρία βήματα: την αρχικοποίηση, το κύριο μέρος του αλγόριθμου και το αποτέλεσμα. Ο s-SGS είναι μέθοδος επίλυσης του RCPSP, και συνεπώς υπόκεινται στους ίδιους περιορισμούς. Χρειάζονται δυο πλασματικές δραστηριότητες που υποδηλώνουν την έναρξη και τη λήξη του έργου και έχουμε μόνο ανανεώσιμους πόρους. Επιπλέον οι συσχετίσεις μεταξύ των δραστηριοτήτων πρέπει να είναι του τύπου Λήξη – Έναρξη (F – S).

2.6.2.1.1 Αρχικοποίηση

Πριν αρχίσει να λειτουργεί ο αλγόριθμος χρειάζεται να γίνει αρχικοποίηση κάποιον συγκεκριμένων μεταβλητών. Η όλη διαδικασία της μεθόδου γίνεται σε ***n*** στάδια, όπου ***n*** είναι ο αριθμός δραστηριοτήτων του έργου μας, συμπεριλαμβανομένων και των πλασματικών. Η αρχικοποίηση λαμβάνει χώρα στο 1ο στάδιο της μεθόδου.

Ορίζονται σαν s_j και f_j ο χρόνος έναρξης και λήξης μιας δραστηριότητας j αντίστοιχα. Επιπλέον ορίζεται και το σύνολο Sec_g σαν το σύνολο των δραστηριοτήτων που έχουν προγραμματιστεί στο στάδιο g. Θεωρούμε τώρα ότι $s_1 = f_1 = 0$ και $Sec_1 = \{1\}$. Δηλαδή, η πλασματική δραστηριότητα έναρξης έχει χρόνο έναρξης και τέλους τη χρονική στιγμή 0 και είναι η μοναδική δραστηριότητα που έχει προγραμματιστεί στο 1ο στάδιο. Τέλος, Ορίζεται σαν R_k την διαθεσιμότητα του ανανεώσιμου πόρου k σε όλο το έργο.

2.6.2.1.2 Αλγόριθμος

Ο αλγόριθμος εκτελείται στα στάδια ***2*** έως ***n-1***. Στην αρχή κάθε σταδίου υπολογίζει τρία σύνολα: D_g, F_g και $\overline{R}_k(t)$. Το σύνολο D_g είναι το σύνολο των δραστηριοτήτων που μπορούν να επιλεγούν για προγραμματισμό, δηλαδή οι δραστηριότητες αυτές για τις οποίες όλες οι προαπαιτούμενες δραστηριότητες έχουν ήδη προγραμματιστεί. Το σύνολο F_g είναι το

σύνολο όλων των χρόνων λήξης f_i των δραστηριοτήτων που περιέχονται στο σύνολο Sec_g. Τέλος, το $\bar{R}_k(t)$ υπολογίζεται από τον εξής τύπο:

$$\bar{R}_k(t) = R_k - \sum_{\substack{j|s_j \le t \\ f_j > t}} r_{jk} \ , \forall k \in K, t \in F_g$$

'οπου K είναι το σύνολο των ανανεώσιμων πόρων k και r_{jk} είναι η απαίτηση της δραστηριότητας j στον πόρο k. Οι δραστηριότητες j επιλέγονται έτσι ώστε να εκτελούνται κατά τη χρονική στιγμή t, δηλαδή να έχουν χρόνο έναρξης μικρότερο ή ίσο του t και χρόνο λήξης μεγαλύτερο του t.

Στη συνέχεια επιλέγεται μία δραστηριότητα j από το σύνολο D_g με βάση κάποιον κανόνα προτεραιότητας. Στη βιβλιογραφία υπάρχουν διάφοροι κανόνες προτεραιότητας, όμως όπως προκύπτει από συγκριτικές μελέτες οι αποτελεσματικότεροι κανόνες προτεραιότητας είναι (Van de Vonder et al., 2007):

- Νωρίτερος αρχικός χρόνος έναρξης με βάρος (Earliest baseline starting time 1 – EBST1)
- Νωρίτερος αρχικός χρόνος έναρξης χωρίς βάρος (Earliest baseline starting time 2 – EBST2)
- Αργύτερος χρόνο έναρξης (Latest starting time – LST)
- Μεγαλύτερο βάρος (Largest weight – LW)
- Μικρότερος αριθμός δραστηριότητας (Lowest activity number – LAN)
- Τυχαίο (Random – RND)

Οι κανόνες ταξινομούν τις δραστηριότητες σε μία σειρά, άυξουσα ή φθίνουσα, με βάση ένα κριτήριο. Στην συνέχεια επιλέγεται η πρώτη δραστηριότητα μέσα σε αυτή τη σειρά η οποία βρίσκεται στο σύνολο D_g.

Αφού επιλεγεί η δραστηριότητα j που θα προγραμματιστεί σε κάθε στάδιο, βρίσκεται ο νωρίτερος χρόνος έναρξης της ES_j, ο οποίος υπολογίζεται σαν ο μέγιστος χρόνος λήξης των προαπαιτούμενων δραστηριοτήτων της j. Στην συνέχεια βρίσκεται ο μικρότερος χρόνος έναρξης t για την δραστηριότητα j, ο οποίος πρέπει να είναι μεγαλύτερος του ES_j, να βρίσκεται στο σύνολο F_g και η απαίτηση του από πόρους:

$$r_{jk} \le \bar{R}_k(\tau) \ , \forall k \in K, \tau \in [t, t + d_j] \cap F_g$$

δηλαδή η απαίτηση της δραστηριότητας j στον πόρο k να είναι μικρότερη από την διαθεσιμότητα του πόρου για τις χρονικές στιγμές που ανήκουν στο F_g και είναι ανάμεσα στο χρόνο t και $t+d_j$, όπου d_j η διάρκεια της j.

Τέλος υπολογίζεται ο χρόνος λήξης της δραστηριότητας j και συμπληρώνεται το σύνολο Sec_g με την j.

2.6.2.1.3 Αποτέλεσμα

Αφού ολοκληρωθούν όλα τα στάδια του 2ου βήματος, υπολογίζεται ο χρόνος λήξης του έργου, που ισούται με τον χρόνο λήξης f_n της πλασματικής δραστηριότητας λήξης, ο οποίος

ορίζεται σαν το μέγιστο από τους χρόνους λήξης των προαπαιτούμενων δραστηριοτήτων της πλασματικής.

Βήμα 1: Αρχικοποίηση

$s_1 = f_1 = 0$

$Sec_1 = \{1\}$

Βήμα 2: FOR $g = 2$ TO $n - 1$ DO

Υπολογίζουμε $D_g, F_g, \bar{R}_k(t)$ για $k \in K$ και $t \in F_g$

Επιλογή $j \in D_g$

$ES_j = \max_{h \in Pred_j} \{f_h\}$

$s_j = \min\{t | t \geq ES_j, t \in F_g, r_{jk} \leq \bar{R}_k(\tau)$ για όλα τα $k \in K, \tau \in [t, t + d_j] \cap F_g\}$

$f_j = s_j + d_j$

$Sec_g = Sec_{g-1} \cup \{j\}$

Βήμα 3: $f_n = \max_{h \in Pred_n} \{f_h\}$

Αλγόριθμος 1: Ψευδοκώδικας για το σειριακό σύστημα παραγωγής χρονοπρογράμματος (Montoya-Torres et al., 2010).

Όλες οι μεταβλητές που χρησιμοποιεί ο αλγόριθμος συνοψίζονται στον Πίνακας 5.

Μεταβλητή	Έννοια
n	Συνολικός αριθμός δραστηριοτήτων του έργου. Δραστηριότητα λήξης του έργου.
j	Τυχαία δραστηριότητα του έργου.
s_j	Χρόνος έναρξης της δραστηριότητας ***j***.
f_j	Χρόνος λήξης της δραστηριότητας ***j***.
d_j	Διάρκεια της δραστηριότητας ***j***.
K	Σύνολο ανανεώσιμων πόρου του έργου.
k	Τυχαίος ανανεώσιμος πόρος του έργου.
r_{jk}	Απαίτηση της δραστηριότητας ***j*** στον πόρο ***k***.
R_k	Συνολική διαθεσιμότητα του πόρου ***k*** σε όλο το έργο.
g	Στάδιο του αλγορίθμου.
Sec_g	Σύνολο προγραμματισμένων δραστηριοτήτων στο στάδιο ***g***.
D_g	Σύνολο δραστηριοτήτων που μπορούν να επιλεγούν για προγραμματισμό στο στάδιο ***g***.
F_g	Σύνολο χρόνων λήξης των προγραμματισμένων δραστηριοτήτων στο στάδιο ***g***.
t	Τυχαία χρονική στιγμή από το σύνολο F_g.
$\bar{R}_k(t)$	Η διαθεσιμότητα του πόρου ***k*** την χρονική στιγμή ***t***.
ES_j	Νωρίτερος χρόνος έναρξης της δραστηριότητας ***j***.
$Pred_j$	Σύνολο προαπαιτούμενων δραστηριοτήτων της δραστηριότητας ***j***.

Πίνακας 5: Μεταβλητές του σειριακού συστήματος παραγωγής χρονοπρογράμματος.

2.6.2.1.4 Κανόνες προτεραιότητας

Οι κανόνες προτεραιότητας EBST1 και EBST2 ταξινομούν τις δραστηριότητες σε μη φθίνουσα σειρά των χρόνων εναρξής τους στο αρχικό χρονοπρόγραμμα. Διαφοροποιούνται όμως ως προς τον τρόπο με τον οποίο διευθετούν τις τυχόν ισοψηφίες που μπορεί να προκύψουν. Στον κανόνα EBST1 οι δραστηριότητες ταξινομούνται και σε φθίνουσα σειρά των βαρών τους, ενώ στον κανόνα EBST2 σε αύξουσα σειρά των αριθμών τους.

Ο κανόνας LW ταξινομεί σε φθίνουσα σειρά των βαρών, ενώ σε περίπτωση ισοψηφίας επιλέγεται ο μικρότερος αριθμός δραστηριότητας. Σε αντίθεση με τους προηγούμενους κανόνες που ασχολούνται με τους χρόνους ή τα βάρη των δραστηριοτήτων, ο κανόνας LAN ταξινομεί τις δραστηριότητες σε αύξουσα σειρά των αριθμών τους.

Τέλος, ο κανόνας RND δημιουργεί μια τυχαία σειρά για τις δραστηριότητες.

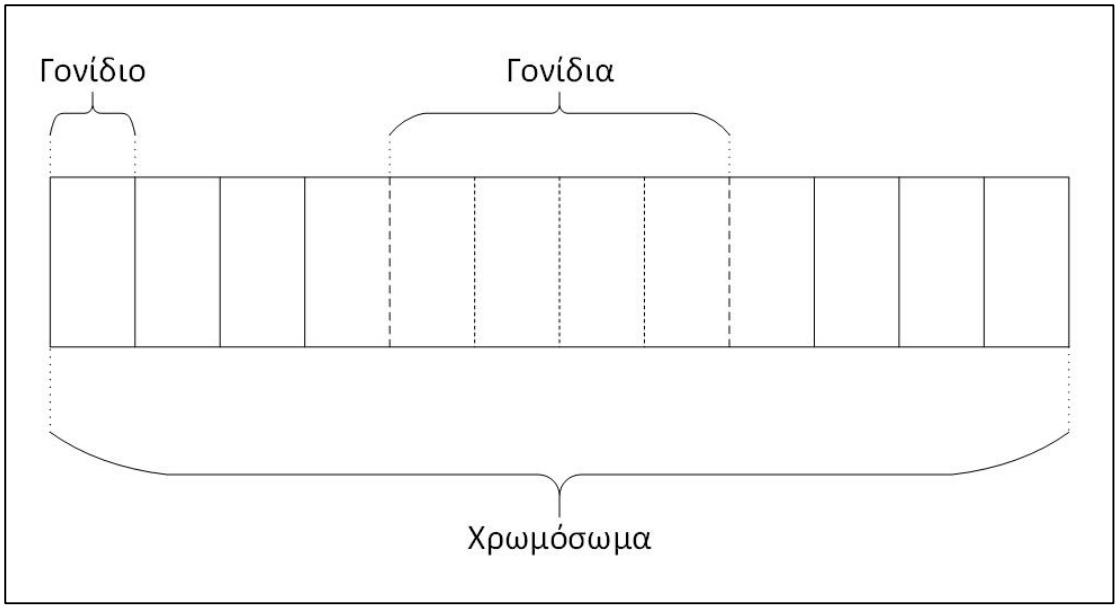
2.6.3 Μεταευρετικές μέθοδοι

Οι μεταευρετικές μέθοδοι έχουν σαν στόχο την επιλογή της καλύτερης λύσης από μια ομάδα λύσεων. Μερικές από αυτές τις μεθόδους είναι η μέθοδος της δειγματοληψίας (Sampling), όπου π.χ. από μία ομάδα κανόνων προτεραιότητας του σειριακού συστήματος παραγωγής χρονοπρογραμμάτος επιλέγεται ο καλύτερος (Van de Vonder et al., 2007), η μέθοδος ταμπού (Tabu Search), όπου βρίσκεται η βέλτιστη λύση μεταξύ χρονοπρογραμμάτων ελέγχοντας τα τμηματικά (Glover, 1989, Glover, 1990), και ο γενετικός αλγόριθμος, που παρουσιάζεται λεπτομερώς.

2.6.3.1 Γενετικός αλγόριθμος

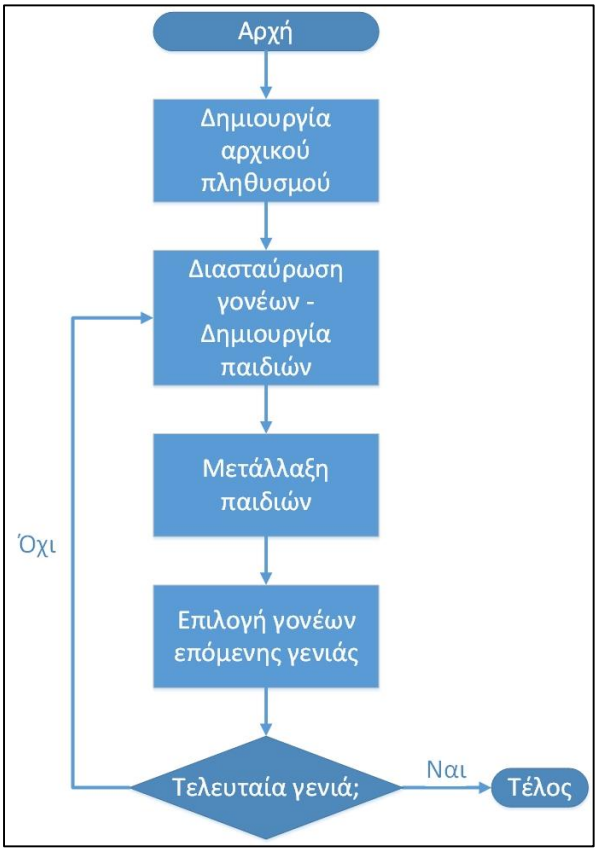
Η χρήση των γενετικών αλγόριθμων σε προβλήματα βελτιστοποίησης είναι ιδιαίτερα διαδεδομένη και αποτελεσματική, όπως προκύπτει από την πρόσφατη βιβλιογραφία (Montoya-Torres et al., 2010). Στο υπό μελέτη πρόβλημα, σκοπός του γενετικού αλγορίθμου είναι να εντοπίσει μία τυχαία σειρά δραστηριοτήτων, η οποία να προσφέρει μικρής διάρκειας εφικτό χρονοπρόγραμμα και μάλιστα οι σχετικοί υπολογισμοί να μην είναι ιδιαίτερα χρονοβόροι σε αντιπαράθεση με τις βέλτιστες λύσεις που προσφέρουν οι αναλυτικές μέθοδοι, αλλά απαιτούν εκθετικό χρόνο σε σχέση με το μέγεθος του προβλήματος. Επιπλέον οι γενετικοί αλγόριθμοι αλλά και γενικώς οι μεταευρετικοί μέθοδοι έχουν παρουσιάσει αποτελεσματικές επιλύσεις του προβλήματος του προγραμματισμού έργων με στοχαστικές διάρκειες(Bruni et al., 2011).

Κάθε γενετικός αλγόριθμος έχει τρία κύρια χαρακτηριστικά: το χρωμόσωμα (chromosome), τον πληθυσμό (population – POP) και τις γενιές (generations – GEN). Οι γενιές είναι το σύνολο των πληθυσμών του αλγορίθμου και ο πληθυσμός είναι το σύνολο των χρωμοσωμάτων κάθε γενιάς. Κάθε γενιά πρέπει να έχει τον ίδιο αριθμό χρωμοσωμάτων, συνεπώς ο πληθυσμός κάθε γενιάς έχει σταθερό πληθάριθμο. Το χρωμόσωμα είναι η βασική μεταβλητή του αλγορίθμου και περιέχει ένα σύνολο γονιδίων (genes) (Σχήμα 4).



Σχήμα 4: Χρωμόσωμα γενετικής μεθόδου.

Ο γενετικός αποτελείται από τέσσερα βασικά βήματα (Σχήμα 5): τον αρχικό πληθυσμό, τη διασταύρωση (crossover), τη μετάλλαξη (mutation) και την επιλογή (selection). Με εξαίρεση τον αρχικό πληθυσμό που παράγεται μόνο στη πρώτη γενιά, κάθε γενιά περιέχει τα τρία τελευταία βήματα.



Σχήμα 5: Διάγραμμα ροής γενετικής μεθόδου.

2.6.3.1.1 Αρχικός πληθυσμός

Ο αρχικός πληθυσμός αποτελεί τα χρωμοσώματα με τα οποία θα ξεκινήσει η πρώτη γενεά. Κάθε χρωμόσωμα στον αρχικό πληθυσμό μπορεί να δημιουργηθεί είτε τυχαία, δηλαδή η σειρά των δραστηριοτήτων παρήχθηκε με τυχαίο τρόπο, είτε με βάση κάποιο κανόνα, όπως τους κανόνες προτεραιότητας που είδαμε σε προηγούμενη ενότητα. Σε αυτή την εργασία θα δημιουργούνται όλοι τυχαία, μιας και η μέθοδος με την οποία δημιουργηθήκαν δεν επηρεάζει τον αλγόριθμο με κάποιο τρόπο.

Αυτός ο αρχικός πληθυσμός θα είναι οι γονείς (parents) της πρώτης γενεάς. Οι γονείς κάθε γενεάς είναι ο πληθυσμός με τον οποίο η γενιά ξεκινάει.

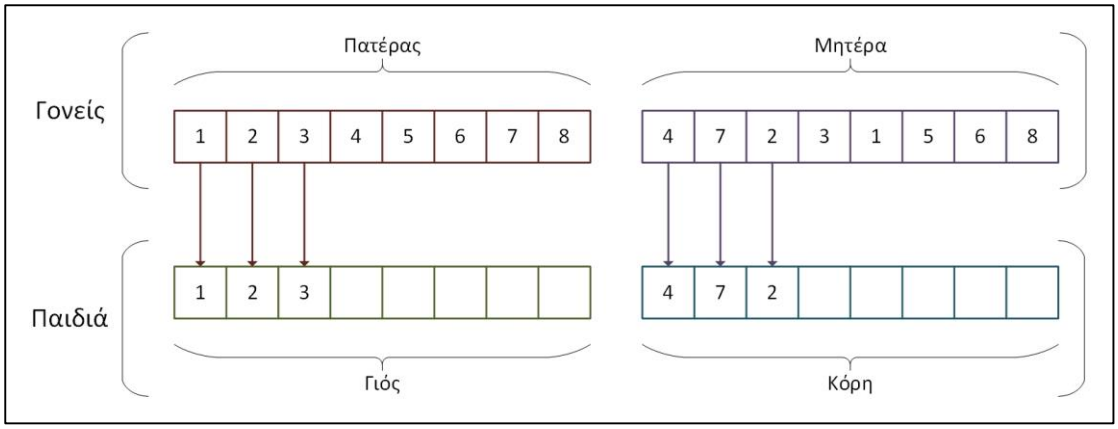
2.6.3.1.2 Διασταύρωση

Κατά τη διαστάυρωση, που είναι και ο βασικός τελεστής που εφαρμόζεται στον πληθυσμό των χρωμοσωμάτων, ο πληθυσμός των γονέων χρησιμοποιείται ως βάση για την παραγωγή των παιδιών (children) . Κάθε ζεύγος γονέων παράγει ένα ζεύγος παιδιών. Με βάση την υφιστάμενη βιβλιογραφια (Montoya-Torres et al., 2010) οι επικρατέστεροι τελεστές διασταύρωσης για το υπό επίλυση πρόβλημα είναι η διασταύρωση ενός σημείου και διασταύρωση δύο σημείων.

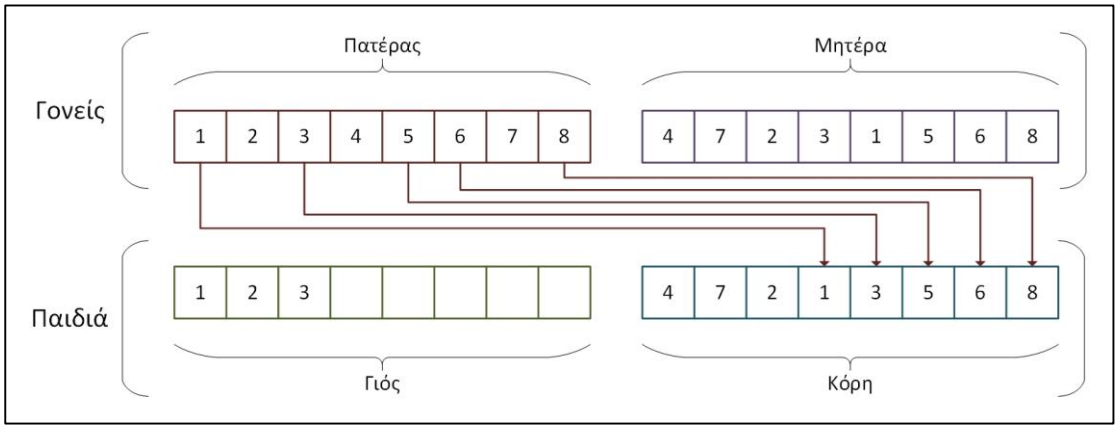
Έστω ότι Ορίζεται από το ζεύγος των γονέων, το ένα χρωμόσωμα ως πατέρα και το άλλο ως μητέρα. Με την ίδια λογική Ορίζεται στα παιδιά, το ένα γιό και το άλλο κόρη. Η μέθοδος του ενός σημείου ακολουθεί την εξής λογική:

- Τα γονίδια από την αρχή του χρωμοσώματος μέχρι ένα τυχαίο αριθμό ***q*** επιλέγονται για τον γιό από τον πατέρα σε αντιστοιχία ένα προς ένα. Δηλαδή τα γονίδια μεταφέρονται από τον πατέρα στην ίδια θέση στον γιό. Το ίδιο συμβαίνει και για την κόρη και την μητέρα.
- Στη συνέχεια για να καλυφθούν τα γονίδια του γιού από την θέση ***q+1*** μέχρι το τέλους του χρωμοσώματος, επιλέγονται από την μητέρα. Η επιλογή αυτή γίνεται κοιτώντας όλα τα γονίδια της μητέρας από την αρχή μέχρι το τέλος και επιλέγοντας αυτά που δεν βρίσκονται ήδη στον γιό. Η τοποθέτησή τους στο γιό γίνεται σειριακά, το πρώτο επιλέξιμο γονίδιο που βρίσκεται στην μητέρα τοποθετείται στο πρώτο κενό γονίδιο που βρίσκεται στο γιό. Αντίστοιχα συμπληρώνεται και το χρωμόσωμα της κόρης από τον πατέρα.

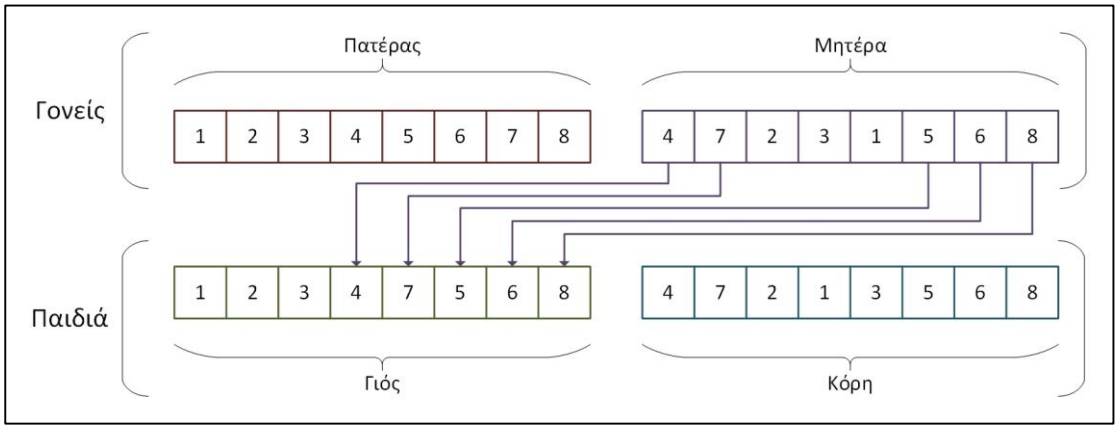
Ακολουθεί παράδειγμα για χρωμοσώματα 8 γονιδίων, δηλαδή για ένα έργο 8 δραστηριοτήτων, με ***q*** ίσο με 3 (Σχήμα 6, Σχήμα 7, Σχήμα 8).



Σχήμα 6: Στάδιο πρώτο διασταύρωσης ενός σημείου.



Σχήμα 7: Στάδιο δεύτερο διασταύρωσης ενός σημείου. Βήμα πρώτο.



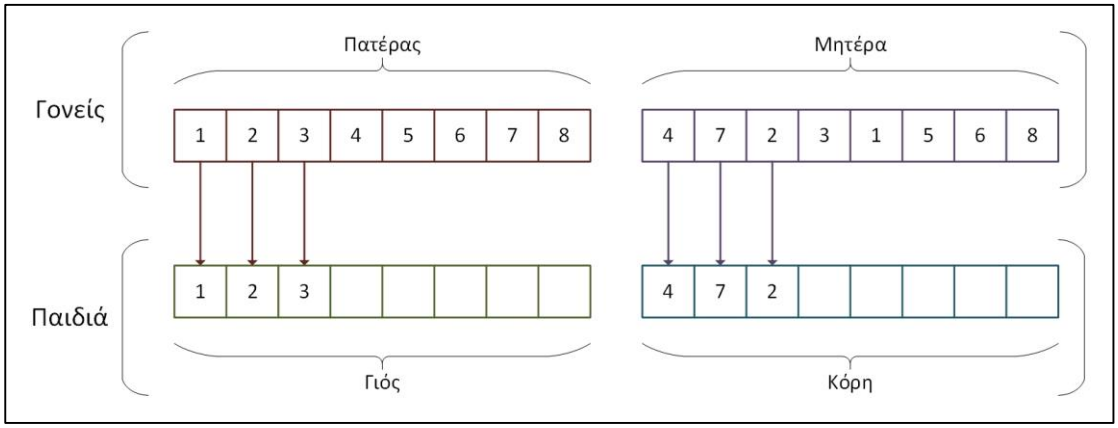
Σχήμα 8: Στάδιο δεύτερο διασταύρωσης ενός σημείου. Βήμα δεύτερο.

Η μέθοδος των δύο σημείων ακολουθεί την ίδια λογική αλλά σε τρία στάδια:

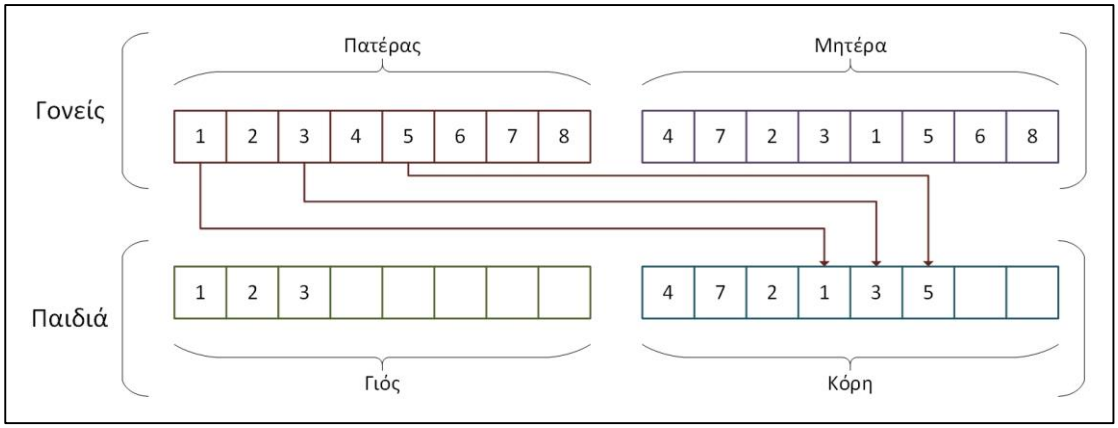
- Το πρώτο στάδιο είναι ίδιο με του ενός σημείου, αλλά αντί ενός αριθμού ***q*** έχουμε έναν αριθμό q_1.
- Το δεύτερο στάδιο επίσης είναι ίδιο με του ενός σημείου, αλλά συμφαίνει για τα γονίδια από q_1 μέχρι q_2.

- Το τρίτο στάδιο είναι αυτό που έχει την μεγαλύτερη διαφορά. Σε αυτό το στάδιο καλύπτονται τα γονίδια από τη θέση q_2 μέχρι το τέλους του χρωμοσώματος, και επιλέγονται από τον πατέρα για τον γιό και από την μητέρα για την κόρη.

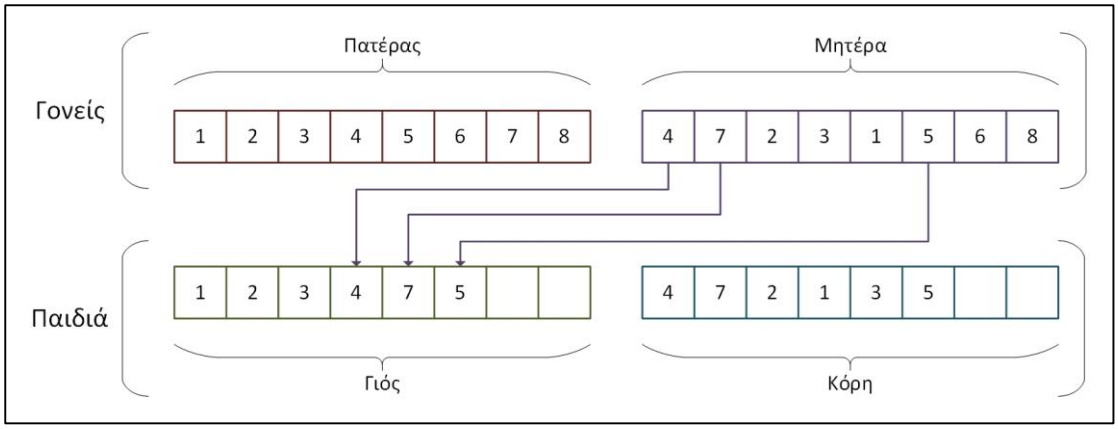
Ακολουθεί παράδειγμα για χρωμοσώματα 8 γονιδίων, δηλαδή για ένα έργο 8 δραστηριοτήτων, με q_1 ίσο με 3 και q_2 ίσο με 6 (Σχήμα 9, Σχήμα 10, Σχήμα 11, Σχήμα 12).



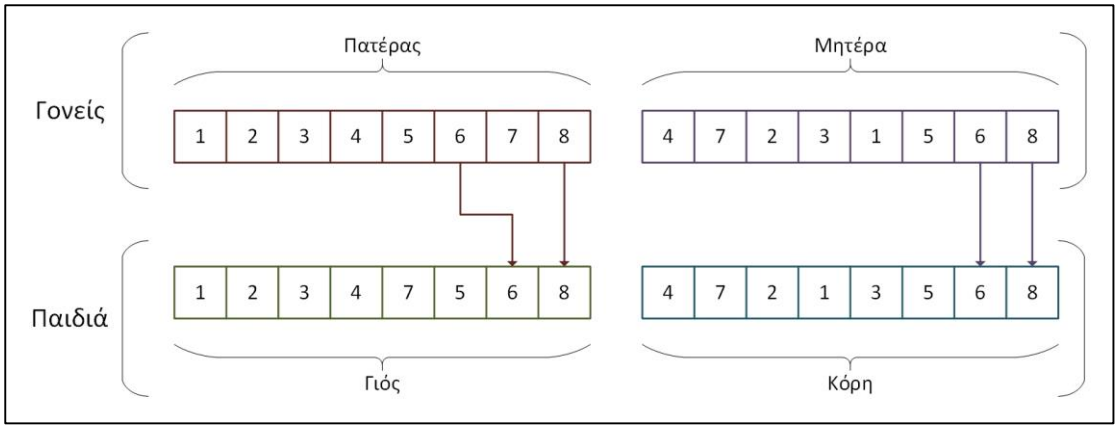
Σχήμα 9: Στάδιο πρώτο διασταύρωσης δύο σημείων.



Σχήμα 10: Στάδιο δεύτερο διασταύρωσης δύο σημείων. Βήμα πρώτο.



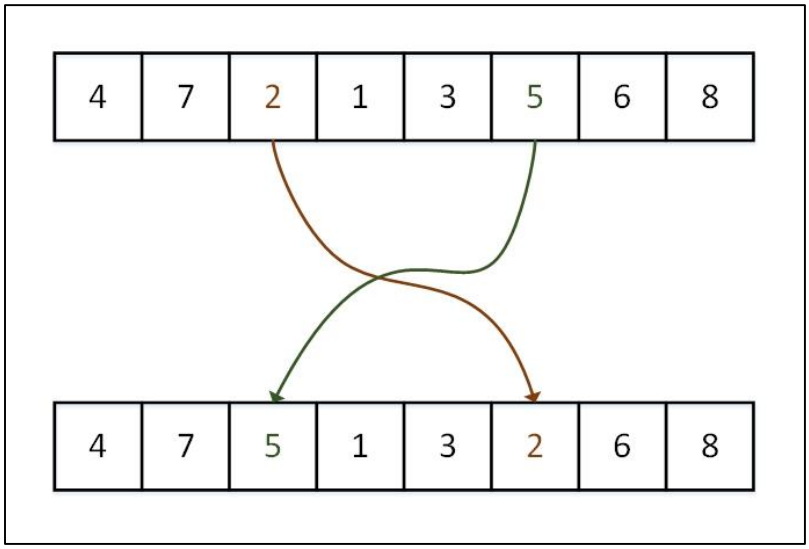
Σχήμα 11: Στάδιο δεύτερο διασταύρωσης δύο σημείων. Βήμα δεύτερο.



Σχήμα 12: Στάδιο τρίτο διασταύρωσης δύο σημείων.

2.6.3.1.3 Μετάλλαξη

Μετά την παραγωγή των παιδιών από τα χρωμοσώματα-γονείς της τρέχουσας γενιάς, εφαρμόζεται ο τελεστής μετάλλαξης. Όπως και στην περίπτωση του τελεστή διασταύρωσης, έτσι και για τον τελεστή μετάλλαξης, υπάρχει πληθώρα εναλλακτικών προτάσεων στη βιβλιογραφία (Montoya-Torres et al., 2010) αλλά ο πιό απλός και με άκρως ικανοποιητικά αποτελέσματα ορίζει ότι με πιθανότητα pmut πραγματοποιείται μετάλλαξη του χρωμοσώματος που ανήκει στα παιδιά, αν πραγματοποιείται μετάλλαξη τότε επιλέγονται τυχαία δυο γονίδια του συγκεκριμένου χρωμοσώματος και αλλάζουν θέση μεταξύ τους (Σχήμα 13).



Σχήμα 13: Παράδειγμα μετάλλαξης χρωμοσώματος.

2.6.3.1.4 Επιλογή

Οι γενιες διατηρούν σταθερό το πλήθος των στοιχείων τους, χρωμοσωμάτων. Στο βήμα της επιλογής, επιλέγεται με βάση ένα κριτήριο, την καταλληλότητα (fitness) του χρωμοσώματος, ποιά χρωμοσώματα από το σύνολο των γονέων και των παιδιών μιας γενιάς θα επιβιώσουν και θα περάσουν στην επόμενης γενιά. Σε κάθε διαδικασία επιλογής, επιβιώνουν τόσα χρωμοσώματα όσο απαιτούνται ώστε οι γονείς της επόμενης γενιας να έχουν ίσο πλήθος με τον αρχικό πληθυσμό.

3 Περιγραφή προβλήματος

3.1 Εισαγωγή

Κατά τον σχεδιασμό ενός έργου προσδιορίζονται αρχικά οι δραστηριότητες και οι διαθέσιμοι πόροι που θα υπάρχουν στο έργο. Προσεγγίζονται οι διάρκειες και οι ανάγκες σε πόρους για κάθε μια δραστηριότητα, καθώς και οι διαθεσιμότητες των πόρων, με βάση παλαιότερα δεδομένα από παρόμοια έργα, την εμπειρία του διαχειριστή του έργου και λοιπά στοιχεία συσχετιζόμενα με τις επικρατούσες συνθήκες που αναμένονται κατά την εκτέλεση του έργου.

Με βάση τα δεδομένα που προκύπτουν ως αποτέλεσμα της φάσης του σχεδιασμού θα παραχθεί το αρχικό χρονοπρόγραμμα του έργου κατά την φάση του προγραμματισμού. Αυτό το αρχικό χρονοπρόγραμμα θα ήταν επιθυμητό να ακολουθηθεί πιστά καθ'όλη τη διάρκεια του έργου, αλλά αυτό συχνά δεν είναι εφικτό στην πράξη λόγω διαφοροποιήσεων που προκύπτουν σε σχέση τον αρχικό προγραμματισμό (Ke and Liu, 2005). Οι διαφοροποιήσεις μπορεί να επηρεάζουν το έργο με δύο τρόπους, είτε αυξάνοντας το τελικό κόστος του είτε αυξάνοντας το τελικό χρόνο λήξης του έργου είτε και τα δυο. Στη παρούσα εργασία εξετάζεται μόνο ο χρόνος λήξης του έργου και θεωρείται το τελικό κόστος είτε αμετάβλητο είτε ανάλογο του χρόνου λήξης.

3.2 Αβεβαιότητα και διαφοροποιήσεις

Όλα τα στοιχεία που υπολογίστηκαν κατά την φάση του σχεδιασμού, όσο καλοί και αναλυτικοί και να ήταν οι υπολογισμοί, στην πράξη προκύπτει ότι ποτέ δεν αντικατοπτρίζουν την πραγματικότητα απόλυτα. Η αβεβαιότητα που εμπεριέχεται σε όλα αυτά τα στοιχεία είναι αποτέλεσμα της φυσικής τους υπόστασης. Όπως και στην φύση δεν είναι δυνατόν να προβλεφθεί με 100% βεβαιότητα τι θα συμβεί, ειδικά σε ένα μεγάλο χρονικό ορίζοντα, έτσι και στα έργα οι διάρκειες των δραστηριοτήτων και οι διαθεσιμότητες των πόρων είναι αβέβαιοι (Sadeh et al., 1993).

Για παράδειγμα, έστω ένα έργο που αποτελεί ένα πρόβλημα τύπου RCPSP, δηλαδή έχει περιορισμένης διαθεσιμότητας ανανεώσιμους πόρους και η διαδοχικότητα μεταξύ των δραστηριοτήτων είναι μόνο τύπου τέλος – αρχή. Έστω δύο δραστηριότητες όπου η πρώτη χρειάζεται 4 μέρες να ολοκληρωθεί και χρησιμοποιεί ένα από τα δύο μηχανήματα που έχει διαθέσιμα το έργο και η δεύτερη χρειάζεται 2 μέρες και χρησιμοποιεί παρομοίως ένα από τα δύο μηχανήματα. Αυτά τα δεδομένα προέκυψαν μετά από εκτενή σχεδιασμό και χρονοπρογραμματίστηκε το έργο με βάση αυτά. Επειδή οι δύο δραστηριότητες δεν συνδέονται μεταξύ τους το χρονοπρόγραμμα προέκυψε με την δεύτερη δραστηριότητα να εκτελείται πριν την πρώτη, διότι το δεύτερο μηχάνημα το χρειάζεται μία τρίτη δραστηριότητα.

Αν δεν υπάρξει καμία διαφοροποίηση το χρονοπρόγραμμα θα εκτελεστεί όπως είναι. Κατά την εκτέλεση του έργου όμως προέκυψε ένα πρόβλημα. Ο εργάτης που κανονικά θα εκτελούσε την δεύτερη δραστηριότητα αρρώστησε και ο εργάτης που τελικά θα την εκτελέσει δεν είναι εξοικειωμένος με το μηχάνημα και θα χρειαστεί 4 μέρες αντί για 2. Αφού υπάρχει μόνο ένα μηχάνημα ελεύθερο θα χρειαστεί να μετακινηθεί και η πρώτη δραστηριότητα 2 μέρες μετά, με αποτέλεσμα να μετακινηθούν και όλες οι δραστηριότητες

που βασίζονται σε αυτήν τουλάχιστον 2 μέρες μετά. Αυτό φυσικά σε ένα πολύ μεγάλο έργο μπορεί να δημιουργήσει τεράστιο πρόβλημα.

Αυτή ήταν φυσικά μόνο μια από τις διαφοροποιήσεις που θα μπορούσαν να συμβούν και μια που μπορεί να λυθεί και κατά την διάρκεια του σχεδιασμού, αν εκπαιδεύαμε και τον δεύτερο εργάτη στο μηχάνημα. Βέβαια θα μπορούσε να είχε πάθει βλάβη το ένα από τα δύο μηχανήματα με αποτέλεσμα να μην μπορούν να γίνουν οι δύο πρώτες διαστηριότητες αν εκτελείται η τρίτη και αντίστροφα. Παρομοίως θα μπορούσε και αυτό το πρόβλημα να λυνόταν με συχνή συντήρηση των μηχανών ή με μία τρίτη μηχανή.

Υπάρχουν όμως και διαφοροποιήσεις που δεν μπορούν να προβλεφθούν με καμία δυνατότητα. Συνήθως τα έργα είναι μεγάλα σε διάρκεια και ο σχεδιασμός τους γίνεται αρκετό καιρό πριν την εκτελεσή τους. Συνεπώς οι καιρικές συνθήκες που θα ισχύουν κατά την εκτέλεση του έργου μπορούν μόνο να εκτιμηθούν και φυσικά δεν θα είναι 100% ακριβής. Μπορεί κατά τον σχεδιασμό να υπολογίσεις μέσα καθυστέρηση λόγω βροχής, επειδή η δραστηριότητα θα εκτελεστεί μέσα στο χειμώνα, αλλά μπορεί να χιονίσει, κάτι το οποίο είναι σπάνιο αλλά όχι απίθανο για την περιοχή που εκτελείται το έργο. Βεβαίως δεν σχεδιάστηκε η δραστηριότητα με συνυπολογισμό το χιόνι, διότι αφού είναι σπάνια περίπτωση, ο συνυπολογισμό του θα επέκτεινε τη διάρκεια του έργου χωρίς σημαντικό λόγο (Herroelen and Leus, 2005).

Η αβεβαιότητα συνεπώς υπάρχει μόνο λόγω στοιχείων των οποίων ο υπολογισμός είναι αδύνατος, αλλά και λόγω στοιχείων που αγνοήσαμε κατά τον σχεδιασμό επειδή αποτελούν σπάνιες περιπτώσεις και υποθέτουμε ότι δεν θα συμβούν.

3.3 Πολυπλοκότητα

Η αβεβαιότητα, όπως παρατηρήθηκε στην προηγούμενη ενότητα, μπορεί να προκαλέσει προβλήματα σε ένα απλό πρόβλημα, όπως στο RCPSP. Όμως υπάρχουν και πιο περίπλοκα προβλήματα χρονοπρογραμματισμού έργων, με πολλαλούς τύπους πόρων, πολλαπλούς τρόπους διαδοχικότητας δραστηριοτήτων καθώς και πολλαπλούς τρόπους εκτέλεσης δραστηριοτήτων. Είναι συνεπώς λογικό όσο πιο περίπλοκο το πρόβλημα τόσο πιο περίπλοκες και οι επιπτώσεις των διαφοροποιήσεων σε αυτό.

3.4 Μαθηματική μοντελοποίηση

3.4.1 Ορισμοί

Πριν την κατανόηση του μαθηματικού μοντέλου του προβλήματος, είναι απαραίτητη η κατανόηση των μεταβλητών που θα χρησιμοποιηθούν σε αυτό.

- Ορίζεται με το σύμβολο ***v*** τον αριθμό των δραστηριοτήτων ενός έργου. Όταν το ***v*** χρησιμοποιείται σαν δείκτης υποδηλώνει την πλασματική δραστηριότητα λήξης του έργου.
- Ορίζεται με το σύμβολο ***A*** το σύνολο των δραστηριοτήτων ενός έργου. Μέσα σε αυτό το σύνολο υπάρχουν και οι πλασματικές δραστηριότητες έναρξης και λήξης του έργου. Το σύνολο έχει μέγεθος ***v***.
- Ορίζεται με τα σύμβολα ***i*** και ***j*** μία τυχαία δραστηριότητα από το σύνολο ***A***. Τα ***i*** και ***j*** μπορούν να πάρουν τιμές από ***1*** έως ***v.***

- Ορίζεται με το σύμβολο d_i η διάρκεια της δραστηριότητας ***i***. Η διάρκεια μετριέται σε χρονικές μονάδες που έχουν οριστεί από το έργο. Συνήθως σε ημέρες εργασίας.
- Ορίζεται με το σύμβολο s_i ο χρόνος έναρξης της δραστηριότητας ***i***. Είναι η χρονική στιγμή στην οποία θα ξεκινήσει την εκτέλεσή της η δραστηριότητα.
- Ορίζεται με το σύμβολο f_i ο χρόνος λήξης της δραστηριότητας ***i***. Είναι η χρονική στιγμή στην οποία θα τερματίσει την εκτέλεσή της η δραστηριότητα.
- Ορίζεται με το σύμβολο w_i το βάρος της δραστηριότητας ***i***. Είναι ο συντελεστής με τον οποίο πολλαπλασιάζονται συγκεκριμένα στοιχεία μιας δραστηριότητας όταν συγκρίνονται με άλλες δραστηριότητες. Δείχνει την σπουδαιότητα (βάρος) που έχει η δραστηριότητα στο έργο.
- Ορίζεται με τη λέξη ***lag*** η καθυστέρη ή προπορεία μεταξύ δυο συσχτιζόμενων δραστηριοτήτων.
- Ορίζεται με το σύμβολο ***κ*** ο αριθμός των ανανεώσιμων πόρων ενός έργου.
- Ορίζεται με το σύμβολο ***K*** το σύνολο των ανανεώσιμων πόρων ενός έργου. Το σύνολο έχει μέγεθος ***κ***.
- Ορίζεται με το σύμβολο ***k*** ένας τυχαίος πόρος από το σύνολο ***K***. Το ***k*** μπορεί να πάρει τιμές από ***1*** έως ***κ***.
- Ορίζεται με το σύμβολο R_k η διαθεσιμότητα του πόρου ***k*** σε ολόκληρο το έργο.
- Ορίζεται με το σύμβολο r_{ik} η απαίτηση στον πόρο ***k*** της δραστηριότητας ***i***.
- Ορίζεται με το σύμβολο ***λ*** ο αριθμός των μη-ανανεώσιμων πόρων ενός έργου.
- Ορίζεται με το σύμβολο ***Λ*** το σύνολο των μη-ανανεώσιμων πόρων ενός έργου. Το σύνολο έχει μέγεθος ***λ***.
- Ορίζεται με το σύμβολο ***l*** ένας τυχαίος πόρος από το σύνολο ***Λ***. Το ***l*** μπορεί να πάρει τιμές από το ***1*** έως ***λ***.
- Ορίζεται με το σύμβολο N_l η διαθεσιμότητα του πόρου ***l*** σε ολόκληρο το έργο.
- Ορίζεται με το σύμβολο n_{il} η απαίτηση στον πόρο ***l*** της δραστηριότητας ***i***.
- Ορίζεται με το σύμβολο ***t*** μία τυχαία χρονική στιγμή κατά την εκτέλεση του έργου.
- Ορίζεται με το σύμβολο $\bar{R}_k(t)$ η διαθεσιμότητα του πόρου ***k*** την χρονική στιγμή ***t***.
- Ορίζεται με το σύμβολο ***S*** το χρονοπρόγραμμα του έργου.
- Ορίζεται με το σύμβολο ***0*** ο εκθέτης που υποδηλώνει ότι η μεταβλητή ανήκει στο αρχικό χρονοπρόγραμμα. Αρχικό χρονοπρόγραμμα είναι το χρονοπρόγραμμα που υπολογίστηκε κατά τη φάση του προγραμματισμού πριν την εκτέλεση του έργου. Συνεπώς έχουμε:
    - S^0 το αρχικό χρονοπρόγραμμα του έργου,
    - d_i^0 τη διάρκεια της δραστηριότητας ***i*** στο αρχικό χρονοπρόγραμμα,
    - s_i^0 το χρόνο έναρξης της δραστηριότητας ***i*** στο αρχικό χρονοπρόγραμμα,
    - f_i^0 το χρόνο λήξης της δραστηριότητας ***i*** στο αρχικό χρονοπρόγραμμα,
    - R_k^0 την διαθεσιμότητα του πόρου ***k*** στο αρχικό χρονοπρόγραμμα,
    - r_{ik}^0 την απαίτηση στον πόρο ***k*** της δραστηριότητας ***i*** στο αρχικό χρονοπρόγραμμα,
- Ορίζεται με το σύμβολο ***T*** ο εκθέτης που υποδηλώνει ότι η μεταβλητή ανήκει στο πραγματικό χρονοπρόγραμμα την χρονική στιγμή ***T***, όπου συμβαίνει μια διαφοροποίηση. Το χρονοπρόγραμμα αυτό λογικά δεν θα είναι εφικτό λόγο της διαφοροποίησης. Συνεπώς έχουμε:

- S^T το πραγματικό χρονοπρόγραμμα του έργου την χρονική στιγμή ***T***,
- d_i^T η διάρκεια της δραστηριότητας ***i*** στο πραγματικό χρονοπρόγραμμα την χρονική στιγμή ***T***,
- s_i^T ο χρόνος έναρξης της δραστηριότητας ***i*** στο πραγματικό χρονοπρόγραμμα την χρονική στιγμή ***T***,
- f_i^T ο χρόνος λήξης της δραστηριότητας ***i*** στο πραγματικό χρονοπρόγραμμα την χρονική στιγμή ***T***,
- R_k^T η διαθεσιμότητα του πόρου ***k*** στο πραγματικό χρονοπρόγραμμα την χρονική στιγμή ***T***,
- r_{ik}^T η απαίτηση στον πόρο ***k*** της δραστηριότητας ***i*** στο πραγματικό χρονοπρόγραμμα την χρονική στιγμή ***T***,

- Ορίζεται με τα σύμβολα **'** και **"** οι εκθέτες που υποδηλώνουν τις δύο νέες δραστηριότητες που προκύπτουν μετά τον χωρισμό μιας δραστηριότητας που εκτελείται την χρονική στιγμή ***T*** αλλά δεν έχει ολοκληρωθεί ακόμη. Ο εκθέτης **'** υποδηλώνει το τμήμα που έχει ολοκληρωθεί ενώ ο **"** το τμήμα που παραμένει προς εκτέλεση.
- Ορίζεται με το σύμβολο A^C το σύνολο των δραστηριοτήτων που έχουν ήδη ολοκληρωθεί την χρονική στιγμή ***T***.
- Ορίζεται με το σύμβολο A^E το σύνολο των δραστηριοτήτων που εκτελούνται την χρονική στιγμή ***T***.

3.4.2 Μαθηματικές Σχέσεις

Ανάμεσα στις μεταβλητές που ορίστηκαν στην προηγούμενη ενότητα υπάρχουν κάποιες μαθηματικές σχέσεις, οι οποίες είτε υποδηλώνουν συσχέτιση μεταξύ των μεταβλητών είτε υποδηλώνουν συσχέτιση μεταξύ των δραστηριοτήτων στις οποίες ανήκουν.

Αρχικά παρουσιάζεται η συσχέτιση μεταξύ μεταβλητών ανεξαρτήτως σε ποιό χρονοπρόγραμμα ανήκουν:

- $s_i + d_i = f_i, \forall i \in A$
- $\bar{R}_k(t) = R_k - \sum_{i|s_i \le t \atop f_i > t} r_{ik}$, $\forall k \in K$

Στη συνέχεια παρουσιάζονται οι σχέσεις που υποδηλώνουν τον τρόπο συσχέτισης δύο δραστηριοτήτων:

- Αρχή – αρχή (start – start): $s_i \pm lag = s_j$
- Αρχή – τέλος (start – finish): $s_i \pm lag = f_j$
- Τέλος – αρχή (finish – start): $f_i \pm lag = s_j$
- Τέλος – τέλος (finish – finish): $f_i \pm lag = f_j$

Στις τέσσερεις παραπάνω εξισώσεις, θεωρείται η δραστηριότητα ***i*** προαπαιτούμενη (predecessor) δραστηριότητα της δραστηριότητας ***j***, και η δραστηριότητα ***j*** διάδοχος (successor) της δραστηριότητας ***i***. Φυσικά αυτή η θεώρηση είναι αυθαίρετη και γίνεται κυρίως για λόγους μαθηματικής ευχέριας αφού καθαρή προαπαιτούμενη και καθαρό διάδοχο έχουμε μόνο στην περίπτωση της διαδοχικότητας τέλους – αρχής.

Τέλος, για τις δραστηριότητες που έχουν χωριστεί τη χρονική στιγμή $\boldsymbol{T}$ ισχύουν:

- $\boldsymbol{s_i = s'_i}$
- $\boldsymbol{d_i = d'_i + d''_i}$
- $\boldsymbol{r_{ik} = r'_{ik} = r''_{ik}}$, $\boldsymbol{\forall k \in K}$
- $\boldsymbol{s''_i \geq s'_i + d'_i}$

3.4.3 Μοντέλο

Επομένως η προκύπτουσα μαθηματική μοντελοποίηση του προς επίλυση προβλήματος είναι:

$\boldsymbol{min\ \Delta(S^0, S^T) = \sum_i w_i E\left|s_i^T - s_i^0\right|}$, $\boldsymbol{\forall i \in A,}$
με:
$\boldsymbol{s_i^T = s_i^0}$, $\boldsymbol{\forall i \in V^C}$
$\boldsymbol{s_i^{T'} = s_i^T}$, $\boldsymbol{\forall i \in V^E}$
$\boldsymbol{s_i^{T''} \geq s_i^{T'} + d_i^{T'}}$, $\boldsymbol{\forall i \in V^E}$
$\boldsymbol{R_k \geq \bar{R}_k(t)}$, $\boldsymbol{\forall t}$, $\boldsymbol{\forall k \in K}$
$\boldsymbol{N_l \geq \sum_{i \in A} n_{il}}$, $\boldsymbol{\forall l \in L}$

4 Προτεινόμενη επίλυση προβλήματος

4.1 Εισαγωγή

Στο υπό μελέτη πρόβλημα έχουμε ένα έργο για το οποίο έχει παραχθεί ένα αρχικό χρονοπρόγραμμα και στη συνέχεια κατά τη διάρκεια της εκτέλεσης του λόγω αλλαγών τόσο σε διάρκειες δραστηριοτήτων όσο και σε διαθεσιμότητες πόρων, ο διαχειριστής έργου καλείται να επαναπρογραμματίσει το εναπομένον έργο με τέτοιο τρόπο ώστε και η διάρκεια του έργου να συνεχίσει να είναι το δυνατόν μικρή και ταυτόχρονα οι αλλαγές σε σχέση με το αρχικό χρονοπρόγραμμα να είναι όσο το δυνατό μικρότερες, προς αποφυγή προβλημάτων σε σχέση με συνεννοήσεις με εξωτερικά συνεργεία, παραλαβές υλικών και γενικότερα αποφυγή σημαντικών αλλαγών σε σχέση με τα προβλευθέντα.Η πολυπλοκότητά του συγκεκριμένου προβλήματος προκαλείται τόσο από τους πολλαπλούς τρόπους συσχέτισης των δραστηριοτήτων, από τους διάφορους τύπους πόρων που υπάρχουν αλλά και από το γεγονός ότι το προυπάρχον χρονοπρόγραμμα αποτελεί περιοριστικό παράγοντα κατά την διαδικασία του επαναπρογραμματισμού

Η μετατροπή του συγκεκριμένου προβλήματος του επαναπρογραμματισμού εν εξελίξει έργου σε μια πιο απλή μορφή αν και θα βελτίωνε σημαντικά την ταχύτητα και την ποιότητα των υπολογιζόμενων λύσεων, είναι μια ιδιαίτερα δύσκολη διαδικασία και χρειάζεται να γίνουν αρκετές παραδοχές για να έχει τελικώς το έργο μορφή που να ανταποκρίνεται στα προβλήματα που οι διαχειριστές έργων καλούνται στην πράξη να επιλύσουν.

4.2 Μετατροπή

Ένα έργο μπορεί να έχει πόρους ανανεώσιμους, μη ανανεώσιμους και διπλά περιορισμένους. Οι τελευταίοι δεν συναντώνται συχνά και συνήθως μπορούν να μετατραπούν με χρήση ενός ανανεώσιμου και ενός μη ανανεώσιμου πόρου και έτσι δεν απαιτείται να ληφθούν υπόψη στην επίλυση ως διακριτή κατηγορία πόρων.

Οι μη ανανεώσιμοι πόροι αποτελούν συνήθως υλικά και πρώτες ύλες που κατά την διάρκεια της διαδικασίας καταναλώνονται ή ακόμη μπορεί και να είναι συνεργεία που προσλαμβάνονται για να εκτελέσουν την εργασία. Στην δεύτερη περίπτωση τυχών διαφοροποιήσεις που να επηρεάζουν το συνεργείο δεν επηρεάζουν το έργο, καθώς η επίλυση του προβλήματος είναι υπευθυνότητα του συνεργείου. Επίσης η πρώτη περίπτωση μη ανανεώσιμων πόρων συνήθως υπολογίζονται έτσι ώστε να έχουμε μεγάλο περιθώριο για τυχόν διαφοροποιήσεις, διότι αν περισσέψουν τέτοιου είδους πόροι μπορούν είτε να κρατηθούν για άλλα έργα είτε να μεταπουληθούν. Συνεπώς, μπορούμε να υποθέσουμε ότι οι μη ανανεώσιμοι πόροι δεν επηρεάζουν το έργο με το ίδιο βαθμό που τον επηρεάζουν οι ανανεώσιμοι και μπορούν να παραληφθούν κατά τον χρονοπρογραμματισμό του. Αυτή είναι η πρώτη παραδοχή που λαμβάνουμε υπ’όψιν στην επίλυση του προβλήματος. Ουσιαστικά οι μη ανανεώσιμοι πόροι λόγω του ότι έχουν σταθερό συνολικό διαθέσιμο για το σύνολο του έργου και θεωρούνται διαθέσιμοι από την αρχή αυτού και για όλη τη διάρκεια της εκτέλεσής του, δεν αναμένεται να μπορούν να επιρεάσουν στις περισσότερες των περιπτώσεων τη διαδικασία επαπαπρογραμματισμού του έργου ούτε και να προκαλέσουν διαταραχές στην ομαλή εκτέλεση του.

4.3 Μέθοδος επίλυσης

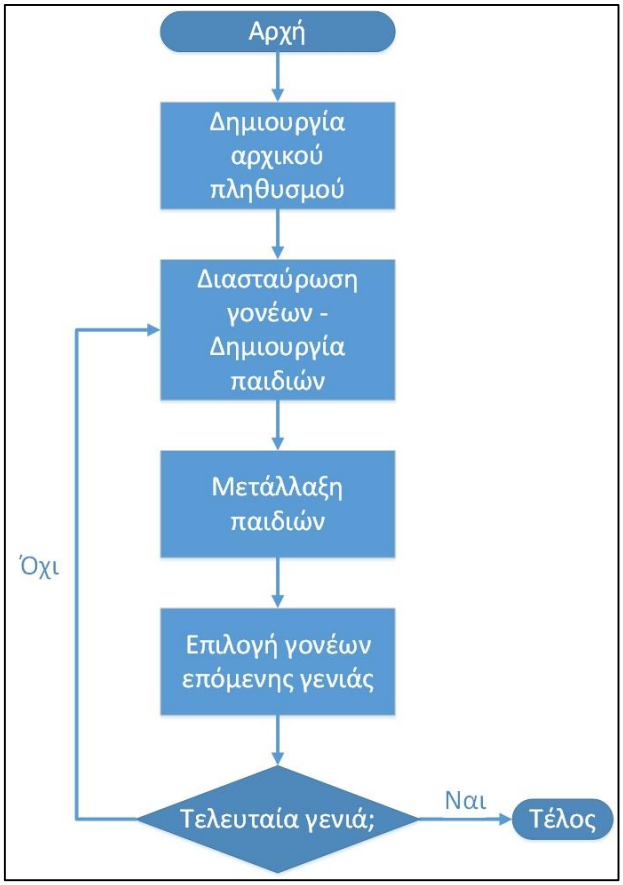
Έστω ένα έργο το οποίο είναι υπό εκτέλεση με βάση ένα αρχικό του χρονοπρόγραμμα. Σε κάποια χρονική στιγμή το χρονοπρόγραμμα υπόκεινται σε διάφορες διαφοροποιήσεις στις απαιτήσεις των πόρων και στις διάρκειες των δραστηριοτήτων του έργου, που δεν έχουν ολοκληρωθεί ακόμη. Εξαιτίας των διαφοροποιήσεων δεν είναι πια εφικτό το αρχικό χρονοπρόγραμμα και απαιτείται να βρεθεί ένα καινούργιο χρονοπρόγραμμα για την συνέχιση του έργου. Η μέθοδος με την οποία θα προκύψει το νέο χρονοπρόγραμμα είναι ένας συνδυασμός του γενετικού αλγορίθμου και του σειριακού συστήματος παραγωγής χρονοπρογράμματος και η διαδικασία της μεθόδου παρουσιάζεται στο Σχήμα 14.

Σχήμα 14: Διάγραμμα επίλυσης.

Αρχικά χωρίζουμε τις δραστηριότητες του έργου σε δύο ομάδες, τις ολοκληρωμένες δραστηριότητες και τις μη ολοκληρωμένες. Στην δεύτερη ομάδα περιλαμβάνονται και οι ημιολοκληρωμένες δραστηριότητες, δηλαδή οι δραστηριότητες που είναι υπό εκτέλεση τη χρονική στιγμή του συμβαίνουν οι διαφοροποιήσεις. Μη την δεύτερη ομάδα δραστηριοτήτων δημιουργείται έναν νέο πλασματικό έργο το οποίο έχει σαν αρχή του την

χρονική στιγμή που συνέβησαν οι διαφοροποιήσεις. Στο νέο έργο οι ημιολοκληρωμένες δραστηριότητες κόβονται έτσι ώστε να έχουν διάρκεια μόνο το τμήμα που δεν έχει ολοκληρωθεί και επιλέγεται από τον διαχειριστή του έργου αν θα έχουν καθυστέρηση στην εκτέλεσή τους.

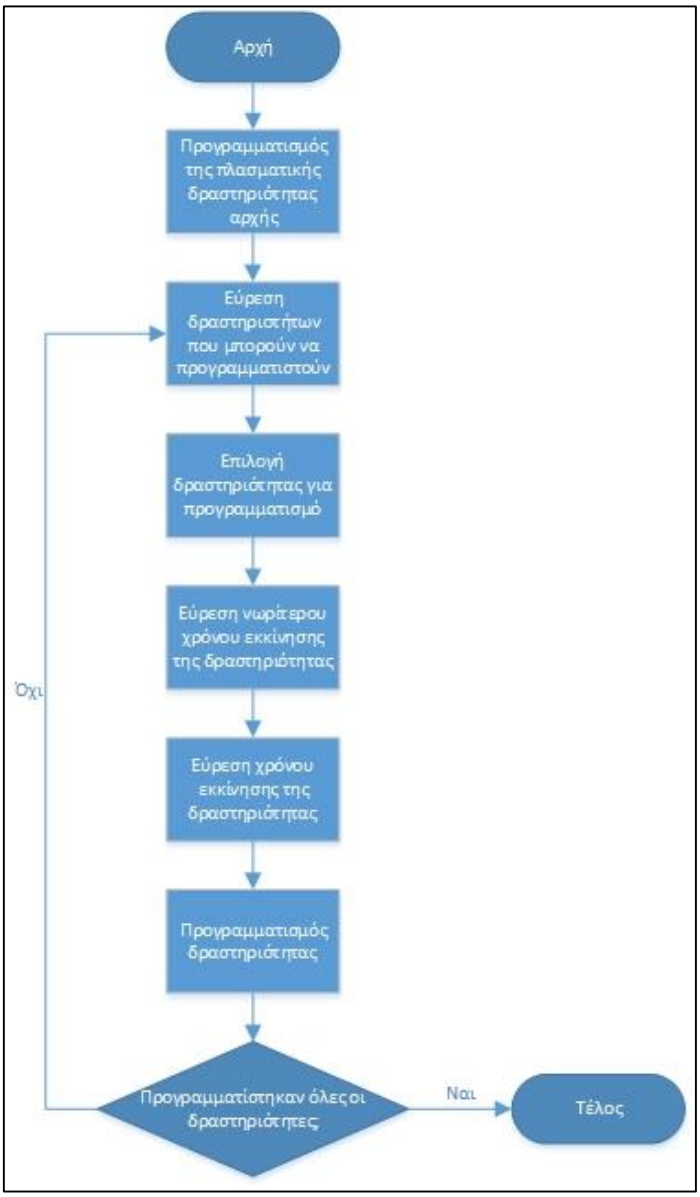
Κατέχοντας το νέο έργο ο διαχειριστής του μπορεί είτε να λειτουργήσει τον γενετικό αλγόριθμο με τις παραμέτρους που έχει αποθηκευμένες το πρόγραμμα είτε να ορίσει τις παραμέτρους ο ίδιος, επιλέγοντας τον πληθυσμό των χρωμοσωμάτων κάθε γενιάς, τον αριθμών των γενιών, την πιθανότητα μετάλλαξης κάθε χρωμοσώματος και την μέθοδο διασταύρωσης, είτε του ενός σημείου είτε των δύο σημείων. Με αυτόν τον τρόπο δεν είναι απαραίτητο να γνωρίζει ο διαχειριστής του έργου τον τρόπο λειτουργίας του γενετικού αλγορίθμου για να λειτουργήσει την αναδραστική μέθοδο, αφού υπάρχουν ήδη αποθηκευμένες παράμετροι με τις οποίες μπορούν να βρεθούν ικανοποιητικές λύσεις. Με αυτές τις παραμέτρους ξεκινάει ο γενετικός αλγόριθμος (Σχήμα 15).



Σχήμα 15: Διάγραμμα ροής γενετικού αλγορίθμου.

Πρώτα, παράγει έναν τυχαίο αρχικό πληθυσμός χρωμοσωμάτων για τους γονείς της πρώτης γενιάς. Τα γονίδια των χρωμοσωμάτων αντιστοιχούν σε δραστηριότητες του νέου έργου με εξαίρεση των πλασματικών δραστηριοτήτων αρχής και τέλους, και τα χρωμοσώματα αντιπροσωπεύουν την σειρά προτεραιότητας των δραστηριοτήτων. Ο αρχικός πληθυσμός με την μέθοδο της διασταύρωσης που επέλεξε ο διαχειριστής παράγει τα παιδιά της γενιάς. Τα παιδιά στην συνέχεια υπόκεινται σε μετάλλαξη. Κάθε παιδί έχει την πιθανότητα που όρισε ο διαχειριστής για να μεταλλαχθεί σε ένα τυχαίο γονίδιό του. Για να γίνει η επιλογή

των γονέων της επόμενης γενιάς χρησιμοποιείται το σειριακό σύστημα παραγωγής χρονοπρογράμματος (Σχήμα 16) για να παραχθεί το χρονοπρόγραμμα που αντιστοιχεί στο κάθε χρωμόσωμα.



Σχήμα 16: Διάγραμμα ροής σειριακού συστήματος παραγωγής χρονοπρογράμματος.

Το σειριακό σύστημα παραγωγής χρονοπρογράμματος ξεκινάει προγραμματίζοντας την πλασματική δραστηριότητα αρχής του έργου την χρονική στιγμή που βρίσκεται το έργο. Στην συνέχεια ψάχνει να βρει τις δραστηριότητες που μπορούν να προγραμματιστούν. Για να μπορεί να προγραμματιστεί μια δραστηριότητα χρειάζεται να έχουν ήδη προγραμματιστεί όλες οι προαπαιτούμενές της δραστηριότητες. Επιλέγεται από τις δραστηριότητες που μπορούν να προγραμματιστούν η πρώτη που συναντάται στα γονίδια του χρωμοσώματος. Βρίσκεται για αυτήν την δραστηριότητα ο νωρίτερος χρόνος εκκίνησης της, ο οποίος είναι ίσος με τον αργύτερο χρόνο λήξης των προαπαιτούμενων δραστηριοτήτων της. Εντοπίζεται ο χρόνος εκκίνησης της δραστηριότητας, όπου πρέπει να είναι μεγαλύτερος ή ίσος του νωρίτερου χρόνου εκκίνησης και να τηρούνται οι περιορισμοί

στους πόρους για την δραστηριότητα καθόλη την διάρκειά της. Αφού βρεθεί ο κατάλληλος χρόνος εκκίνησης προγραμματίζεται η δραστηριότητα και επαναλαμβάνεται η όλη διαδικασία μέχρι να προγραμματιστούν όλες οι δραστηριότητες.

Στην συνέχεια, ο γενετικός αλγόριθμος αφού γνωρίζει τα χρονοπρογράμματα για κάθε χρωμόσωμα ορίζει σαν αρχική καταλληλότητα κάθε χρωμοσώματος την χρονική στιγμή τέλους του έργου και αρχικό πολλαπλασιαστή της καταλληλότητας ίσο με 1. Μετά, ελέγχει για κάθε δραστηριότητα αν ο χρόνος εκκίνησης της στο νέο χρονοπρόγραμμα του χρωμοσώματος αφαιρούμενος τον χρόνο εκκίνησης στο αρχικό χρονοπρόγραμμα είναι μεγαλύτερος από το μισό της διάρκειας της δραστηριότητας. Για κάθε δραστηριότητα που ισχύει αυτό αυξάνει τον πολλαπλασιαστή της καταλληλότητας κατά 0,2. Αφού ολοκληρωθεί αυτή η διαδικασία για κάθε δραστηριότητα πολλαπλασιάζεται η αρχική καταλληλότητα με τον πολλαπλασιαστή της και βρίσκεται η πραγματική καταλληλότητα του χρωμοσώματος.

Από το σύνολο γονέων και παιδιών της γενιάς επιλέγονται τα χρωμοσώματα με τις μικρότερες καταλληλότητες έτσι ώστε να βρεθούν οι γονείς της επόμενης γενιάς. Η όλη διαδικασία επαναλαμβάνεται για όσες γενιές όρισε ο διαχειριστής και στο τέλος της τελευταίας γενιάς επιλέγεται το χρωμόσωμα με την μικρότερη καταλληλότητα ως λύση του προβλήματος.

5 Υλοποίηση

5.1 Εισαγωγή

Η υλοποίηση της προτεινόμενης λύσης έγινε στη γλώσσα προγραμματισμού C# 5.0 της Microsoft και στο πλαίσιο εργασίας .Net Framework 4 της Microsoft επίσης. Οι αλγόριθμοι γραφθήκαν με τη χρήση του Visual Studio 2012. Η επιλογή της γλώσσας προγραμματισμού καθώς και του πλαισίου εργασίας δεν έγιναν αυθαίρετα. Η γλώσσα C# αποτελεί τον συνηθέστερο τρόπο γραφής κώδικα για τα addin του Microsoft Office, λόγω της ευκολίας στην χρήση της σε σχέση με τις υπόλοιπες γλώσσες προγραμματισμού της Microsoft, ενώ το πλαίσιο εργασίας είναι το πλαίσιο που χρησιμοποιούν οι εκδόσεις του Microsoft Office 2010 και μετά.

Η επίλυση υλοποιήθηκε σε δύο κύρια τμήματα: τον γενετικό αλγόριθμο και τον σειριακό αλγόριθμο. Ο σειριακός αλγόριθμος είναι ένας αλγόριθμος της ευρετικής μεθόδου του σειριακού συστήματος παραγωγής χρονοπρογράμματος και ακολουθεί την λογική του ψευδοκώδικα *Αλγόριθμος 1*, ο οποίος θα αναλυθεί στην επόμενη ενότητα.

5.2 Ορισμοί – μαθηματικές σχέσεις

Πριν την ανάλυση των αλγορίθμων της επίλυσης απαιτείται αναφορά σε συγκεκριμμένες μεταβλητές, οι οποίες χρησιμοποιούνται σε αυτούς αλλά δεν ορίζονται από αυτούς. Οι μεταβλητές αυτές αντιστοιχούν σε δεδομένα που αφορούν το έργο και ειδικά στο έργο αφού έχει μετατραπεί στη κλασσική μορφή του RCPSP (Artigues et al., 2003).

Αρχικά υπάρχουν οι μεταβλητές που έχουν σχέση με τους πόρους του έργου:

- Ορίζεται με το σύμβολο $\boldsymbol{\kappa}$ τον αριθμό των ανανεώσιμων πόρων ενός έργου.
- Ορίζεται με το σύμβολο $\boldsymbol{K}$ το σύνολο των ανανεώσιμων πόρων ενός έργου. Το σύνολο έχει μέγεθος $\boldsymbol{\kappa}$.
- Ορίζεται με το σύμβολο $\boldsymbol{k}$ έναν τυχαίο πόρο από το σύνολο $\boldsymbol{K}$. Το $\boldsymbol{k}$ μπορεί να πάρει τιμές από $\boldsymbol{1}$ έως $\boldsymbol{\kappa}$.
- Ορίζεται με το σύμβολο $\boldsymbol{R_k}$ την διαθεσιμότητα του πόρου $\boldsymbol{k}$ σε ολόκληρο το έργο.

Στη συνέχεια οι μεταβλητές που έχουν σχέση με τις δραστηριότητες του έργου:

- Ορίζεται με το σύμβολο $\boldsymbol{n}$ τον αριθμό των δραστηριοτήτων ενός έργου. Όταν το $\boldsymbol{n}$ χρησιμοποιείται σαν δείκτης υποδηλώνει την πλασματική δραστηριότητα λήξης του έργου.
- Ορίζεται με το σύμβολο $\boldsymbol{A}$ το σύνολο των δραστηριοτήτων ενός έργου. Μέσα σε αυτό το σύνολο υπάρχουν και οι πλασματικές δραστηριότητες έναρξης και λήξης του έργου. Το σύνολο έχει μέγεθος $\boldsymbol{n}$.
- Ορίζεται με τα σύμβολα $\boldsymbol{i}$ και $\boldsymbol{j}$ μία τυχαία δραστηριότητα από το σύνολο $\boldsymbol{A}$. Τα $\boldsymbol{i}$ και $\boldsymbol{j}$ μπορούν να πάρουν τιμές από $\boldsymbol{1}$ έως $\boldsymbol{n}$.
- Ορίζεται με το σύμβολο $\boldsymbol{d_i}$ την διάρκεια της δραστηριότητας $\boldsymbol{i}$. Η διάρκεια μετριέται σε χρονικές μονάδες που έχουν οριστεί από το έργο. Συνήθως σε ημέρες εργασίας.
- Ορίζεται με το σύμβολο $\boldsymbol{s_i}$ τον χρόνο έναρξης της δραστηριότητας $\boldsymbol{i}$. Είναι η χρονική στιγμή στην οποία θα ξεκινήσει την εκτέλεσή της η δραστηριότητα.

- Ορίζεται με το σύμβολο f_i τον χρόνο λήξης της δραστηριότητας ***i***. Είναι η χρονική στιγμή στην οποία θα τερματίσει την εκτέλεσή της η δραστηριότητα.
- Ορίζεται με τη λέξη ***lag*** την καθυστέρη ή (?)
- Ορίζεται με το σύμβολο r_{ik} την απαίτηση στον πόρο ***k*** της δραστηριότητας ***i***.
- Ορίζεται με το σύμβολο ***S*** το χρονοπρόγραμμα του έργου.
- Ορίζεται με το σύμβολο ***0*** τον εκθέτη που υποδηλώνει ότι η μεταβλητή ανήκει στο αρχικό χρονοπρόγραμμα. Αρχικό χρονοπρόγραμμα είναι το χρονοπρόγραμμα που υπολογίστηκε κατά τη φάση του προγραμματισμού πριν την εκτέλεση του έργου. Συνεπώς έχουμε:
    - S^0 το αρχικό χρονοπρόγραμμα του έργου,
    - d_i^0 τη διάρκεια της δραστηριότητας ***i*** στο αρχικό χρονοπρόγραμμα,
    - s_i^0 το χρόνο έναρξης της δραστηριότητας ***i*** στο αρχικό χρονοπρόγραμμα,
    - f_i^0 το χρόνο λήξης της δραστηριότητας ***i*** στο αρχικό χρονοπρόγραμμα,
    - R_k^0 την διαθεσιμότητα του πόρου ***k*** στο αρχικό χρονοπρόγραμμα,
    - r_{ik}^0 την απαίτηση στον πόρο ***k*** της δραστηριότητας ***i*** στο αρχικό χρονοπρόγραμμα,

5.3 Σειριακός αλγόριθμος

Ο σειριακός αλγόριθμος βασίζεται πάνω στην ευρετική μέθοδο του σειριακού συστήματος παραγωγής χρονοπρογράμματος (S-SGS). Ξαναδίνεται ο γενικός ψευδοκώδικας του S-SGS (Montoya-Torres et al., 2010) που παρουσιάστηκε στον Αλγόριθμος 1.

Βήμα 1: Αρχικοποίηση
$s_1 = f_1 = 0$
$Sec_1 = \{1\}$
Βήμα 2: FOR $g = 2$ TO $n - 1$ DO
Υπολογίζουμε $D_g, F_g, \bar{R}_k(t)$ για $k \in K$ και $t \in F_g$
Επιλογή $j \in D_g$
$ES_j = \max_{h \in Pred_j} \{f_h\}$
$s_j = \min\{t | t \geq ES_j, t \in F_g, r_{jk} \leq \bar{R}_k(\tau)$ για όλα τα $k \in K, \tau \in [t, t + d_j] \cap F_g\}$
$f_j = s_j + d_j$
$Sec_g = Sec_{g-1} \cup \{j\}$
Βήμα 3: $f_n = \max_{h \in Pred_n} \{f_h\}$

Αλγόριθμος 2: Ψευδοκώδικας για το σειριακό σύστημα παραγωγής χρονοπρογράμματος (Montoya-Torres et al., 2010).

Ο σειριακός αλγόριθμος ακολουθεί τη λογική του ψευδοκώδικα σε μεγάλο βαθμό. Στην συνέχεια θα παρουσιάσουμε τις κύριες μεταβλητές του και τον τρόπο με τον οποίο τις χρησιμοποιούμε συγκεκριμένα στον αλγοριθμό μας.

5.3.1 Ορισμοί και αρχικοποίηση

Ο Αλγόριθμος 2, που παρουσιάζει τον ψευδοκώδικα του Montoya-Torres, μπορεί να απλοποιηθεί ταυτίζοντας το f_j με το F_g. Λόγω αυτής της απλοποίησης χρειάζεται πολύ προσοχή στον ορισμού του των μεταβλητών F_g και D_g . Ορίζεται συνεπώς το F_g ως τον πίνακα μεγέθους ***n***, όπου το $F_g(j)$ αντιστοιχεί στον χρόνο λήξης της δραστηριότητας ***j***, δηλαδή στο f_j. Ιδιαίτερη προσοχή απαιτείται στην αρχικοποίηση αυτού του πίνακα. Επειδή

όλες οι τιμές μπορούν να είναι μόνο θετικές ή μηδέν, αρχικοποιείται στο ***-1*** για να εντοπίζονται και να αποφεύγονται τυχόν σφάλματα στον κώδικα.

Ορίζεται στην συνέχεια το Sec_g ως τον πινάκα μεγέθους ***n***, όπου το $Sec_g(j)$ αντιστοιχεί στο εάν η δραστηριότητα ***j*** έχει προγραμματιστεί ή όχι. Λαμβάνει δύο τιμές μόνο, ***1*** εάν έχει ήδη προγραμματιστεί και ***-1*** αν όχι. Αρχικοποιείται στο ***-1*** διότι με δύο μόνο τιμές οι πιθανότητες να εμφανιστεί σφάλμα είναι μικρές και επιπλέον η μοναδική φορά που αλλάζει τιμή η Sec_g είναι όταν προγραμματίζεται μία δραστηριότητα.

Ορίζεται επιπλέον τα s_j και D_g ως πίνακες μεγέθους ***n.*** Όπως φαίνεται με όλους τους ορισμούς προτιμίθηκε να δουλεύει ο αλγόριθμος με μεταβλητές σε μορφή πινάκων αντί των αντίστοιχων συνόλων (λιστών) του ψευδοκώδικα. Για τον πίνακα s_j ισχύει ότι $s_j(j) = s_j$, δηλαδή απλώς τοποθετήθηκαν οι χρόνοι έναρξης των δραστηριοτήτων ***j*** σε ένα πίνακα. Επειδή, όπως και στην περίπτωση του πίνακα F_g, ο πίνακας s_j μπορεί να έχει τιμές μόνο θετικές ή μηδέν, επιλέχθηκε να αρχικοποιήθει στο ***-1***. Στον πίνακα D_g το $D_g(j)$ αντιστοιχεί στο εάν η δραστηριότητα ***j*** μπορεί να επιλεγεί για προγραμματισμό ή όχι στο στάδιο ***g***. Παίρνει δύο τιμές, ***1*** έαν η δραστηριότητα μπορεί να επιλεγεί και ***0*** αν όχι. Επειδή το D_g αλλάζει τιμές σε κάθε στάδιο είναι χρήσιμο να αρχικοποιήθει στο ***-1***, καθώς και να (?) ξανααρχικοποιείται σε κάθε κύκλο ***g***.

Ορίζεται τέλος την λίστα ***Rkt***. Κάθε στοιχείο της λίστας αποτελείται από μια μεταβλητή ***Fg*** και μία λίστα ***RESOURCEUSED***. Η λίστα ***RESOURCEUSED*** αποτελείται από μία μεταβλητή ***RESID*** και μία μεταβλητή ***RESUSED.*** Η λίστα αρχικοποιείται κενή.

5.3.2 Διαδικασία αλγορίθμου

Η διαδικασία του αλγορίθμου μπορεί να χωριστεί σε εφτά τμήματα, όπου κάθε τμήμα κάνει μια χαρακτηριστική ενέργεια του αλγορίθμου. Όλα τα τμήματα γίνονται κατά σειρά για ***n-2*** κύκλους, δηλάδη για κύκλους ίσους με τις δραστηριότητες του έργου αν αγνοηθούν οι πλασματικές δραστηριότητες αρχής και τέλους του έργου.

Αφού αρχικοποιηθούν οι μεταβλητές του αλγορίθμου, ξεκινούν οι κύκλοι ενεργειών από το ***g=1*** μέχρι το ***n-2***.

5.3.2.1 Επαναφορά

Το πρώτο τμήμα του αλγορίθμου είναι το “Reset” ή αλλίως η επαναφορά. Σε αυτό το τμήμα επαναφέρονται οι μεταβλητές D_g, ***Rkt,*** ES_j και ***j*** στις αρχικές τους τιμές. Συνεπώς ξανααρχικοποιούνται διότι αυτές οι μεταβλητές είναι συνδεδεμένες με κάθε κύκλο και όχι με όλων των αλγόριθμο.

5.3.2.2 Εύρεση του D_g

Στο δεύτερο τμήμα βρίσκονται ποιές δραστηριότητες μπορούν να προγραμματιστούν στον συγκεκριμμένο κύκλο. Για κάθε δραστηριότητα με εξαίρεση την πλασματική δραστηριότητα αρχής ελέγχεται ένα από δύο πράγματα.

Αρχικά ελέγχεται αν η δραστηριότητα ***j*** έχει ήδη προγραμματιστεί, δηλαδή αν $Sec_g(j)=1$. Αν αυτό ισχύει τότε προφανώς δεν μπορεί να ξαναπρογραμματιστεί και άρα το $D_g(j)=0$.

Αν δεν έχει ήδη προγραμματιστεί, τότε ελέγχεται αν όλες οι προκάτοχες δραστηριότητές της έχουν προγραμματιστεί. Αν έστω μία από αυτές δεν έχει προγραμματιστεί θέτει $D_g(j)=0$ και σταματάει την διαδικασία εκεί, αφού δεν έχει νόημα να ελεγχθούν οι υπόλοιπες προκάτοχες δραστηριότητες. Διαφορετικά θέτει $D_g(j)=0$ και ελέγχει την επόμενη δραστηριότητα.

Αφού ελεχθούν όλες οι δραστηριότητες είναι έτοιμος ο πίνακας D_g και συνεχίζει στο επόμενο τμήμα.

5.3.2.3 Εύρεση του Rkt

Σε αυτό το τμήμα γίνεται συμπλήρωση της λίστας ***Rkt*** με τα στοιχεία που απαιτούνται για την λειτουργία του αλγορίθμου. Για κάθε δραστηριότητα ***j*** του έργου δημιουργείται ένα στοιχείο ***rkt*** της λίστας ***Rkt*** χωρίς να το εισαχθεί σε αυτήν. Ελέγχεται αν η δραστηριότητα έχει ήδη προγραμματιστεί και αν δεν ισχύει αυτό συνεχίζει στην επόμενη δραστηριότητα.

Αν όμως η δραστηριότητα έχει προγραμματιστεί, ελέγχει αρχικά αν το $F_g(j)$ είναι ίσο με κάποιο ***Fg*** που μπορεί να έχει ήδη η λίστα ***Rkt.*** Αν ναι, συνεχίζει στην επόμενη δραστηριότητα. Αν όχι, τότε θέτει το ***Fg*** του ***rkt*** ίσο με $F_g(j)$. Στην συνέχεια βρίσκει όλες τις δραστηριότητες που έχουν χρόνο εκκίνησης μικρότερο ή ίσο του ***Fg*** και ταυτόχρονα χρόνο τέλους μεγαλύτερο του ***Fg,*** να μην είναι δηλαδή δραστηριότητες ορόσημο. Δημιουργεί στην λίστα ***RESOURCEUSED*** του ***rkt*** στοιχεία ***κ*** και σε κάθε στοιχείο θέτει το ***RESID*** ίσο με τον αύξοντα αριθμού του πόρου και το ***RESUSED*** ίσο με τους πόρους αυτού του είδους που χρησιμοποιούν όλες οι δραστηριότητες που βρέθηκαν προηγουμένως. Εισάγει, τέλος, την λίστα ***rkt*** στην ***Rkt.***

5.3.2.4 Εύρεση του j

Το ***j*** βρίσκεται με βάση τον κανόνα προτεραιότητας που έχει ορισθεί στην αρχή του κώδικα. Από τους έξι κανόνες που παρουσιάστηκαν στην ενότητα 2.6.2.1.4 υλοποιήθηκαν εξής πέντε: EBST_1, EBST_2, LW, LAN και Random. Ο κανόνας LST δεν υλοποιήθηκε στην παρούσα εργασία διότι απαιτούσε επίλυση με την μέθοδο του κρίσιμου δρόμου και δεν ήταν αυτός ο σκοπός της εργασίας. Προστέθηκε όμως ένας νέος κανόνας προτεραιότητας, αυτός του γενετικού αλγορίθμου.

- EBST_1
  Ελέγχει για κάθε δραστηριότητα που μπορεί να προγραμματιστεί αν ο αρχικός χρόνος εκκίνησής της είναι μικρότερος από την προηγούμενή της. Αν ισχύει ότι είναι μικρότερος επιλέγεται η νέα δραστηριότητα, ενώ αν έχουν ίδιο αρχικό χρόνο επιλέγεται η νέα δραστηριότητα μόνο αν το βάρος της είναι μεγαλύτερο.
- EBST_2
  Ομοίως με το EBST_1 με την διαφορά ότι αν έχουν ίδιους αρχικούς χρόνους παραμένει η παλιά δραστηριότητα.
- LW
  Ελέγχει κάθε δραστηριότητα που μπορεί να προγραμματιστεί και επιλέγεται αυτή με το μεγαλύτερο βάρος.
- LAN
  Επιλέγεται η πρώτη δραστηριότητα του πίνακα D_g που έχει την τιμή 1.

- Random
  Επιλέγεται μία τυχαία δραστηριότητα του πίνακα D_g που έχει την τιμή 1.
- Γενετικός αλγόριθμος
  Επιλέγεται η πρώτη δραστηριότητα στα γονίδια του χρωμοσώματος που μπορεί να προγραμματιστεί.

5.3.2.5 Εύρεση του ES_j

Για την εύρεση του ES_j βρίσκονται όλοι οι προκατόχους της δραστηριότητας ***j*** και αν το F_g ενός προκατόχου αυξημένο κατά το ***LAG*** του είναι μεγαλύτερο από το ES_j θέτει το ES_j ίσο με το F_g αυτού του προκατόχου αυξημένο κατά το ***LAG*** του**.** Αυτή τη διαδικασία γίνεται για όλους τους προκατούχους της ***j***.

5.3.2.6 Προγραμματισμός του j

Για κάθε στοιχείο της λίστας ***Rkt*** ελέγχει αν το ***FG*** του στοιχείου είναι μεγαλύτερο του ES_j και αν ισχύει ένα από τα δύο: το $s_j(j)$ ή είναι μεγαλύτερο του ***FG*** ή έχει την αρχικοποιημένη τιμή ***-1***. Αν δεν ισχύουν αυτά συνεχίζει στο επόμενο στοιχείο.

Αλλιώς, για το συγκεκριμένο στοιχείο, έστω ***i,*** ελέγχει αν όλα τα στοιχεία του ***Rkt,*** συμπεριλαμβανομένου του στοιχείου ***i***, καλύπτουν ένα από τα εξής κριτήρια:

- Το ***FG*** του στοιχείου είναι μεγαλύτερο ή ίσο του ***FG*** του ***i*** και μικρότερο του ***FG*** του ***i*** αυξημένο κατά την διάρκεια της δραστηριότητας ***j.*** Αν ισχύει αυτό ελέγχει για κάθε πόρο αν η απαίτηση στον πόρο της δραστηριότητας ***j*** αυξημένη κατά το ***RESUSED*** του στοιχείου είναι μικρότερη από τη διαθεσιμότητα του πόρου. Αυτό πρέπει να ισχύει για όλους τους πόρους.
- Το ***FG*** του στοιχείου είναι μικρότερο του ***FG*** του ***i.***
- Το ***FG*** του στοιχείου είναι μεγαλύτερο ή ίσο του ***FG*** του ***i*** αυξημένο κατά την διάρκεια της δραστηριότητας ***j.***

Αν καλύφθηκαν τα κριτήρια για όλα τα στοιχεία, θέτει σαν $s_j(j)$ το ***FG*** του ***i*** και θέτει το $Sec_g(j)=1$.

5.3.2.7 Ανανέωση του χρόνου λήξης του έργου

Στο τέλος κάθε κύκλου θέτει τον χρόνο εκκίνησης της πλασματικής δραστηριότητας τέλους ως το μεγαλύτερο από τα διάφορα F_g**.**

5.4 Γενετικός Αλγόριθμος

Ο γενετικός αλγόριθμος έχει τέσσερα στάδια: τον αρχικό πληθυσμό, τη διασταύρωση, την μετάλλαξη και την επιλογή. Πριν ξεκινήσει όμως η ανάλυση του αλγορίθμου πρέπει να οριστούν κάποιες μεταβλητές.

Οι μεταβλητές του αλγορίθμου που χρειάζεται να ορίσει ο χρήστης είναι ο αριθμός των γενεών ***GEN***, ο αριθμός το πληθυσμού ***POP*** και η πιθανότητα μετάλλαξης ***PMUT***. Επίσης ο χρήστης του αλγορίθμου πρέπει να ορίσει την μέθοδο με την οποία θα γίνει η διασταύρωση των χρωμοσωμάτων. Υπάρχουν δύο μεθόδοι για την διασταύρωση: του ενός σημείου και των δύο σημείων.

5.4.1 Ορισμοί

Οι μεταβλητές του αλγορίθμου ***GEN, POP*** και ***PMUT***, που ορίστηκαν στην εισαγωγή αυτής της ενότητας είναι και οι τρεις απλοί ακέραιοι.

Τα χρωμοσώματα του γενετικού αλγορίθμου ορίζονται σαν πίνακες μεγέθους ***n-2***, ίσο δηλαδή με τον αριθμό των δραστηριοτήτων αν αφαιρεθούν οι πλασματικές δραστηριότητες αρχής και τέλους του έργου. Αυτό ορίζεται έτσι διότι πάντα οι πλασματικές δραστηριότητες αρχής και τέλους του έργου θα είναι η πρώτη και η τελευταία αντίστοιχα δραστηριότητα που θα προγραμματίζεται. Συνεπώς δεν προσφέρουν κάτι αν βρίσκονται μέσα στο χρωμόσωμα, αφού το χρωμόσωμα είναι η σειρά προτεραιότητας με την οποία θα προγραμματιστούν οι δραστηριότητες. Κάθε στοιχείου του πίνακα αντιστοιχεί σε ένα γονίδιο του χρωμοσώματος, δηλαδή στο κλειδί μιας δραστηριότητας.

Επιπλέον, ορίζονται τρεις λίστες χρωμοσωμάτων ***PARENTS, CHILDREN*** και ***GENERATION*** οι οποίες αντιστοιχούν στους γονείς, τα παιδιά και την συνολική γενιά αντίστοιχα. Οι τρεις αυτές λίστες ανανεώνονται για κάθε γενιά. (?)

5.4.2 Αρχικός πληθυσμός

Στο στάδιο του αρχικού πληθυσμού παράγονται ***POP*** τυχαία χρωμοσώματα, τα οποία εισάγονται στην λίστα ***PARENTS***. Τα χρωμοσώματα παράγονται με την χρήση ενός τυχαιοποιητή, που χρησιμοποιεί τη συνάρτηση παραγωγής τυχαίων αριθμών της C# (Παράρτημα 3: Τυχαιοποιητής). Παράγονται τυχαίοι αριθμοί από ***2*** έως ***n-1***, αφού ***1*** είναι η πλασματική δραστηριότητα αρχής του έργου και ***n*** η πλασματική δραστηριότητα τέλους του έργου. Αν ένας αριθμός δεν είναι ήδη στο χρωμόσωμα τότε τοποθετείται στην πρώτη ελεύθερη θέση μέσα του, διαφορετικά παράγεται νέος τυχαίος αριθμός.

Αφού ολοκληρωθεί αυτή η διαδικασία για κάθε χρωμόσωμα συνεχίζει στο κυρίως τμήμα του αλγορίθμου το οποίο επαναλαμβάνεται ***GEN*** φορές και αρχίζει με την διασταύρωση.

5.4.3 Διασταύρωση

Στην αρχή του σταδίου της διασταύρωσης έχουμε ***POP*** συνολικά χρωμοσώματα βρίσκονται στη λίστα ***PARENTS.*** Αυτοί οι ***POP*** γονείς θα παράξουν ανά δύο δύο παιδιά. Συνεπώς στο τέλος θα υπάρχουν ***POP*** χρωμοσώματα που θα αποτελούν τα παιδιά της γενιάς και θα εισαχθούν στην λίστα ***CHILDREN.*** Συνεπώς η γενιά θα αποτελείται από ***2* POP*** χρωμοσώματα τώρα.

Η διαδικασία με την οποία παράγονται τα παιδιά είναι το στάδιο της διασταύρωσης. Οι δύο μέθοδοι που χρησιμοποιούνται για την διασταύρωση αναλύθηκαν στην ενότητα 2.6.3.1.2 και δείχτηκαν περιγραμματικά στα Σχήμα 6 έως Σχήμα 12. Στην συνέχεια να αναλυθεί η εφαρμογή τους στο κώδικα του αλγορίθμου μας.

5.4.3.1 Μέθοδος του ενός σημείου

Έχοντας την λίστα ***PARENTS*** λαμβάνονται ανά δύο από την αρχή ζεύγη χρωμοσωμάτων τα οποία θα αποτελέσουν τους γονείς για την διασταύρωση. Συγκεκριμένα το πρώτο χρωμόσωμα θα είναι ο πατέρας και το δεύτερο η μητέρα. Αυτό ισχύει για κάθε ζεύγος.

Για κάθε ζεύγος χρωμοσωμάτων – γονέων λοιπόν χρησιμοποιείται ο τυχαιοποιητή του Παράρτημα 3: Τυχαιοποιητής για να βρεθεί ένας ακέραιος αριθμός ***q,*** που χρειάζεται για την μέθοδο. Ο τυχαιοποιητής βρίσκει ένα τυχαίο αριθμό ανάμεσα στο ***0*** και στο ***n-3***, διότι οι πίνακες στην C# ξεκινάνε την αρίθμησή τους από το ***0***. Λόγω αυτού ένας πίνακας μεγέθους ***n-2*** θα έχει τελευταίο στοιχείο το ***n-3***.

Αφού βρεθεί ο αριθμός ***q*** μεταφέρονται τα στοιχεία ***0*** έως και ***q-1*** του χρωμοσώματος του πατέρα στο χρωμόσωμα του γιού στις αντίστοιχες θέσεις ***0*** έως και ***q-1.*** Η ίδια ακριβώς διαδικασία γίνεται και για τα χρωμοσώματα της μητέρας και της κόρης.

Στην συνέχεια, για να συμπληρωθούν τα στοιχεία ***q*** έως και ***n-3*** του χρωμοσώματος του γιού ελέγχονται σε σειρά τα στοιχεία του χρωμοσώματος της μητέρας για γονίδια που δεν υπάρχουν ήδη στο χρωμόσωμα του γιού και τοποθετούνται σε σειρά στα κενά στοιχεία του χρωμοσωματός του. Παρομοίως συμπληρώνεται το γονίδιο της κόρης ελέγχοντας το χρωμόσωμα του πατέρα.

Τα χρωμοσώματα των παιδιών εισάγονται μετά το τέλος της διαδικασίας για κάθε ζεύγος γονέων στη λίστα ***CHILDREN.***

5.4.3.2 Μέθοδος των δύο σημείων

Χωρίζονται οι γονείς της λίστας ***PARENTS*** με τον ίδιο τρόπο που χωρίστηκαν στη μέθοδο του ενός σημείου. Υπάρχουν συνεπώς ζεύγη χρωμοσωμάτων στα οποία ορίζεται το ένα σαν πατέρα και το άλλο σαν μητέρα για την διαδικασία της διασταύρωσης.

Για κάθε ζεύγος χρειάζεται να βρεθούν δύο ακέραιους αριθμούς q_1 και q_2. Για την εύρεση των δύο αυτών μεταβλητών χρησιμοποιείται ο τυχαιοποιητής του Παράρτημα 3: Τυχαιοποιητής. Ο αριθμός q_1 είναι ένας τυχαίος αριθμός ανάμεσα στο ***0*** και στο ***n-4***, ενώ ο αριθμός q_2 είναι ένας τυχαίος αριθμός ανάμεσα στο q_1 και στο ***n-3***.

Μετά την εύρεση των q_1 και q_2 μεταφέρονται τα στοιχεία ***0*** έως και q_1***-1*** του πατέρα στο γιό στις αντίστοιχες θέσεις ***0*** έως και q_1***-1,*** και εκτελείται η αντίστοιχη διαδικασία για τη μητέρα και τη κόρη.

Στην συνέχεια, τοποθετούνται στις θέσεις q_1 έως και q_2***-1*** του χρωμοσώματος του γιού στοιχεία της μητέρας τα οποία δεν βρίσκονται ήδη στον γιό, και εκτελείται το ίδιο για τον πατέρα και την κόρη. Τέλος, συμπληρώνονται οι τελευταίες θέσεις από q_2 έως και ***n-3*** του γιού με στοιχεία του πατέρα και της κόρης με στοιχεία της μητέρας.

Τα χρωμοσώματα των παιδιών εισάγονται μετά το τέλος της διαδικασίας για κάθε ζεύγος γονέων στη λίστα ***CHILDREN.***

5.4.4 Μετάλλαξη

Αφού παραχθούν τα παδιά με την διαδικασία της διασταύρωσης, ακολουθεί η μετάλλαξη τους. Για κάθε χρωμόσωμα της λίστας ***CHILDREN*** επιλέγονται δύο τυχαίοι αριθμοί ***i*** και ***j*** από το ***0*** έως και το ***n-3,*** διαφορετικοί μεταξύ τους, με την χρήση του τυχαιοποιητή του Παράρτημα 3: Τυχαιοποιητής. Στην συνέχεια τα στοιχεία ***i*** και ***j*** του χρωμοσώματος αλλάζουν θέση μεταξύ τους.

5.4.5 Επιλογή

Μετά την διαδικασία της μετάλλαξης, ενώνονται οι δύο λίστες, ***PARENTS*** και ***CHILDREN***, στη λίστα ***GENERATION*** και καθαρίζονται οι λίστες ***PARENTS*** και ***CHILDREN***. Έχοντας τώρα μία λίστα μεγέθους ***2*POP***, που περιέχει όλα τα χρωμοσώματα της γενιάς, των γονέων και των παιδιών, πρέπει να βρεθούν τα ***POP*** καλύτερα από αυτά και να μεταφερθούν στην λίστα ***PARENTS*** για να συνεχίσει με αυτά η επόμενη γενιά.

Για κάθε χρωμόσωμα της λίστας ***GENERATION*** βρίσκεται η καταλληλότητά του με την χρήση του σειριακού αλγορίθμου. Βρίσκεται για κάθε δραστηριότητα του έργου ο χρόνος εκκίνησής της με βάση την σειρά προτεραιότητας που έχει ορίσει το χρωμόσωμα και ορίζεται σαν αρχική καταλληλότητα του χρωμοσώματος ***FITNESS o*** χρόνος εκκίνησης της πλασματικής δραστηριότητας τέλους του έργου. Ορίζεται μια μεταβλητή ***FIT_MULTIPLIER***, που υποδηλώνει κατά πόσο διαφέρει το νέο χρονοπρόγραμμα από το αρχικό και αρχικοποιείται στο 1. Στην συνέχεια για κάθε δραστηριότητα ***j*** βρίσκουμε τον αρχικό χρόνο εκκίνησης της $s^0(j)$ και τον νέο χρόνο εκκίνησης της ***s'(j)***, που προέκυψε από τον σειριακό αλγόριθμο. Αν η διαφορά $s'-s^0$ είναι μεγαλύτερη από το ***d(j)/2***, δηλαδή το μισό της διάρκειας της δραστηριότητας, τότε η μεταβλητή ***FIT_MULTIPLIER*** αυξάνει κατά 0,2 , διαφορετικά παραμένει το ίδιο. Αφού γίνει αυτό για κάθε δραστηριότητα προκύπτει ένας νέος αριθμός ***FIT_MULTIPLIER*** ο οποίος πολλαπλασιάζεται με το ***FITNESS*** για να προκύψει η αληθινή καταλληλότητα του χρωμοσώματος. Ο νέος αριθμός ***FITNESS*** εισάγεται στην μεταβλητή ***FITNESS.***

Αφού έχουν βρεθεί όλα τα ***FITNESS*** της λίστας ***GENERATION*** ταξινομείται η λίστα σε αύξουσα σειρά του ***FITNESS***. Στη συνέχεια επιλέγονται τα ***POP*** πρώτα χρωμοσώματα της λίστας και εισάγονται στην λίστα ***PARENTS*** και επαναλαμβάνεται η διαδικασία μέχρι την ***GEN*** γενιά.

6 Επαλύθευση αποτελεσματών και έλεγχος ορθότητας

6.1 Εισαγωγή

Για την επαλύθευση των αποτελεσμάτων έγιναν πειράματα του κώδικα στο MS-Project με την βοήθεια των έργων της PSPLIB (Kolisch and Sprecher, 1997). Συγκεκριμένα ελέγχθηκαν τα έργα J301_1 μέχρι J301_10.

6.2 Παράμετροι πειράματος

Όλα τα έργα είχαν προγραμματιστεί με την βοήθεια του MS-Project σε ένα αρχικό χρονοπρόγραμμα και θεωρείται ότι το έργο βρισκόταν στην χρονική περίοδο ***t=20*** στην διαδικασία του.

6.2.1 Διαφοροποιήσεις

Σε αυτή τη χρονική περίοδο προκλήθηκαν 10 τυχαίες διαφοροποιήσεις, 5 διαφοροποιήσεις διάρκειας και 5 διαφοροποιήσεις πόρων. Η κάθε μη ολοκληρωμένη δραστηριότητα μπορούσε να διαφοροποιηθεί μόνο μία φορά ή την διάρκειά της ή τους πόρους της, αλλά όχι και τα δύο. Οι διαφοροποιήσεις προκαλούσαν μια αύξηση της τάξεω ς του 50% στην συγκεκριμμένη μεταβλητή και στην περίπτωση των πόρων αν αυτή η αύξηση προκαλούσε απαίτηση σε πόρο μεγαλύτερη της διαθεσιμότητας του πόρου, η απαίτηση θέτονταν ως η διαθεσιμότητα του πόρου. Όλες οι δραστηριότητες είχαν την ίδια πιθανότητα να διαφοροποιηθούν.

6.2.2 Γενετικός αλγόριθμος

Μετά την προκάλεση των τυχαίων διαφοροποιήσεων εκτελέστηκε ο γενετικός αλγόριθμος για την εύρεση του νέου χρονοπρογράμματος. Ο αλγόριθμος έτρεξε για ***Gen=4, Pop=10*** και ***Pmut=20%*** με την μέθοδο διασταύρωσης των δύο σημείων.

Επιπλέον απαιτήθηκε όλες οι δραστηριότητες που ήταν ενεργές την χρονική στιγμή ***t*** να μην χωριστούν και αν είναι εφικτό να προγραμματιστούν τη νωρίτερη χρονική στιγμή που μπορούν.

6.3 Αποτελέσματα

6.3.1 Σύνοψη

Τα αποτελέσματα από τα πειράματα με την ομάδα J301 της PSPLIB παρουσιάζονται συνοπτικά στον Πίνακας 6: Αποτελέσματα ομάδας J301. Θα παρουσιάσουμε τα αποτελέσματα του κάθε έργου της ομάδας σε διαφορετικούς πίνακες έτσι ώστε να φανεί καλύτερα η διαφορά του αρχικού χρονοπρογράμματος από το νέο σε κάθε δραστηριότητα που μένει να εκτελεστεί.

Όνομα Έργου	S^0	S'	$\Delta(S^0,S')$
J301_1	48	53	5
J301_2	46	59	13
J301_3	46	56	10
J301_4	62	71	9
J301_5	47	47	0
J301_6	48	55	7
J301_7	60	63	3
J301_8	53	61	8
J301_9	57	63	6
J301_10	46	57	11

Πίνακας 6: Αποτελέσματα ομάδας J301

6.3.2 Αναλυτικά

Σε κάθε έργο η πλασματική δραστηριότητα της αρχής ονομάζεται "Start" και του τέλους "Finish", ενώ οι υπόλοιπες δραστηριότητες έχουν σαν όνομα το κλειδί τους. Η απαίτηση σε πόρους παρουσιάζεται με ένα νούμερο και δίπλα του σε παρένθεση το όνομα του πόρου. Τα ονόματα των πόρων είναι ίδια με τα κλειδιά τους. Τέθηκε ο αρχικός χρόνος εκκίνησης της δραστηριότητας ίσο με ***20*** αν η δραστηριότητα είναι ήδη σε εξέλιξη την χρονική στιγμή ***20*** και βάλθηκε σε παρένθεση δίπλα τον πραγματικό αρχικό χρόνο εκκίνησης με αστερίσκο.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S'	$\Delta(S^0,S')$
Start	0	0	20	20	0
6	12	8(4)	39	39	0
11	6	5(2)	20(12*)	20	0
15	6	3(1)	20(12*)	20	0
16	8	5(4)	20(12*)	29	9
17	9	8(4)	23	20	-3
19	1	2(2)	20(18*)	20	0
20	7	13(2)	21	28	7
21	2	9(4)	37	37	0
22	7	3(1)	29	37	8
23	2	3(1)	36	44	8
24	3	13(2)	38	46	8
25	3	4(1)	28	35	7
26	7	4(3)	21	26	5
27	8	7(4)	29	29	0
28	3	8(2)	41	39	-2
29	7	7(2)	28	21	-7
30	2	7(2)	47	51	4
31	2	2(3)	44	42	-2
Finish	0	0	48	53	5

Πίνακας 7: Στοιχεία έργου J301_1

Στο έργο J301_1 (Πίνακας 7) παρατηρήθηκε ομαλή αλλαγή στους χρόνους εκκίνησης με 5 να είναι ακριβώς ίδιοι, 5 να είναι σχετικώς μικροί ενώ 7 παρουσιάζουν αρκετή διαφορά. Ο χρόνος τέλους του έργου διαφέρει μόνο 5 χρονικές μονάδες.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$Δ(S^0,S')$
Start	0	0	20	20	0
11	7	2(1)	20(13*)	20	0
14	3	10(3)	23	20	-3
16	9	10(1)	20(14*)	20	0
17	8	7(3)	20(13*)	23	3
18	10	5(1)	24	29	5
19	8	10(4)	20(18*)	20	0
20	7	4(3)	25	29	4
21	2	10(2)	23	39	16
23	2	12(1)	33	39	6
24	2	12(4)	30	30	0
25	2	7(4)	28	28	0
26	9	7(1)	24	41	17
27	1	1(4)	33	50	17
28	4	8(1)	35	29	-6
29	7	3(2)	32	32	0
30	8	8(3)	39	51	12
31	9	6(2)	35	41	6
Finish	0	0	46	59	13

Πίνακας 8: Στοιχεία έργου J301_2

Στο έργο J301_2 (Πίνακας 8) παρατηρήθηκε ότι 4 δραστηριότητες έχουν μεγάλη διαφορά στους χρόνους εκκίνησής τους, αλλά υπάρχει και αρκετή διαφορά στον χρόνο τέλους του έργου. Αυτό είναι λογικό αποτέλεσμα αφού οι τυχαίες διαφοροποιήσεις είναι ποσοστιαίες άρα η αλλαγή σε διάρκεια ή πόρους μπορεί να προκαλέσει μεγάλα προβλήματα αν είναι σε δραστηριότητες με μεγάλες διάρκειες ή ανάγκες σε πόρους.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$Δ(S^0,S')$
Start	0	0	20	20	0
8	5	10(3)	20(17*)	20	0
11	5	9(1)	20(13*)	20	0
14	4	6(4)	22	25	3
16	15	3(1)	28	25	-3
17	3	3(3)	21	25	4
19	1	4(4)	25	29	4
21	7	3(2)	22	25	3
22	9	12(4)	36	32	-4
23	9	10(3)	22	25	3
24	4	6(2)	23	32	9
25	4	3(2)	38	40	2
26	1	4(3)	42	44	2
27	1	9(1)	27	40	13
28	8	9(4)	28	41	13
29	1	1(4)	45	41	-4
30	2	8(2)	45	45	0
31	7	4(2)	36	49	3
Finish	0	0	46	56	10

Πίνακας 9: Στοιχεία έργου J301_3

Στο έργου J301_3 (Πίνακας 9) παρουσιάζεται το ίδιο πρόβλημα με το J301_2, 3 δραστηριότητες έχουν μεγάλη διαφορά στους χρόνους αλλά έχουν και μεγάλες αναγκές σε πόρους.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$\Delta(S^0,S')$
Start	0	0	20	20	0
3	15	8(2)	25	20	-5
8	7	4(3)	20(17*)	20	0
9	14	10(4)	24	27	3
10	6	4(4)	24	27	3
11	6	9(4)	33	41	8
12	9	4(3)	28	33	5
13	5	11(2)	20(18*)	35	15
16	7	14(4)	37	47	10
17	1	3(2)	37	47	10
18	7	4(4)	28	40	12
20	9	7(2)	37	42	5
21	2	3(3)	25	40	15
22	7	2(2)	24	27	3
23	3	4(4)	35	34	-1
24	10	10(3)	37	42	5
25	1	1(4)	38	42	4
26	4	5(1)	33	41	8
27	1	7(3)	20(17*)	20	0
28	1	1(3)	44	54	10
29	10	6(2)	46	51	5
30	9	6(4)	46	62	16
31	8	10(4)	55	54	-1
Finish	0	0	62	71	9

Πίνακας 10: Στοιχεία έργου J301_4

Στο έργου J301_4 (Πίνακας 10) παρατηρήθηκαν μεγάλες διαφορές σε 3 δραστηριότητες που δεν έχουν ούτε μεγάλη διάρκεια ούτε μεγάλες ανάγκες σε πόρους, αλλά λόγω των πολλών δραστηριοτήτων προς προγραμματισμό οι 3 δραστηριότητες δεν παρουσιάζουν μεγάλο πρόβλημα αν προγραμματιστούν αρκετά μακριά του αρχικού τους χρόνου.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$\Delta(S^0,S')$
Start	0	0	20	20	0
12	3	7(1)	20(17*)	20	0
20	4	5(3)	20	20	0
21	6	2(4)	20(16*)	20	0
22	5	1(4)	20(19*)	20	0
23	9	5(4)	20(17*)	20	0
24	1	8(4)	25	29	4
25	8	11(2)	32	32	0
26	7	8(2)	20(19*)	20	0
27	5	11(2)	27	27	0
28	1	6(2)	40	40	0
29	5	4(2)	40	40	0
30	2	9(2)	45	45	0
31	1	10(4)	26	30	4
Finish	0	0	47	47	0

Πίνακας 11: Στοιχεία έργου J301_5

Στο έργο J301_5 (Πίνακας 11) παρατηρήθηκε ότι οι διαφοροποιήσεις δεν προκάλεσαν μεγάλο πρόβλημα, παραμόνο στον προγραμματισμό 2 δραστηριοτήτων οι οποίες έπρεπε να προγραμματιστούν απλώς σε 4 χρονικές μονάδες μετά.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$\Delta(S^0,S')$
Start	0	0	20	20	0
9	3	10(1)	20(17*)	22	2
17	4	7(4)	20	20	0
18	15	9(4)	23	24	1
19	2	6(2)	20	20	0
20	4	5(1)	33	39	6
22	2	12(1)	21	20	-1
24	2	4(1)	21	25	4
25	4	10(2)	21	22	1
26	6	10(4)	42	48	6
27	9	10(4)	33	39	6
28	2	5(4)	33	22	-11
29	1	9(3)	48	54	6
30	1	9(3)	44	48	4
31	9	4(3)	35	39	4
Finish	0	0	48	55	7

Πίνακας 12: Στοιχεία έργου J301_6

Στο έργο J301_6 (Πίνακας 12) όλες οι δραστηριότητες έχουν μικρές διαφορές χρόνων, με εξαίρεση μια δραστηριότητα η οποία απαιτείται να προγραμματιστεί αρκετά νωρίτερα από το προβλεπόμενο στο αρχικό χρονοπρόγραμμα.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$Δ(S^0,S')$
Start	0	0	20	20	0
8	3	9(2)	22	24	2
11	4	5(2)	20(18*)	20	0
12	2	5(4)	20(18*)	20	0
14	10	4(1)	22	24	2
15	9	5(3)	31	37	6
16	8	11(3)	21	22	1
17	5	2(3)	29	30	1
19	8	10(3)	37	46	9
20	5	12(2)	34	35	1
21	2	11(3)	29	35	6
22	1	8(1)	29	34	5
23	3	1(4)	30	35	5
25	5	1(2)	39	40	1
26	10	9(4)	34	37	3
27	8	4(2)	44	47	3
28	5	1(3)	44	45	1
29	5	8(3)	45	54	9
30	8	5(1)	52	55	3
31	6	6(4)	49	50	1
Finish	0	0	60	63	3

Πίνακας 13: Στοιχεία έργου J301_7

Στο έργο J301_7 (Πίνακας 13) παρατηρήθηκαν ομαλές αλλαγές στους χρόνους εκκίνησης με μέγιστη διαφορά 9 χρονικών μονάδων σε 2 μόνο δραστηριότητες με μεγάλες αναγκες πόρων.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$Δ(S^0,S')$
Start	0	0	20	20	0
9	8	8(2)	20	20	0
15	2	4(3)	20	20	0
16	8	3(4)	20	20	0
18	15	10(3)	25	28	3
19	3	3(3)	20(19*)	20	0
21	10	3(2)	35	43	8
22	1	8(3)	35	43	8
23	1	3(2)	25	38	13
24	3	4(1)	26	39	13
25	10	12(2)	25	28	3
26	1	3(4)	35	38	3
27	1	1(2)	36	44	8
28	7	5(3)	36	44	8
29	8	10(2)	45	53	8
30	9	7(3)	35	44	9
31	2	6(2)	43	51	8
Finish	0	0	53	61	8

Πίνακας 14: Στοιχεία έργου J301_8

Στο έργο J301_8 (Πίνακας 14) παρατηρήθηκε ότι για διαφορά 8 χρονικών μονάδων στον χρόνο τέλους του έργου χρειάζεται να χρονοπρογραμματιστούν οι περισσότερες δραστηριότητες αρκετά μακριά από το αρχικό χρονοπρόγραμμα, κυρίως οι τελευταίες.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$\Delta(S^0,S')$
Start	0	0	20	20	0
7	9	8(1)	38	37	-1
11	4	5(3)	20(15*)	20	0
12	8	10(3)	44	46	2
13	7	2(1)	20(19*)	20	0
14	12	6(2)	28	28	0
15	2	6(1)	20(15*)	20	0
16	7	10(1)	31	30	-1
18	6	11(2)	22	22	0
19	7	9(1)	24	47	13
21	8	9(1)	20(19*)	22	2
22	10	7(4)	28	28	0
23	7	5(4)	39	54	15
24	1	10(4)	20(15*)	20	0
25	5	3(4)	28	30	2
26	1	9(4)	38	38	0
27	1	9(1)	44	46	2
28	5	8(3)	49	54	5
29	1	5(1)	46	61	15
30	4	1(2)	54	59	5
31	2	6(2)	38	40	2
Finish	0	0	57	63	6

Πίνακας 15: Στοιχεία έργου J301_9

Στο έργο J301_9 (Πίνακας 15) παρατηρήθηκαν μεγάλες διαφορές χρόνων εκκίνησης σε 3 δραστηριότητες με σχετικώς μεγάλους χρόνους και σχετικά μεγάλες ανάγκες σε πόρους με εξαίρεση μία που διαρκεί μόνο μια χρονική μονάδα.

Όνομα δραστηριότητας	Διάρκεια	Πόροι	S^0	S’	$\Delta(S^0,S')$
Start	0	0	20	20	0
4	7	4(4)	20(12*)	20	0
6	8	10(2)	27	29	2
10	8	6(4)	26	27	1
12	7	3(2)	20(12*)	20	0
14	4	2(3)	20(18*)	20	0
15	5	8(2)	22	24	2
16	3	2(4)	18	20	2
17	4	6(4)	22	42	20
18	3	9(3)	22	27	5
19	7	4(4)	34	35	1
21	2	2(3)	20	20	0
23	8	10(1)	25	30	5
24	3	7(3)	26	46	20
25	3	5(1)	22	27	5
26	2	8(1)	33	38	5
27	10	5(3)	29	30	1
28	3	9(4)	31	49	18
29	5	8(1)	40	52	12
30	5	7(1)	35	43	8
31	1	5(1)	45	42	-3
Finish	0	0	46	57	11

Πίνακας 16: Στοιχεία έργου J301_10

Στο έργο J301_10 (Πίνακας 16) παρατηρήθηκαν ξανά μεγάλες διαφορές στους χρόνους εκκίνησης δραστηριοτήτων με μεγάλες ανάγκες στους πόρους τους.

Γενικώς, ο αλγόριθμος έδωσε μία λύση αρκετά κοντινή στο αρχικό χρονοπρόγραμμα. Το κύριο πρόβλημα το παρουσιάζουν οι δραστηριότητες που έχουν μεγάλες ανάγκες σε πόρους και άρα χρειάζεται να μετακινηθούν αρκετά έτσι ώστε να μην χρειαστεί να μετακίνηθουν πολλές άλλες δραστηριότητες περισσότερο. Αυτό ούτως ή αλλως είναι και το επιθυμητό.

7 Διεπαφή χρήστη – μηχανής και μελέτη περίπτωσης

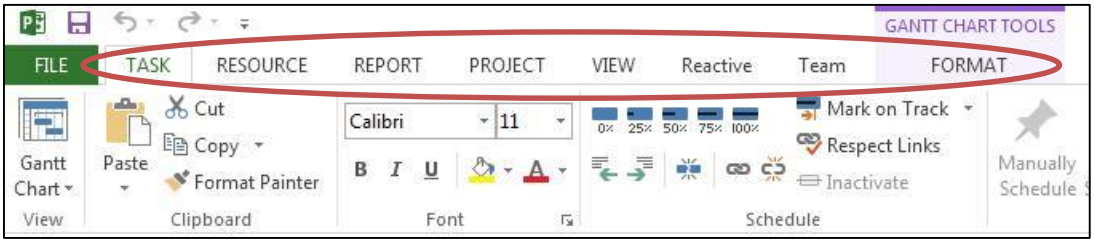
7.1 Εισαγωγή

Όπως παρουσιάστηκε στο κεφάλαιο 3 το πρόβλημα το οποίο ζητείται να επιλυθεί περιέχει δεδομένα τα οποία θεωρήθηκαν ανεξάρτητα της επίλυσης, διότι την επηρεάζουν σε αμελητέο ποσοστό. Το πρόβλημα πρέπει να μετατραπεί σε μία μορφή που να είναι επιλύσιμη από τους αλγορίθμους μας και για αυτόν το λόγο δημιουργήθηκε ένα add-in στο MS-Project 2013, που κάνει αυτή τη μετατροπή και επιλύει και στην συνέχεια το πρόβλημα.

Παρουσιάζονται στις επόμενες ενότητες βήμα προς βήμα τις διάφορες λειτουργίες που έχει το add-in, καθώς και το πως βοηθούν το χρήστη για να επιλύση το πρόβλημα που έχει μπροστά του.

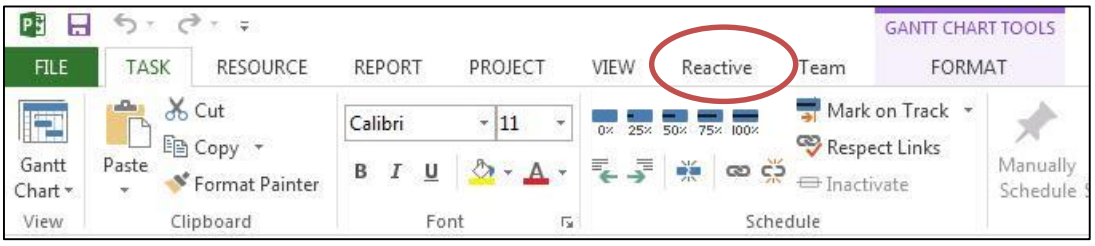
7.2 Ribbon

Στο MS-Project 2013 οι διάφορες εντολές που μπορεί να χρησιμοποιήσει ο χρήστης βρίσκονται στο πάνω μέρος του προγράμματος. Οι εντολές είναι ομαδοποιημένες σε διάφορες καρτέλες και η επιλογή μιας καρτέλας εμφανίζει τις εντολές που τις αντιστοιχούν (Εικόνα 1).



Εικόνα 1: Καρτέλες MS-Project 2013.

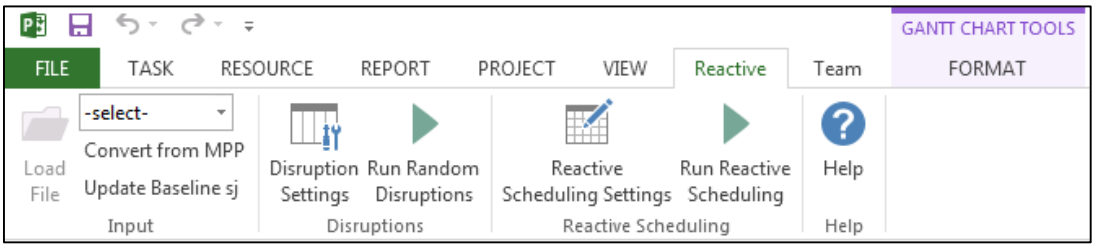
Όταν δημιουργείται ένα νέο add-in για το MS-Project, δημιουργείται αυτόματα μέσα στο add-in μία νέα καρτέλα με το όνομα που της δίνει ο προγραμματιστής έτσι ώστε να φαίνεται στο χρήστη του προγράμματος (Εικόνα 2).



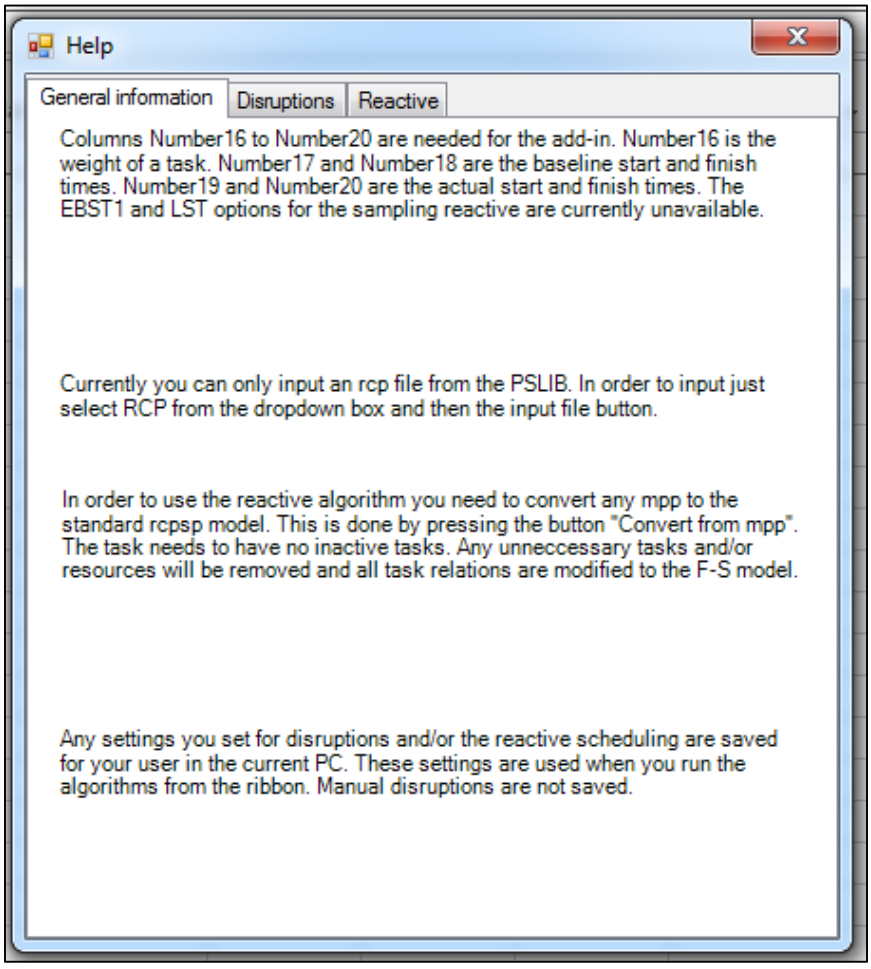
Εικόνα 2: Καρτέλα add-in.

Σε αυτή τη νέα καρτέλα ο προγραμματιστής μπορεί να ξεκινήσει να εισάγει τις εντολές του add-in του με όποιον τρόπο επιθυμεί. Δημιουργήθηκαν τέσσερις ομάδες εντολών για την καρτέλα μας: την ομάδα εισαγωγής δεδομένων, ομάδα διαφοροποιήσεων, ομάδα αναδραστικής μεθόδου και την ομάδα βοήθειας (Εικόνα 3). Οι τρεις πρώτες ομάδες θα αναλυθούν λεπτομερώς στις επόμενες ενότητες, ενώ η τελευταία ομάδα περιέχει μόνο ένα κουμπί με τον τίτλο “Help”, η οποία εμφανίζει ένα παράθυρο που περιέχει διάφορες πληροφορίες σχετικά με τις λειτουργίες του add-in για να βοηθήσει τον χρήστη (Εικόνα 4).

Το παράθυρο χωρίζεται με καρτέλες σε τρία μέρη: τις γενικές πληροφορίες (General Information), τις πληροφορίες για τις διαφοροποιήσεις (Disruptions) και τις πληροφορίες για την αναδραστική μέθοδο (Reactive).



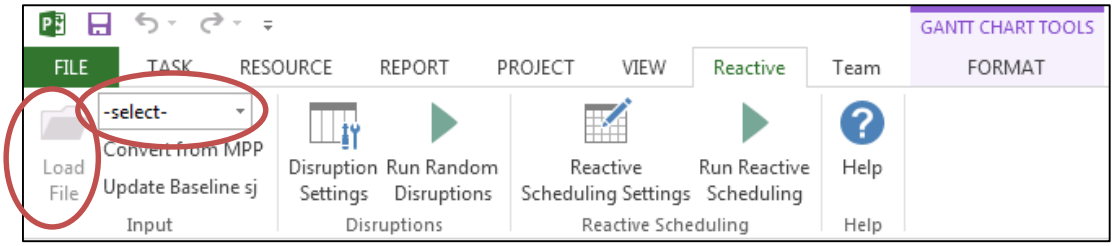
Εικόνα 3: Εντολές καρτέλας add-in.



Εικόνα 4: Παράθυρο βοήθειας.

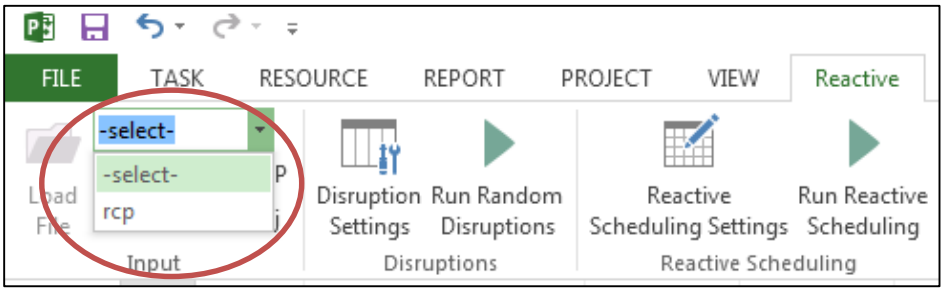
7.3 Εισαγωγή δεδομένων

Η πρώτη ομάδα στο ribbon του add-in περιέχει τις μεθόδους με τις οποίες μπορεί ο χρήστης να εισάγει τα δεδομένα ενός έργου στην μορφή που απαιτείται για την λειτουργία του αλγορίθμού. Αρχικά, υπάρχει η εισαγωγή από αρχείο άλλης μορφής από mpp, που είναι ο τύπους αρχείου που χρησιμοποιεί το MS-Project. Η εισαγωγή γίνεται με την χρήση ενός dropdown box και του κουμπιού “Load File” (Εικόνα 5).

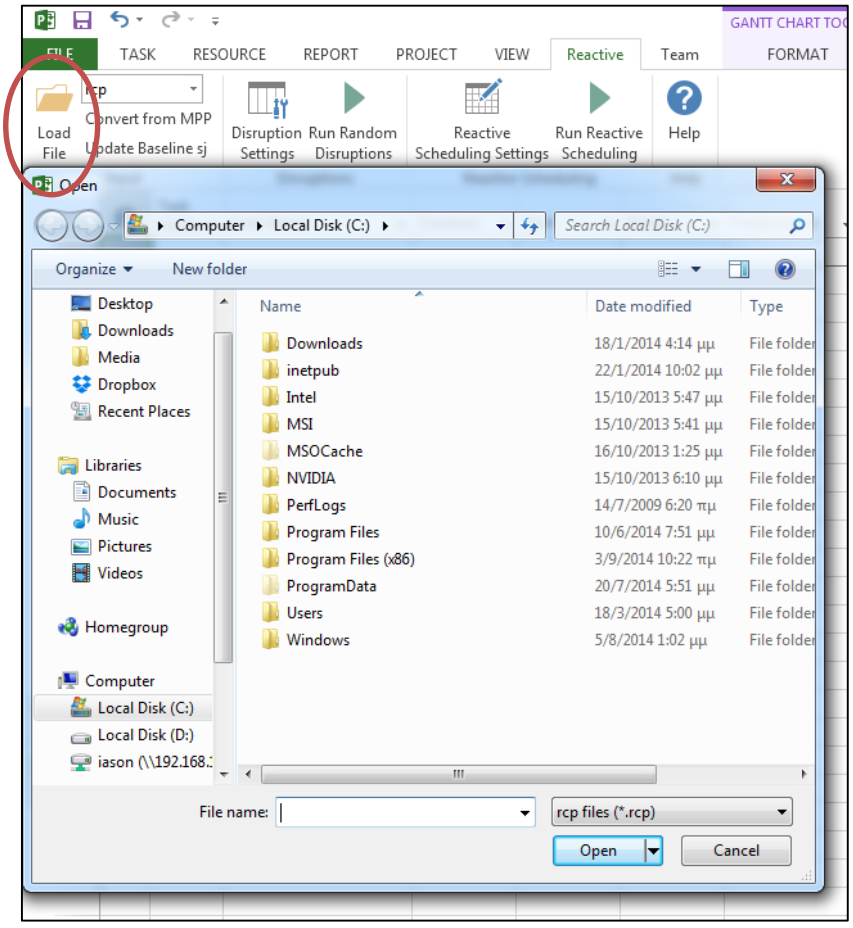


Εικόνα 5: Εισαγωγή αρχείου.

Ο χρήστης επιλέγει από το dropdown box το είδος του αρχείου που επιθυμεί να εισάγει στο πρόγραμμα (Εικόνα 6) και μετά κάνοντας κλίκ στο κουμπί “Load File”, ανοίγει το παράθυρο εύρεσης αρχείου για άνοιγμα (Εικόνα 7).



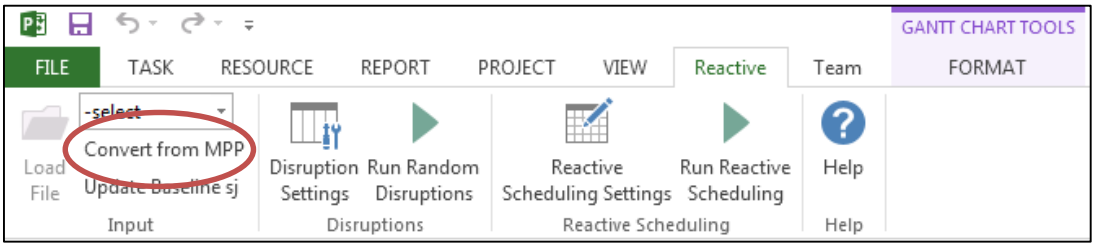
Εικόνα 6: Επιλογή είδους αρχείου για εισαγωγή.



Εικόνα 7: Παράθυρο ανοίγματος αρχείου.

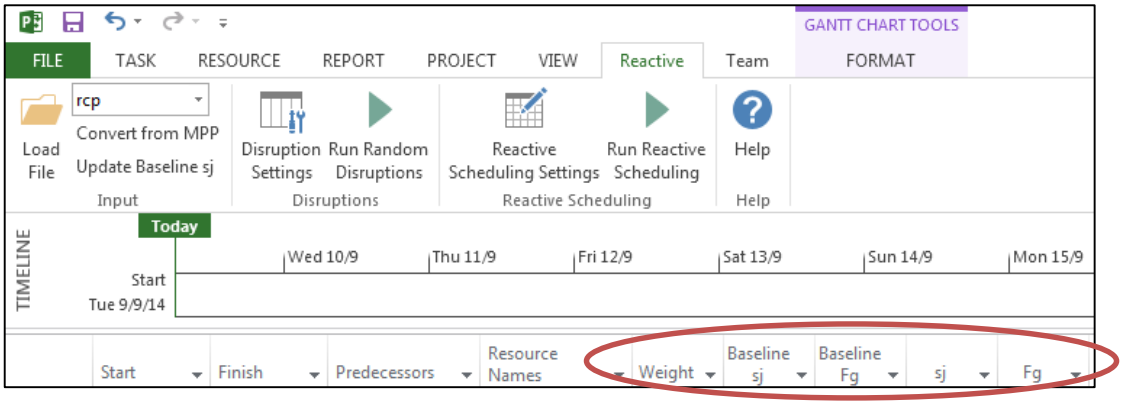
Το είδους αρχείου RCP, που εμφανίζεται στην επιλογή της Εικόνα 6, είναι είδος αρχείο που διατίθεται για τον έλεγχο της αποτελεσματικότητας των αλγορίθμων που επιλύουν το πρόβλημα του χρονοπρογραμματισμού και έχουν συγκεκριμένη κωδικοποίηση. Παρέχεται η δυνατότητα εισόδου τους στο εργαλείο ώστε να είναι δυνατή και η άμεση χρήση του από την επιστημονική κοινότητα και όχι μόνο από τους διαχειριστές έργων στην πράξη.

Αν ο χρήστης έχει ήδη ένα αρχείο τύπου mpp ανοικτό στο πρόγραμμά του, χρειάζεται να πατήσει το κουμπί “Convert from MPP” για να μετατρέψει το έργο που είναι ανοιγμένο εκείνη τη στιγμή στη μορφή που χρειάζεται ο αλγόριθμος (Εικόνα 8). Για να γίνει η μετατροπή πρέπει να ισχύουν κάποιες προυποθέσεις, οι οποίες εξηγούνται στο παράθυρο βοηθείας, αλλά και εμφανίζονται μηνύματα λάθους σαν ο χρήστης προσπαθεί να κάνει την διαδικασία χωρίς να τα έχει καλύψει.



Εικόνα 8: Μετατροπή από αρχείο τύπου mpp.

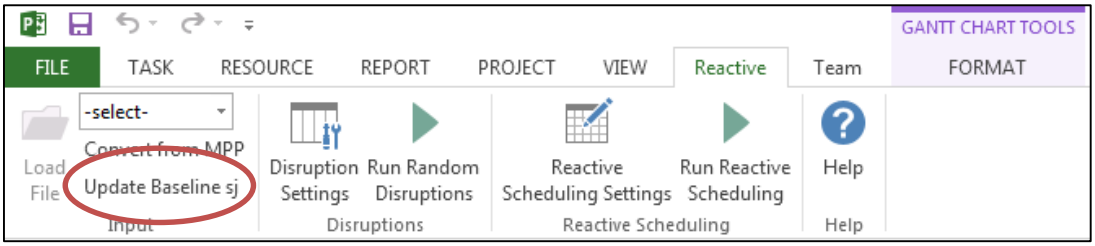
Ο αλγόριθμος, όμως παρουσιάστηκε στο κεφάλαιο 0, κάνει χρήση μεταβλητών ο οποίες δεν βρίσκονται κανονικά στο MS-Project. Για να μπορέσει το add-in να διατηρήσει και να χρησιμοποιήσει αυτές τις μεταβλητές χρησιμοποιεί τις κολώνες “Number16” ‘εως “Number20” για τις μεταβλητές Weight***, Baseline sj, Baseline Fg, sj*** και ***Fg*** (Εικόνα 9).



Εικόνα 9: Κολώνες του add-in.

Η τελευταία διαδικασία που υπάρχει στην ομάδα εισαγωγής δεδομένων είναι η ανανέωση του αρχικού χρονοπρογράμματος του αλγορίθμου από τα δεδομένα που βρίσκονται στο έργου του MS-Project εκείνη τη στιγμή. Αυτό γίνεται με την χρήση του κουμπιού “Update Baseline sj”.

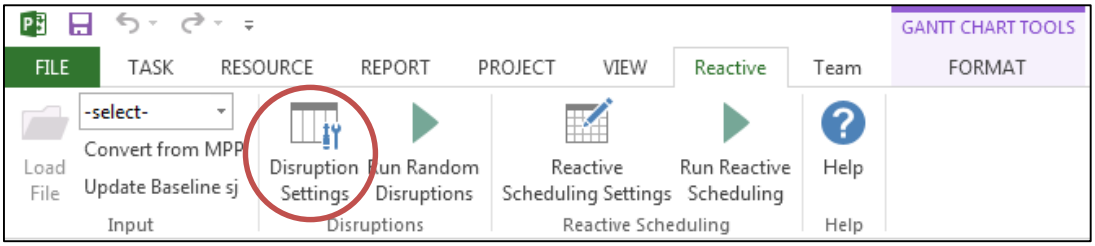
(?)



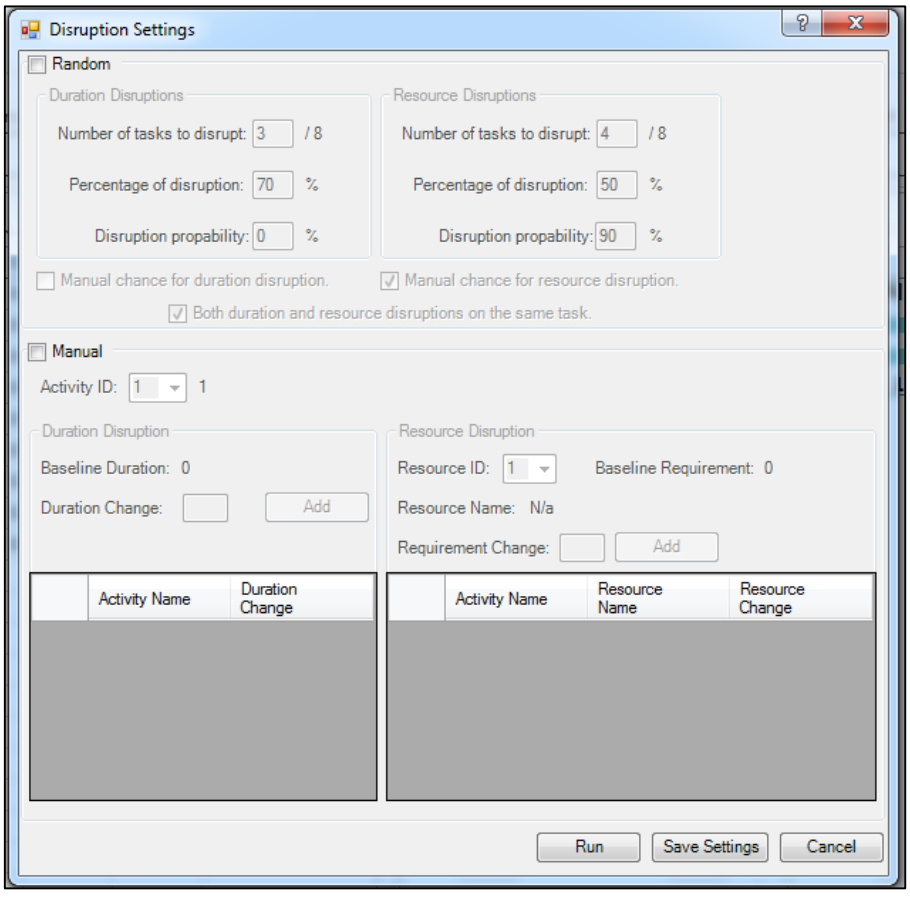
Εικόνα 10: Ανανέωση του αρχικού χρονοπρογράμματος.

7.4 Διαφοροποιήσεις

Η ομάδα διαφοροποιήσεων περιλαμβάνει δύο κουμπιά, το “Disruption Settings” και το “Run Random Disruptions”. Το πρώτο κουμπί (Εικόνα 11) εμφανίζει το παράθυρο ρυθμίσεων των διαφοροποιήσεων (Εικόνα 12).

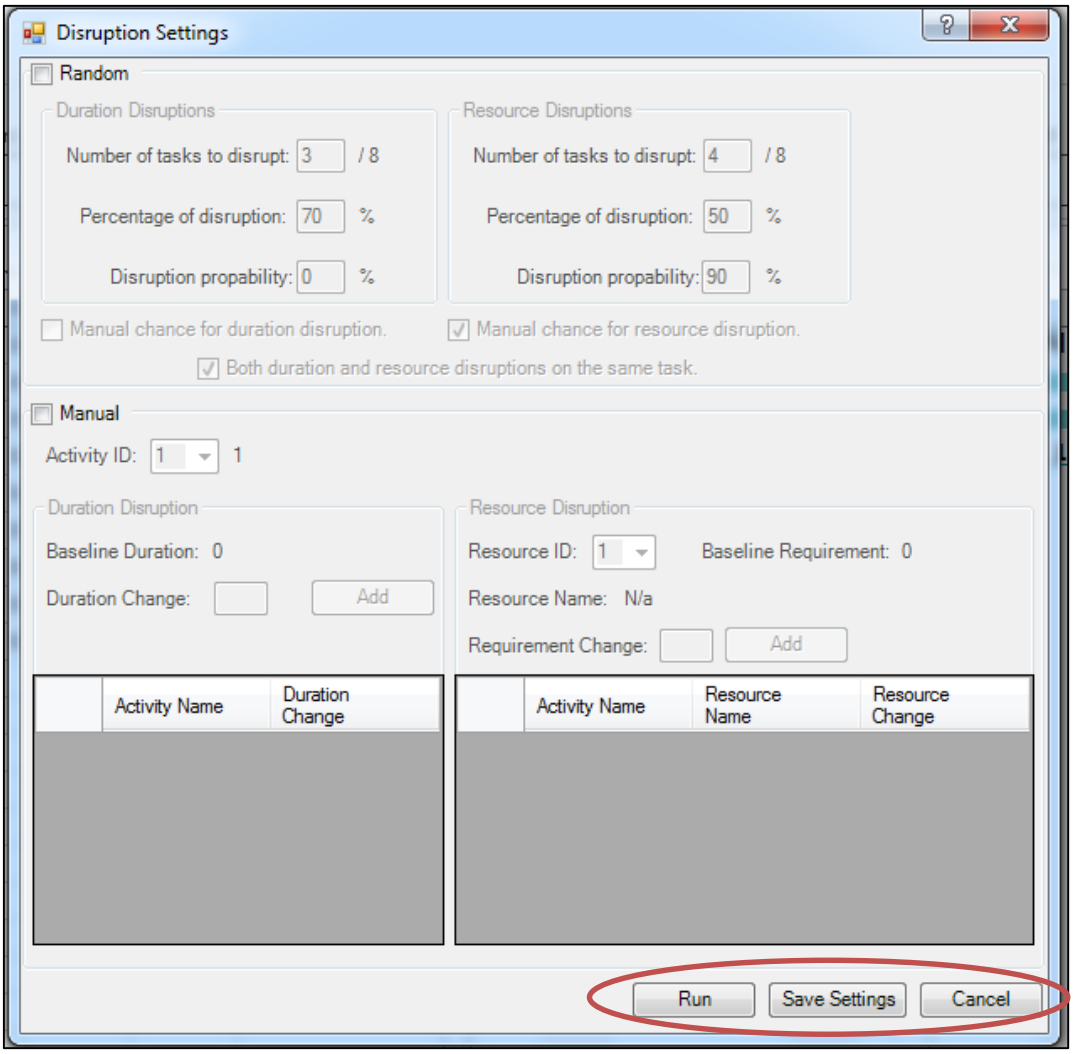


Εικόνα 11: Κουμπί ρυθμίσεων διαφοροποιήσεων.



Εικόνα 12: Παράθυρο ρυθμίσεων διαφοροποιήσεων.

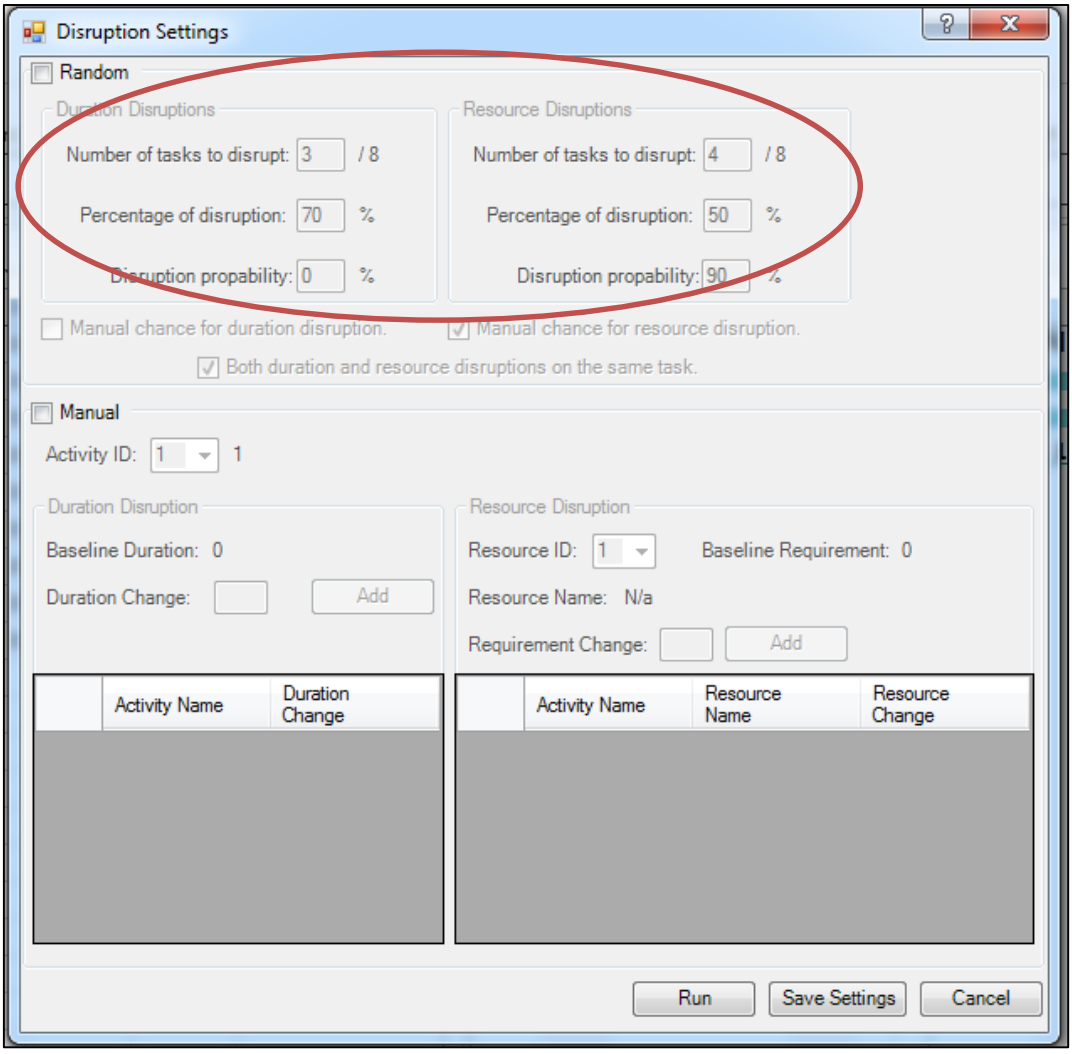
Το παράθυρο των διαφοροποιήσεων χωρίζεται σε δύο μέρη: το “Random”, που περιέχει τις ρυθμίσεις για τις τυχαίες διαφοροποιήσεις, και το “Manual”, που περιέχει τις ρυθμίσεις για τις χειροκίνητες διαφοροποιήσεις. Ο χρήστης μπορεί να επιλέξει τι είδους διαφοροποιήσεις θα εκτελέσει το add-in για το συγκεκριμένο έργο, οι οποίες μπορεί να είναι είτε τυχαίες είτε χειροκίνητες έιτε και τα δύο. Στο κάτω μέρος του παραθύρου υπάρχουν τρια κουμπιά: “Run”, “Save Settings” και “Cancel” (Εικόνα 13). Το κουμπί “Run” εκτελεί ό,τι ρυθμίσεις έχει εισάγει ο χρήστης, ενώ το “Save Settings” αποθηκεύει τις ρυθμίσεις για τις τυχαίες διαφοροποιήσεις έτσι ώστε ο χρήστης να μπορέσει να τις επαναλάβει ξανά. Οι ρυθμίσεις αυτές αποθηκεύονται και όταν ο χρήστη πατήσει το κουμπί “Run”. Το κουμπί “Cancel” κλείνει το παράθυρο χωρίς καμία αλλαγή στις ρυθμίσεις και χωρίς να εκτελέσει τίποτα.



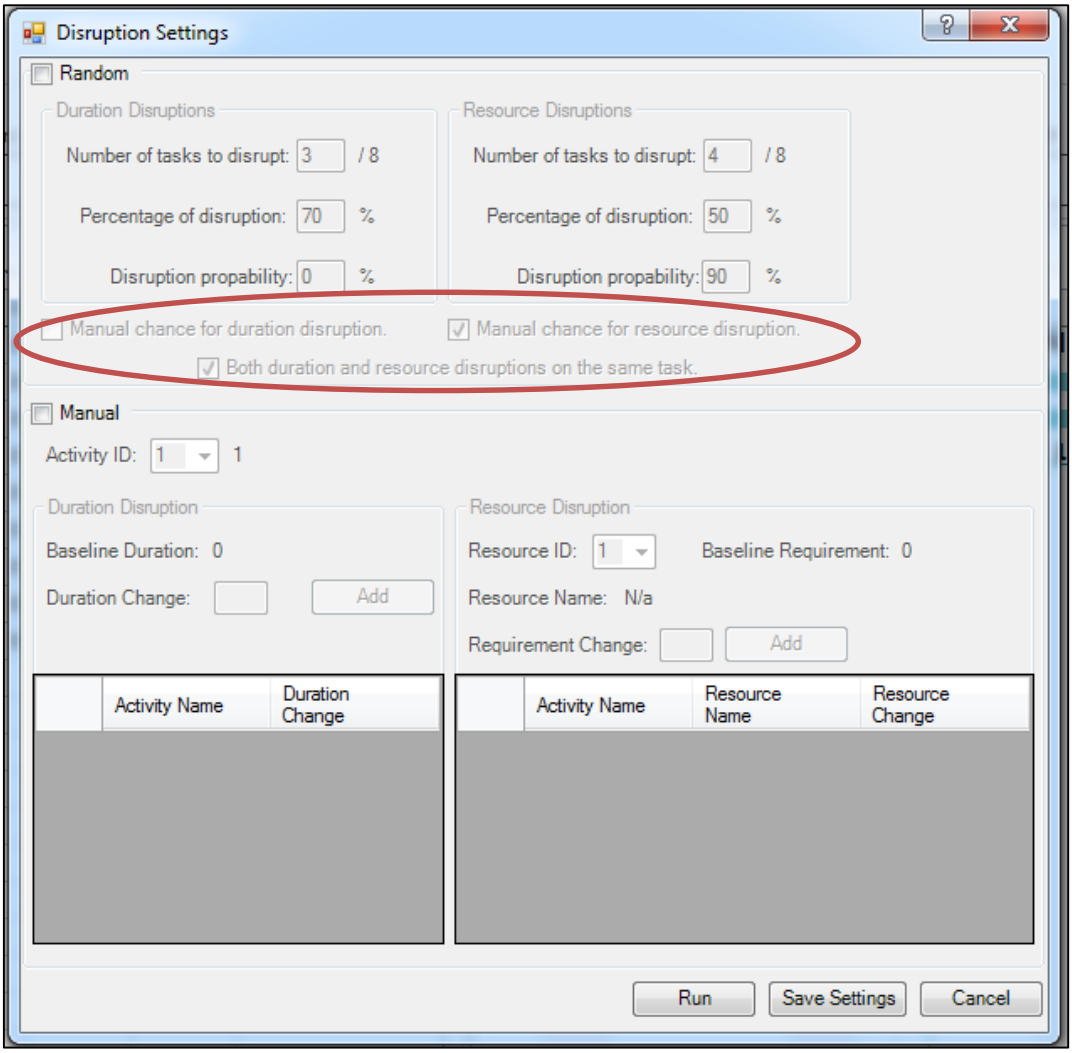
Εικόνα 13: Κουμπιά ρυθμίσεων διαφοροποιήσεων.

Το τμήμα των τυχαίων διαφοροποιήσεων χωρίζεται επίσης σε δύο μέρη: οι ρυθμίσεις για αλλαγές στην διάρκεια και οι ρυθμίσεις για αλλαγές στους πόρους (Εικόνα 14). Για κάθε διαφορετικό μέρος ο χρήστης μπορεί να επηρεάσει τρεις διαφορετικές ρυθμίσεις. Αρχικά επιλέγει τον αριθμό των δραστηριοτήτων που θα επηρεαστούν από τις διαφοροποιήσεις. Στην συνέχεια επιλέγει το ποσοστό κατά το οποίο θα διαφοροποιηθεί η δραστηριότητα, κατά πόσο δηλαδή θα αλλάξει η τιμή της διάρκειας ή του πόρου. Επιπλέον, αν ο χρήστης επιλέξει από τις ρυθμίσεις που υπάρχουν στο κάτω μέρος των ρυθμίσεων (Εικόνα 15) να

έχει χειροκίνητη πιθανότητα για να διαφοροποιηθεί η δραστηριότητα, μπορεί να ρυθμίσει τη πιθανότητα η δραστηριότητα να διαφοροποιηθεί. Διαφορετικά η πιθανότητα να διαφοροποιηθεί μια δραστηριότητα είναι ίση με ένα προς τον αριθμό των δραστηριοτήτων του έργου. Τέλος, ο χρήστης μπορεί να επιλέξει αν οι δραστηριότητες μπορούν να διαφοροποιηθούν στην διάρκειά τους και στους πόρους τους, αλλίως η κάθε δραστηριότητα μπορεί να διαφοροποιηθεί μόνο μια φορά, στην διάρκεια ή στους πόρους.

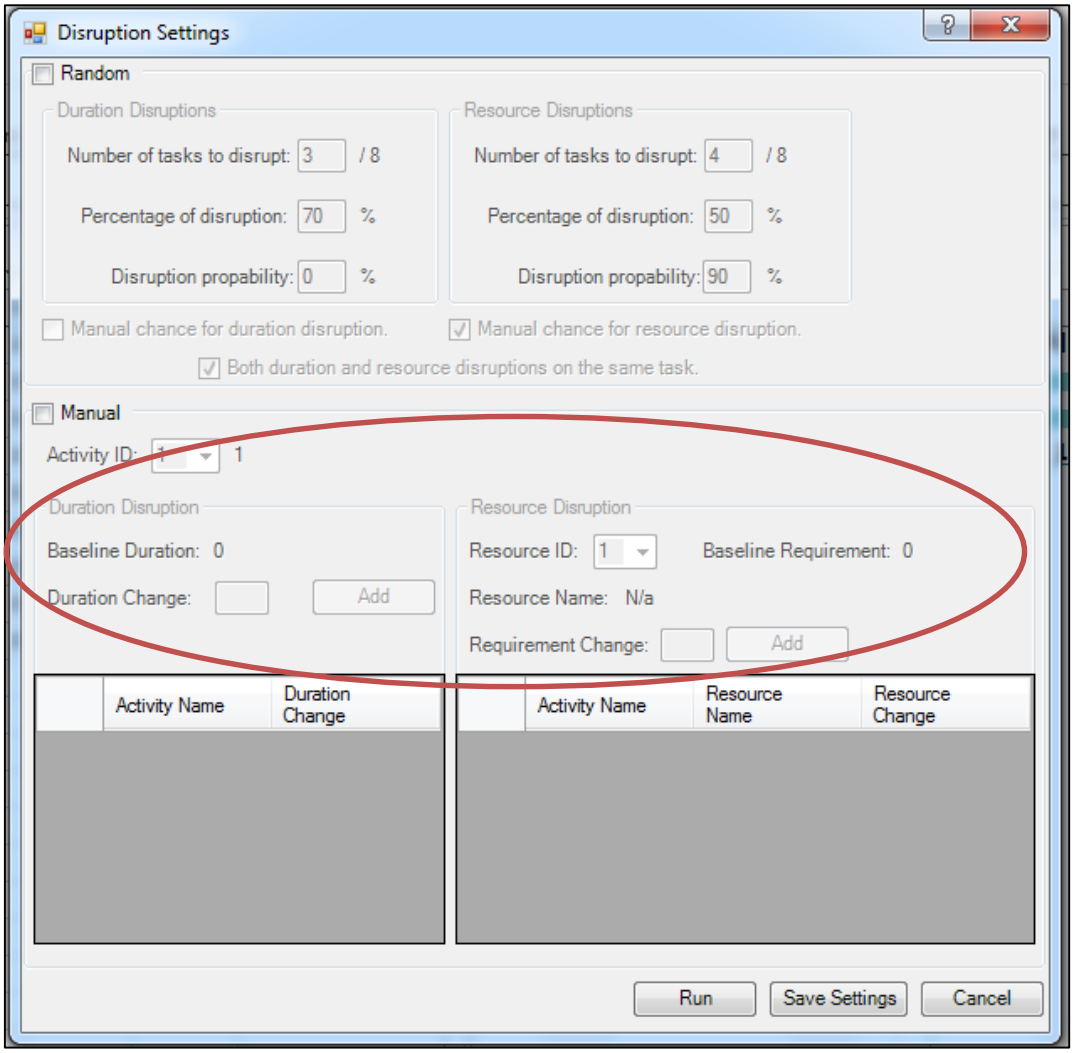


Εικόνα 14: Ρυθμίσεις τυχαίων διαφοροποιήσεων.

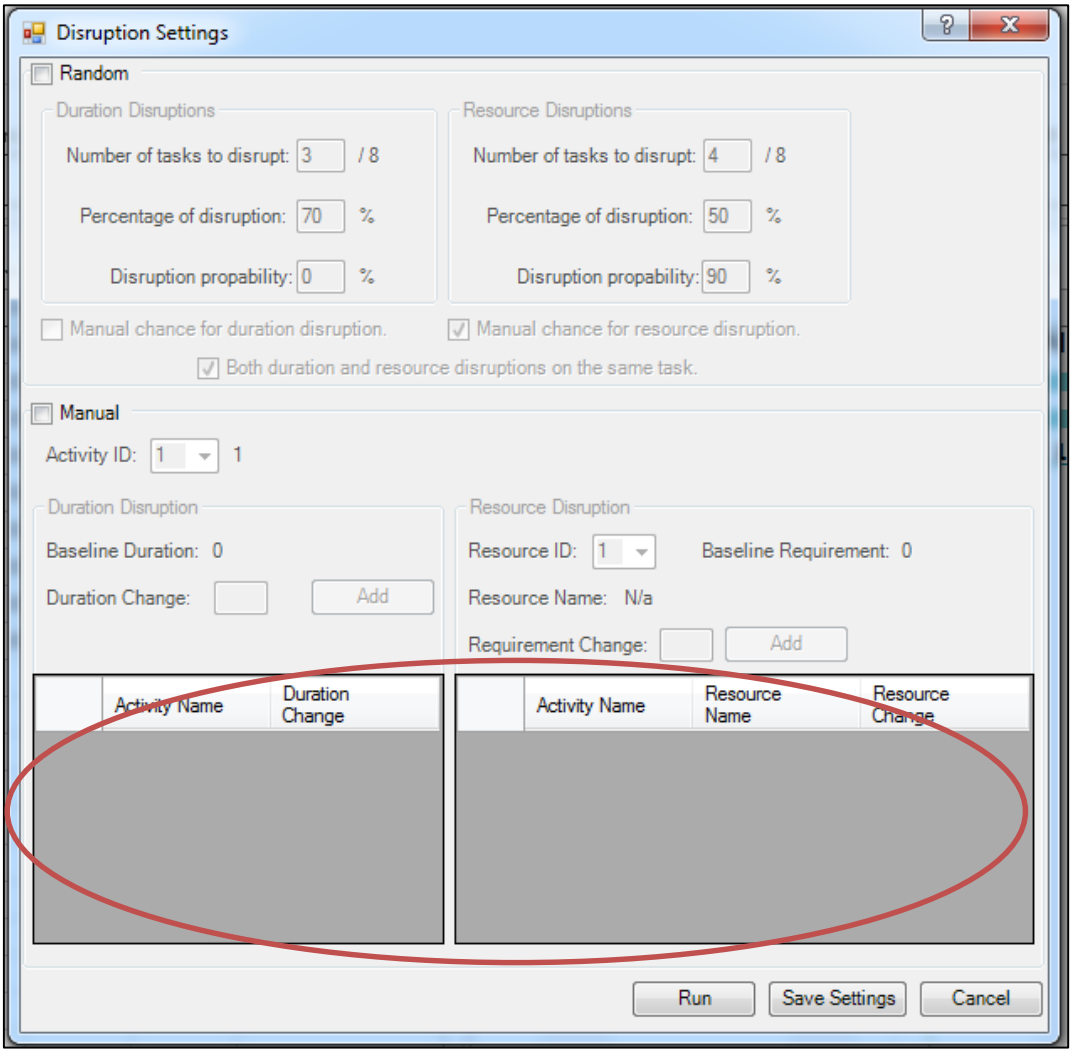


Εικόνα 15: Περεταίρω ρυθμίσεις για τις τυχαίες διαφοροποιήσεις.

Στο τμήμα των χειροκίνητων διαφοροποιήσεων (Εικόνα 16) ο χρήστης επιλέγει από ένα dropdown box την δραστηριότητα που επιθυμεί να διαταράξει. Στην συνέχεια, παρατηρώντας τα αρχικά δεδομένα για την δραστηριότητα, μπορεί να αλλάξει την διάρκεια ή την απαίτηση σε κάποιον πόρο εισάγοντας τον νέο αριθμό για αυτές τις μεταβλητές και μετά πατώντας το κουμπί “Add” δίπλα από το νέο αριθμό να συγκρατήσει αυτή την αλλαγή στους πίνακες που είναι πιο κάτω (Εικόνα 17). Στους πίνακες αυτούς βλέπει τις αλλαγές που έχει ήδη εισάγει ο χρήστης και μπορεί είτε να τις αλλάξει είτε να τις διαγράψει.

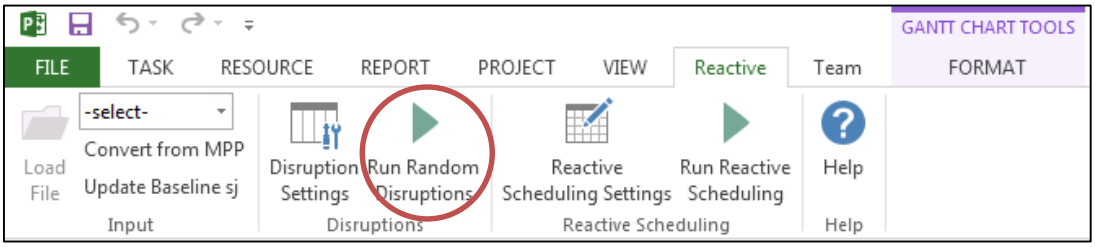


Εικόνα 16: Ρυθμίσεις χειροκίνητων διαφοροποιήσεων.



Εικόνα 17: Πίνακες χειροκίνητων διαφοροποιήσεων.

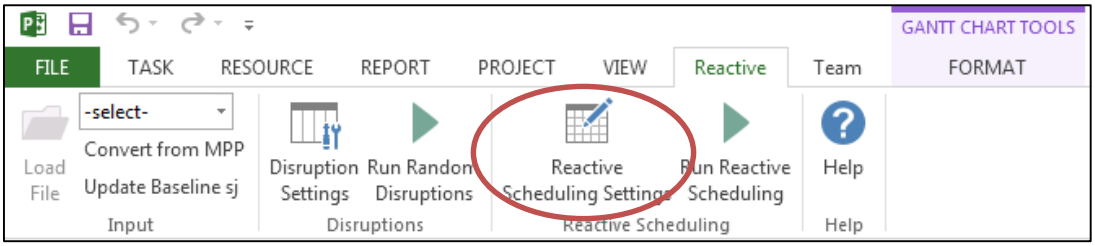
Το κουμπί “Run Random Disruptions” (Εικόνα 18) εκτελεί τις ρυθμίσεις του είχε εισάγει ο χρήστης προηγουμένως σε οποιοδήποτε έργο που είναι ενεργό εκείνη τη στιγμή.



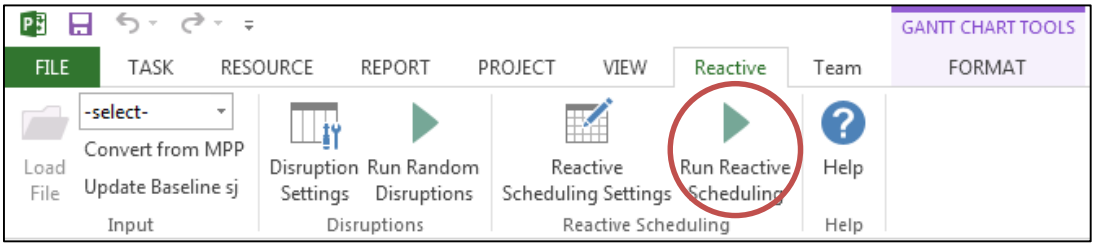
Εικόνα 18: Κουμπί εκτέλεσης τυχαίων διαφοροποιήσεων.

7.5 Αναδραστική μέθοδος

Η ομάδα της αναδραστικής μεθόδου περιέχει όπως και η ομάδα διαφοροποιήσεων δύο κουμπιά: το “Reactive Scheduling Settings” (Εικόνα 19), που ανοίγει το παράθυρο ρυθμίσεων αναδραστικής μεθόδου, και το “Run Reactive Scheduling” (Εικόνα 20), που εκτελεί την αναδραστική μέθοδο που έχει ρυθμίσει ο χρήστης.



Εικόνα 19: Κουμπί ρυθμίσεων αναδραστικής μεθόδου.

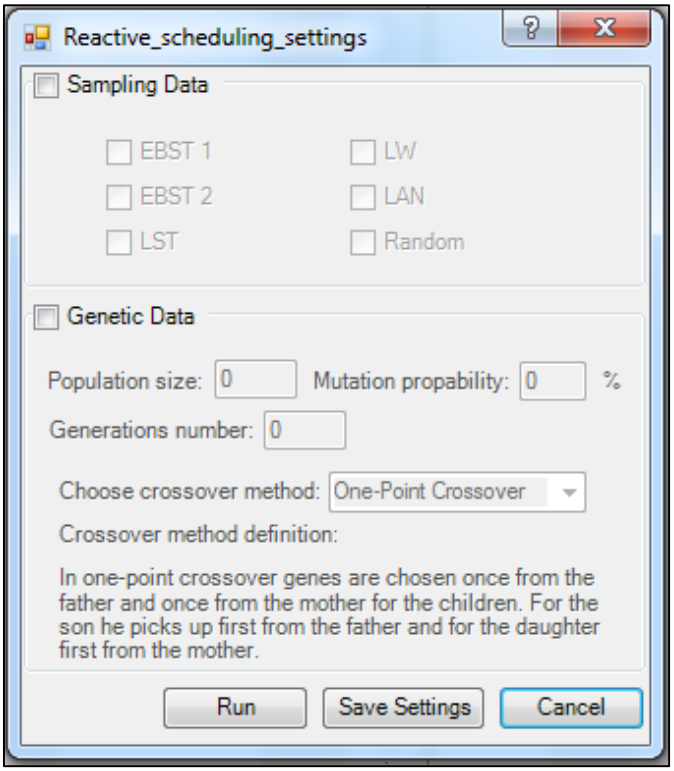


Εικόνα 20: Κουμπί εκτέλεσης αναδραστικής μεθόδου.

Στο παράθυρο αναδραστικής μεθόδου () ο χρήστης έχει την επιλογή μεταξύ δύο διαφορετικών τρόπων εκτέλεσης της αναδραστικής μεθόδου: “Sampling Data” και “Genetic Data”.

Στον τρόπο “Sampling Data”, ή αλλιώς τρόπος της δειγματοληψίας, ο χρήστης επιλέγει από μια ομάδα κανόνων προτεραιότητας της ευρετικής μεθόδου του σειριακού αλγορίθμου. Το add-in επιλύει κάθε έναν από τους κανόνες προτεραιότητας και τους συγκρίνει μεταξύ τους για να βρει τον καλύτερο.

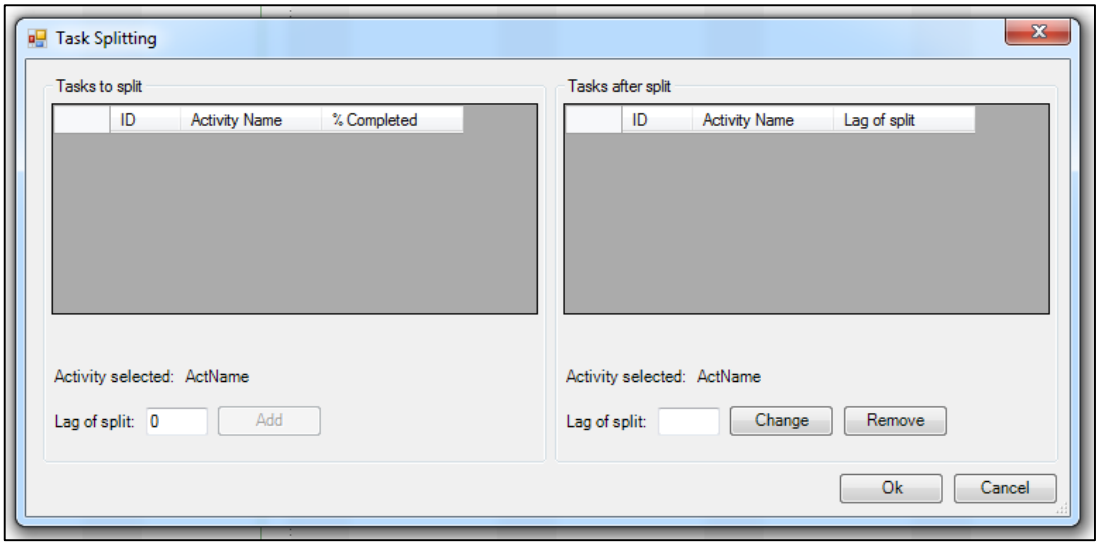
Στον τρόπο “Genetic Data”, που αποτελεί τον γενετικό τρόπο επίλυσης του προβλήματος, ο χρήστης εισάγει τα δεδομένα που χρειάζεται ο αλγόριθμος. Εισάγει τον πληθυσμό, τις γενεές και την πιθανότητα μετάλλαξης του γενετικού αλγορίθμου, καθώς και την μέθοδο διασταύρωσης των χρωμοσωμάτων. Επίσης, δίνεται και μια εξήγηση για τον τρόπο λειτουργίας για τις διαφορετικές μεθόδους διασταύρωσης που μπορεί να επιλέξει ο χρήστης.



Εικόνα 21: Παράθυρο αναδραστικής μεθόδου.

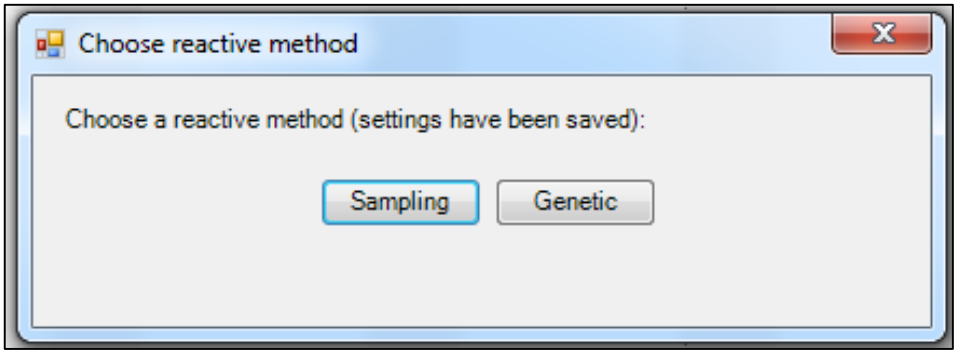
Όπως και με τις ρυθμίσεις διαφοροποιήσεων, έτσι και στις ρυθμίσεις αναδραστικής μεθόδου υπάρχουν τρία κουμπιά στο κάτω μέρος του παραθύρου, που αντιστοιχούν στην εκτέλεση των προεπιλεγμένων ρυθμίσεων, στην αποθήκευσή τους και στο κλείσιμο του παραθύρου χωρίς αποθήκευση των ρυθμίσεων.

Όταν επιλεχθεί να τρεχθεί η αναδραστική μέθοδος, εμφανίζεται το παράθυρο διαχωρισμού των δραστηριοτήτων (Εικόνα 22). Αυτό το παράθυρο χωρίζεται σε δύο τμήματα: πριν και μετά τον διαχωρισμό. Στο αριστερό τμήμα βρίσκονται οι δραστηριότητες που είναι ενεργές την χρονική στιγμή που τρέχουμε την μέθοδο. Όλες οι δραστηριότητες αυτές πρέπει να επιλεχθούν και να τους δωθεί ένα ***LAG***. Αφού πατήσουμε το κουμπί “ADD” η δραστηριότητα μεταφέρεται στο δεξί τμήμα. Αν δεν έχουν μείνει άλλες δραστηριότητες στα αριστερά, πατώντας το κουμπί “Ok” τρέχουμε την αναδραστική μέθοδο.



Εικόνα 22: Παράθυρο διαχωρισμού δραστηριοτήτων.

Τέλος, όταν πατηθεί το κουμπί εκτέλεσης αναδραστικής μεθόδου (Εικόνα 20) αρχικά εμφανίζεται το παράθυρο διαχωρισμού δραστηριοτήτων και στην συνέχεια εμφανίζεται το παράθυρο επιλογής τρόπου (Εικόνα 23), στο οποίο ο χρήστης επιλέγει με ποιόν τρόπο θα εκτελεσθεί η αναδραστική μέθοδος.

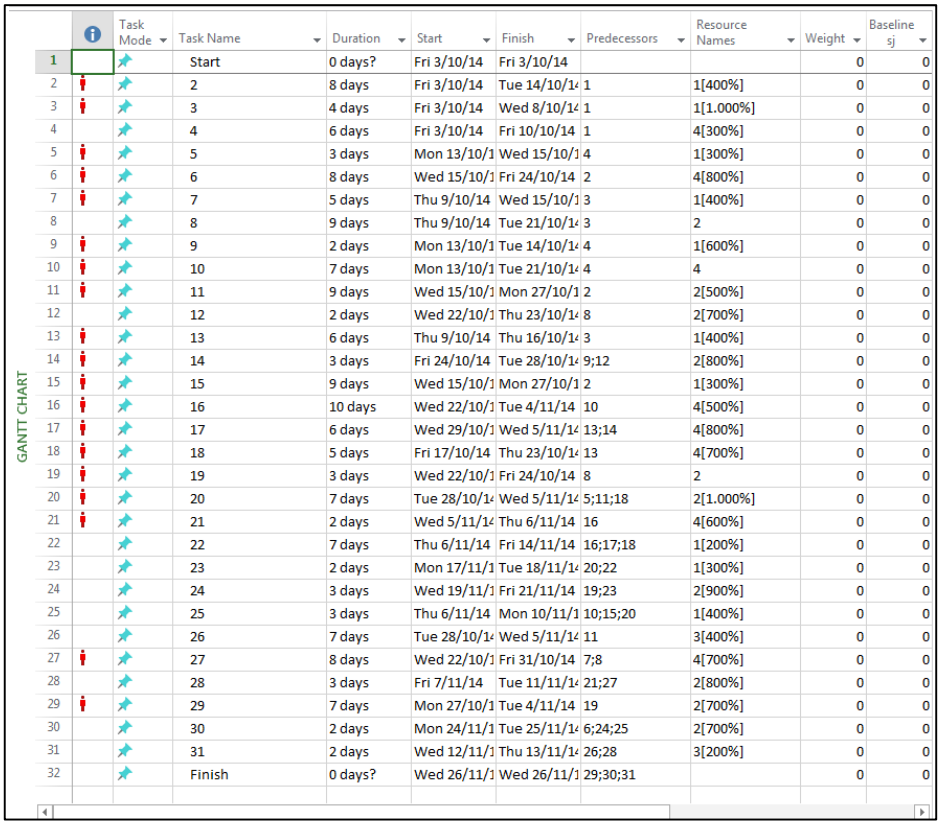


Εικόνα 23: Παράθυρο επιλογής τρόπου αναδραστικής μεθόδου.

7.6 Μελέτη περίπτωσης

Για καλύτερη κατανόηση της διεπαφής παρουσιάζεται η διαδικασία με την οποία προέκυψαν τα αριθμητικά αποτελέσματα για το έργο J301_1. Το αρχείο του έργου J301_1 της PSPLIB έχει την μορφή που φαίνεται στο Παράρτημα 6: Ομάδα έργων J301 σε μορφή RCP αρχείου. Αυτό το αρχείο δεν είναι δυνατόν να τον ανοίξει το MS Project και για αυτό χρησιμοποιήθηκε η εισαγωγή δεδομένων του addin.

Επιλέχτηκε συνεπώς από το dropdown box η μορφή αρχείου RCP και στην συνέχεια από το παράθυρο ανοίγματος αρχείου (Εικόνα 7) επιλέχτηκε το αρχείο για το έργο J301_1. Αφού ολοκληρώθηκε η διαδικασία εισαγωγής των δεδομένων προέκυψε ένα αρχείο τύπου mpp που έχει την μορφή της εικόνας Εικόνα 24.



GANTT CHART

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Weight	Baseline sj
1		Start	0 days?	Fri 3/10/14	Fri 3/10/14			0	0
2		2	8 days	Fri 3/10/14	Tue 14/10/14	1	1[400%]	0	0
3		3	4 days	Fri 3/10/14	Wed 8/10/14	1	1[1.000%]	0	0
4		4	6 days	Fri 3/10/14	Fri 10/10/14	1	4[300%]	0	0
5		5	3 days	Mon 13/10/1	Wed 15/10/1	4	1[300%]	0	0
6		6	8 days	Wed 15/10/1	Fri 24/10/14	2	4[800%]	0	0
7		7	5 days	Thu 9/10/14	Wed 15/10/1	3	1[400%]	0	0
8		8	9 days	Thu 9/10/14	Tue 21/10/14	3	2	0	0
9		9	2 days	Mon 13/10/1	Tue 14/10/14	4	1[600%]	0	0
10		10	7 days	Mon 13/10/1	Tue 21/10/14	4	4	0	0
11		11	9 days	Wed 15/10/1	Mon 27/10/1	2	2[500%]	0	0
12		12	2 days	Wed 22/10/1	Thu 23/10/14	8	2[700%]	0	0
13		13	6 days	Thu 9/10/14	Thu 16/10/14	3	1[400%]	0	0
14		14	3 days	Fri 24/10/14	Tue 28/10/14	9;12	2[800%]	0	0
15		15	9 days	Wed 15/10/1	Mon 27/10/1	2	1[300%]	0	0
16		16	10 days	Wed 22/10/1	Tue 4/11/14	10	4[500%]	0	0
17		17	6 days	Wed 29/10/1	Wed 5/11/14	13;14	4[800%]	0	0
18		18	5 days	Fri 17/10/14	Thu 23/10/14	13	4[700%]	0	0
19		19	3 days	Wed 22/10/1	Fri 24/10/14	8	2	0	0
20		20	7 days	Tue 28/10/14	Wed 5/11/14	5;11;18	2[1.000%]	0	0
21		21	2 days	Wed 5/11/14	Thu 6/11/14	16	4[600%]	0	0
22		22	7 days	Thu 6/11/14	Fri 14/11/14	16;17;18	1[200%]	0	0
23		23	2 days	Mon 17/11/1	Tue 18/11/14	20;22	1[300%]	0	0
24		24	3 days	Wed 19/11/1	Fri 21/11/14	19;23	2[900%]	0	0
25		25	3 days	Thu 6/11/14	Mon 10/11/1	10;15;20	1[400%]	0	0
26		26	7 days	Tue 28/10/14	Wed 5/11/14	11	3[400%]	0	0
27		27	8 days	Wed 22/10/1	Fri 31/10/14	7;8	4[700%]	0	0
28		28	3 days	Fri 7/11/14	Tue 11/11/14	21;27	2[800%]	0	0
29		29	7 days	Mon 27/10/1	Tue 4/11/14	19	2[700%]	0	0
30		30	2 days	Mon 24/11/1	Tue 25/11/14	6;24;25	2[700%]	0	0
31		31	2 days	Wed 12/11/1	Thu 13/11/14	26;28	3[200%]	0	0
32		Finish	0 days?	Wed 26/11/1	Wed 26/11/1	29;30;31		0	0

Εικόνα 24: Άνοιγμα αρχείου J301_1

Το αρχείο που προέκυψε δεν είναι χρονοπρογραμματισμένο εφικτώς και άρα δεν μπορούμε να γνωρίζουμε τους αρχικούς χρόνους εκκίνησης κάθε δραστηριότητας.Για αυτόν τον λόγο, ρυθμίζονται οι χρονικές παραμέτρους του έργου με αρχή του έργου της 2/6/14 και με ημερομηνία κατάστασης του έργου της 29/6/14. έτσι ώστε να μπορεί να χρονοπρογραμματιστεί το έργο και να βρίσκεται στη χρονική στιγμή 20.

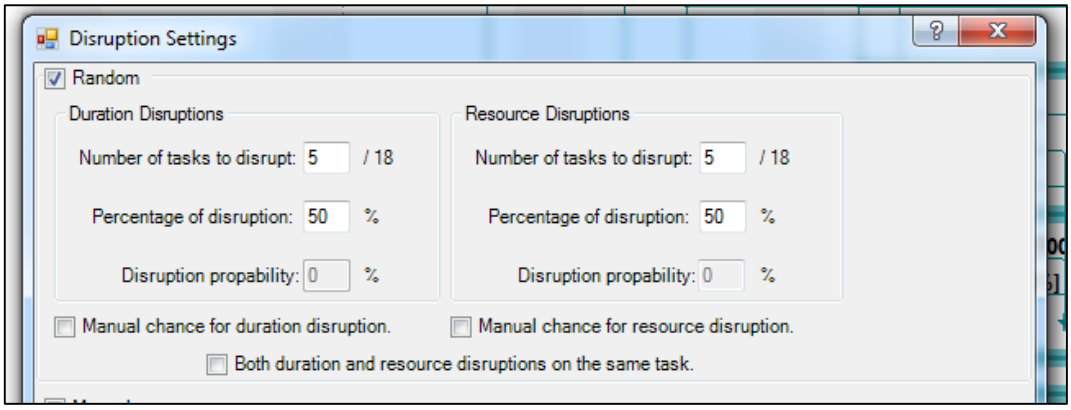
Το MS Project έχει την δυνατότητα να χρονοπρογραμματίσει ένα έργο που του έχει εισαχθεί έτσι ώστε να τηρούνται οι περιορισμοί στους πόρους. Αυτό γίνεται με την χρήση του κουμπιού Leveling όπου μετακινούνται οι δραστηριότητες έτσι ώστε να ομαλοποιηθεί η χρήση των πόρων στο έργο. Αφού προγραμματιστούν οι δραστηριότητες, χρησιμοποιήθηκε το κουμπί της Εικόνα 10 για να βρεθούν οι τιμές των αρχικών χρόνων εκκίνησης των δραστηριοτήτων. Επίσης, ανανεώθηκε το έργο για την ημερομηνία 29/6/14 έτσι ώστε να βρεθούν ποιες δραστηριότητες έχουν ολοκληρωθεί και ποιες είναι υπό εκτέλεση. Έτσι καταλήξαμε σε ένα έργο της μορφής της Εικόνα 25.

GANTT CHART

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Weight	Baseline sj
1	✓	Start	0 days?	Mon 2/6/14	Mon 2/6/14			0	0
2	✓	2	8 days	Fri 6/6/14	Tue 17/6/14	1	1[400%]	0	4
3	✓	3	4 days	Mon 2/6/14	Thu 5/6/14	1	1[1.000%]	0	0
4	✓	4	6 days	Mon 2/6/14	Mon 9/6/14	1	4[300%]	0	0
5	✓	5	3 days	Thu 12/6/14	Mon 16/6/14	4	1[300%]	0	8
6		6	8 days	Fri 25/7/14	Tue 5/8/14	2	4[800%]	0	39
7	✓	7	5 days	Tue 17/6/14	Mon 23/6/14	3	1[400%]	0	11
8	✓	8	9 days	Fri 6/6/14	Wed 18/6/14	3	2	0	4
9	✓	9	2 days	Tue 10/6/14	Wed 11/6/14	4	1[600%]	0	6
10	✓	10	7 days	Tue 10/6/14	Wed 18/6/14	4	4	0	6
11		11	9 days	Wed 18/6/14	Mon 30/6/14	2	2[500%]	0	12
12	✓	12	2 days	Thu 19/6/14	Fri 20/6/14	8	2[700%]	0	13
13	✓	13	6 days	Thu 12/6/14	Thu 19/6/14	3	1[400%]	0	8
14	✓	14	3 days	Mon 23/6/14	Wed 25/6/14	9;12	2[800%]	0	15
15		15	9 days	Wed 18/6/14	Mon 30/6/14	2	1[300%]	0	12
16		16	10 days	Thu 19/6/14	Wed 2/7/14	10	4[500%]	0	13
17		17	6 days	Thu 3/7/14	Thu 10/7/14	13;14	4[800%]	0	23
18	✓	18	5 days	Fri 20/6/14	Thu 26/6/14	13	4[700%]	0	14
19		19	3 days	Thu 26/6/14	Mon 30/6/14	8	2	0	18
20		20	7 days	Tue 1/7/14	Wed 9/7/14	5;11;18	2[1.000%]	0	21
21		21	2 days	Wed 23/7/14	Thu 24/7/14	16	4[600%]	0	37
22		22	7 days	Fri 11/7/14	Mon 21/7/14	16;17;18	1[200%]	0	29
23		23	2 days	Tue 22/7/14	Wed 23/7/14	20;22	1[300%]	0	36
24		24	3 days	Thu 24/7/14	Mon 28/7/14	19;23	2[900%]	0	38
25		25	3 days	Thu 10/7/14	Mon 14/7/14	10;15;20	1[400%]	0	28
26		26	7 days	Tue 1/7/14	Wed 9/7/14	11	3[400%]	0	21
27		27	8 days	Fri 11/7/14	Tue 22/7/14	7;8	4[700%]	0	29
28		28	3 days	Tue 29/7/14	Thu 31/7/14	21;27	2[800%]	0	41
29		29	7 days	Thu 10/7/14	Fri 18/7/14	19	2[700%]	0	28
30		30	2 days	Wed 6/8/14	Thu 7/8/14	6;24;25	2[700%]	0	47
31		31	2 days	Fri 1/8/14	Mon 4/8/14	26;28	3[200%]	0	44
32		Finish	0 days?	Fri 8/8/14	Fri 8/8/14	29;30;31		0	48

Εικόνα 25: Χρονοπρογραμματισμένο J301_1

Σε αυτό το χρονοπρογραμματισμένο έργο προκαλούνται διαφοροποιήσεις στις μη ολοκληρώμενες δραστηριότητες. Στο παράθυρο ρυθμίσεως διαφοροποιήσεων εισάγονται οι παράμετροι των διαφοροποιήσεων που θα προκληθούν (Εικόνα 26). Επιλέγονται μόνο τυχαίες διαφοροποιήσεις οι οποίες δεν έχουν χειροκίνητες πιθανότητες να συμβούν και μπορεί να συμβεί μόνο ενός είδους διαφοροποίηση σε κάθε δραστηριότητα. Και για τα δύο είδη διαφοροποιήσεων επιλέγονται 5 δραστηριότητες να διαφοροποιηθούν και η διαφοροποίηση είναι της τάξεως του 50%.



Εικόνα 26: Παράμετροι διαφοροποιήσεων J301_1

Αφού γίνουν οι τυχαίες διαφοροποιήσεις με τις παραμέτρους που εισάχθηκαν, εμφανίζεται ένα νέο έργο που έχει την μορφή της Εικόνα 27. Αυτό το έργο δεν είναι πλέον εφικτό, αφού οι περιορισμοί στους πόρους δεν τηρούνται πλέον. Πάνω σε αυτό το νέο έργο θα λειτουργήσει ο γενετικός αλγόριθμος.

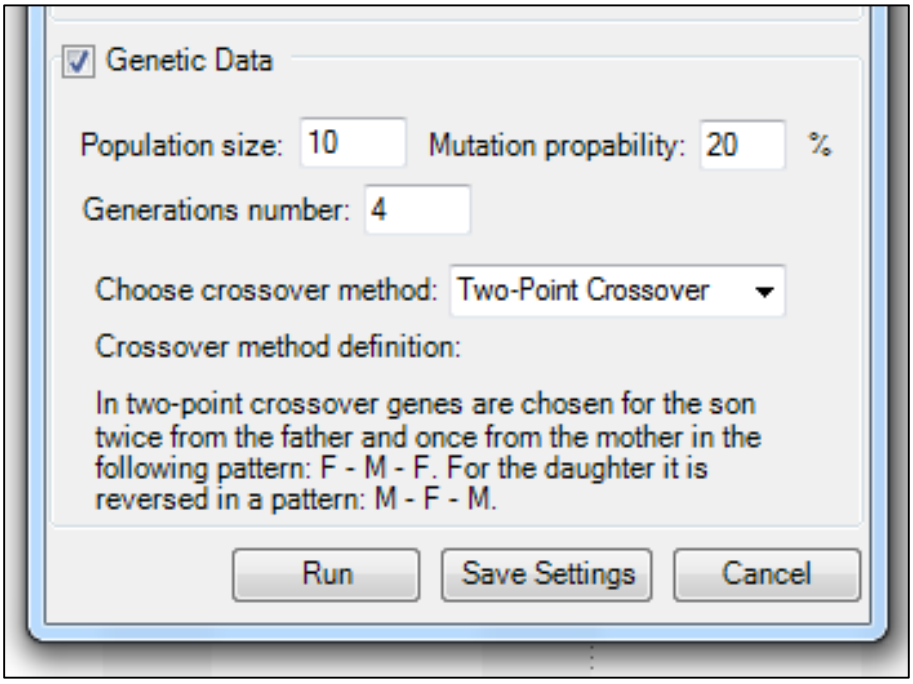
GANTT CHART

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Weight	Baseline sj
1		Start	0 days?	Mon 2/6/14	Mon 2/6/14			0	0
2		2	8 days	Fri 6/6/14	Tue 17/6/14	1	1[400%]	0	4
3		3	4 days	Mon 2/6/14	Thu 5/6/14	1	1[1.000%]	0	0
4		4	6 days	Mon 2/6/14	Mon 9/6/14	1	4[300%]	0	0
5		5	3 days	Thu 12/6/14	Mon 16/6/14	4	1[300%]	0	8
6		6	12 days	Fri 25/7/14	Mon 11/8/14	2	4[800%]	0	39
7		7	5 days	Tue 17/6/14	Mon 23/6/14	3	1[400%]	0	11
8		8	9 days	Fri 6/6/14	Wed 18/6/14	3	2	0	4
9		9	2 days	Tue 10/6/14	Wed 11/6/14	4	1[600%]	0	6
10		10	7 days	Tue 10/6/14	Wed 18/6/14	4	4	0	6
11		11	14 days	Wed 18/6/14	Mon 7/7/14	2	2[500%]	0	12
12		12	2 days	Thu 19/6/14	Fri 20/6/14	8	2[700%]	0	13
13		13	6 days	Thu 12/6/14	Thu 19/6/14	3	1[400%]	0	8
14		14	3 days	Mon 23/6/14	Wed 25/6/14	9;12	2[800%]	0	15
15		15	14 days	Wed 18/6/14	Mon 7/7/14	2	1[300%]	0	12
16		16	15 days	Thu 19/6/14	Wed 9/7/14	10	4[500%]	0	13
17		17	9 days	Thu 3/7/14	Tue 15/7/14	13;14	4[800%]	0	23
18		18	5 days	Fri 20/6/14	Thu 26/6/14	13	4[700%]	0	14
19		19	3 days	Thu 26/6/14	Mon 30/6/14	8	2[200%]	0	18
20		20	7 days	Tue 1/7/14	Wed 9/7/14	5;11;18	2[1.300%]	0	21
21		21	2 days	Wed 23/7/14	Thu 24/7/14	16	4[900%]	0	37
22		22	7 days	Fri 11/7/14	Mon 21/7/14	16;17;18	1[300%]	0	29
23		23	2 days	Tue 22/7/14	Wed 23/7/14	20;22	1[300%]	0	36
24		24	3 days	Thu 24/7/14	Mon 28/7/14	19;23	2[1.300%]	0	38
25		25	3 days	Thu 10/7/14	Mon 14/7/14	10;15;20	1[400%]	0	28
26		26	7 days	Tue 1/7/14	Wed 9/7/14	11	3[400%]	0	21
27		27	8 days	Fri 11/7/14	Tue 22/7/14	7;8	4[700%]	0	29
28		28	3 days	Tue 29/7/14	Thu 31/7/14	21;27	2[800%]	0	41
29		29	7 days	Thu 10/7/14	Fri 18/7/14	19	2[700%]	0	28
30		30	2 days	Wed 6/8/14	Thu 7/8/14	6;24;25	2[700%]	0	47
31		31	2 days	Fri 1/8/14	Mon 4/8/14	26;28	3[200%]	0	44
32		Finish	0 days?	Fri 8/8/14	Fri 8/8/14	29;30;31		0	48

Εικόνα 27: Μη εφικτό χρονοπρόγραμμα J301_1

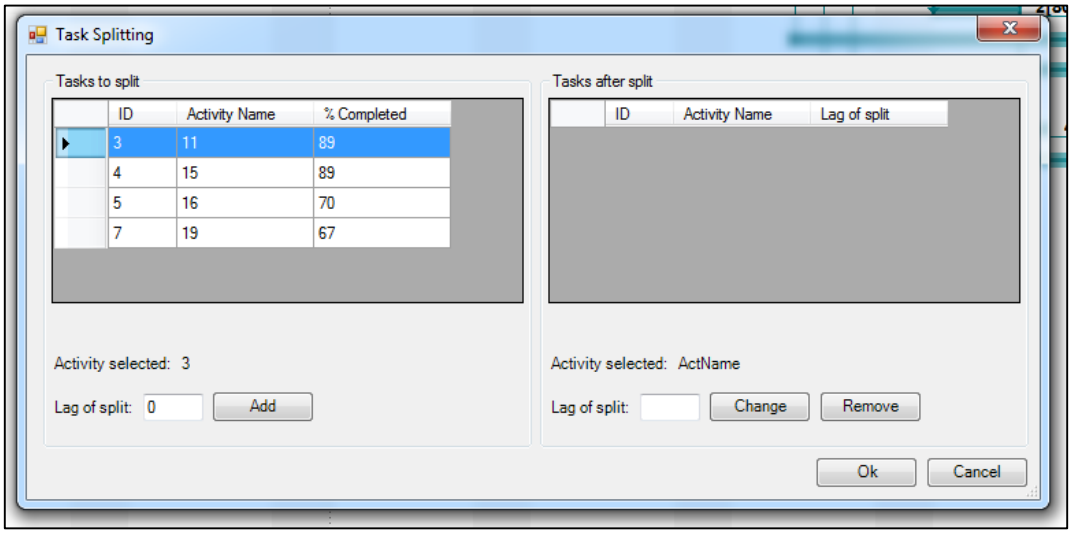
Ανοίγοντας το παράθυρο ρυθμίσεων αναδραστικής μεθόδου (Εικόνα 21) επιλέγεται η μέθοδος του γενετικού αλγορίθμου. Εισάγονται οι παράμετροι για τον πληθυσμό, τις γενιές, την πιθανότητα μετάλλαξης και την μέθοδο διασταύρωσης του αλγορίθμου (Εικόνα 28). Επιλέγεται πληθυσμός 10 χρωμοσωμάτων με πιθανότητα μετάλλαξης 20%. Η

διασταύρωση γίνεται με την μέθοδο των δύο σημείων, ενώ ο αλγόριθμος θα λειτουργήσει για 4 γενιές. Πατώντας το κουμπί “Run” ξεκινάει η αναδραστική μέθοδος.

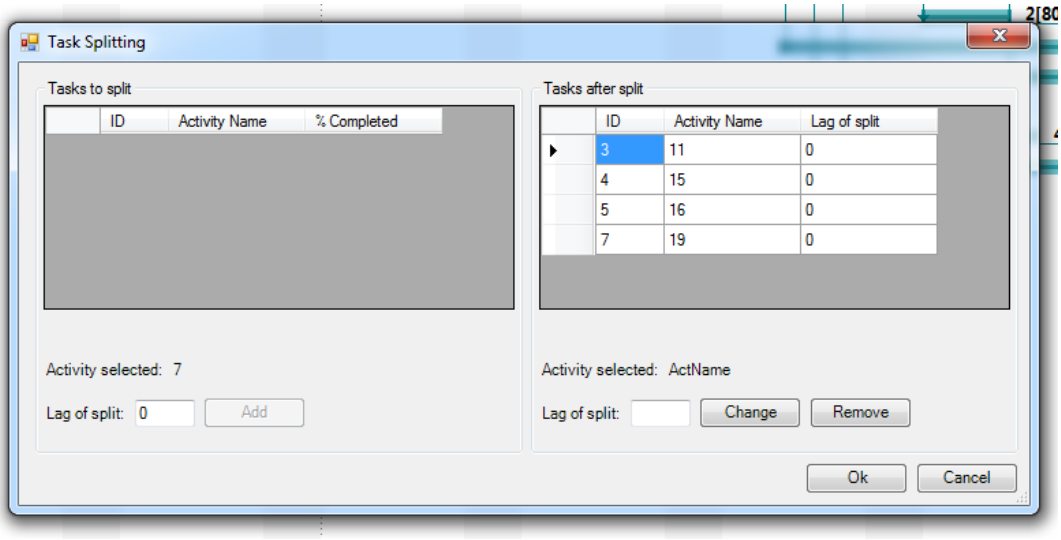


Εικόνα 28: Παράμετροι γενετικού αλγορίθμου.

Πριν την εκκίνηση του αλγορίθμου, απαιτείται ο χωρισμός των ημιολοκληρώμενων δραστηριοτήτων (Εικόνα 29). Υπάρχουν 4 δραστηριότητες οι οποίες είναι υπό εκτέλεση την χρονική στιγμή ***20*** και χρειάζεται να αποφασιστεί πως θα χωριστούν. Στο συγκεκριμένο πείραμα επιλέχτηκε να μην έχει καμία δραστηριότητα καθυστέρηση στον χωρισμό της, άρα να προγραμματιστούν όσο το νωρίτερο δυνατόν γίνεται (Εικόνα 30).

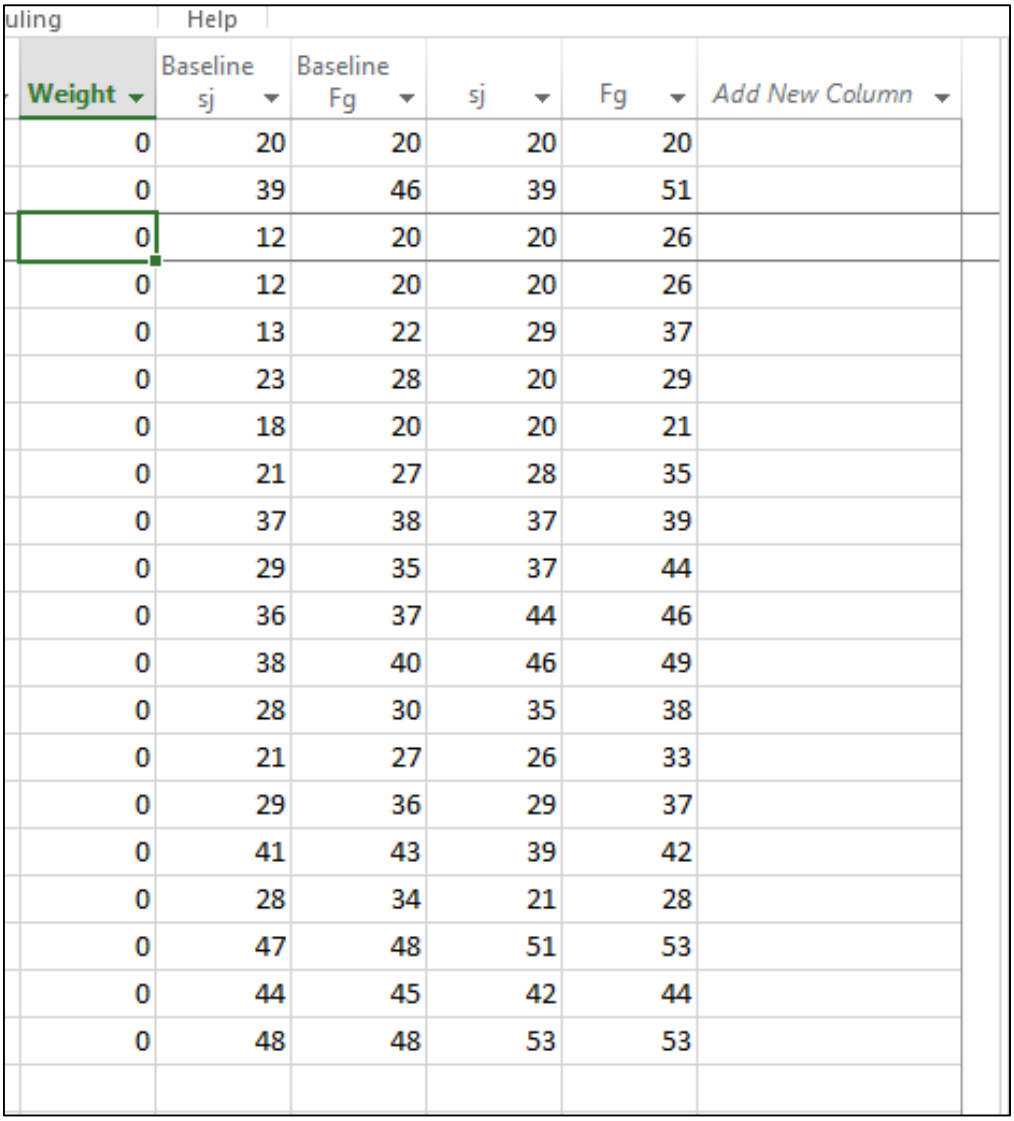


Εικόνα 29: Δραστηριότητες προς διαχωρισμό.



Εικόνα 30: Δραστηριότητες μετά τον διαχωρισμό.

Αφού επιλέχτηκε ο διαχωρισμός των δραστηριοτήτων άρχισε να λειτουργεί ο αλγόριθμος. Μετά το πέρας του, δημιουργήθηκε ένα νέο έργο στο οποίο έχουν αφαιρεθεί οι ολοκληρωμένες δραστηριότητες και έχουν μείνει μόνο οι υπόλοιπες. Για αυτές τις δραστηριότητες έχουν βρεθεί οι νέοι χρόνοι εκκίνησης τους και παρουσιάζονται δίπλα στους αρχικούς τους χρόνους. Επιπλέον, φαίνονται και οι χρόνοι λήξης τους και για το νέο και για το αρχικό χρονοπρόγραμμα, όπου παρατηρείται οι διαφορά στις διάρκειες που μπορεί να έχουν από τις διαφοροποιήσεις που προκλήθηκαν (Εικόνα 31).



uling Help

Weight	Baseline sj	Baseline Fg	sj	Fg	Add New Column
0	20	20	20	20	
0	39	46	39	51	
0	12	20	20	26	
0	12	20	20	26	
0	13	22	29	37	
0	23	28	20	29	
0	18	20	20	21	
0	21	27	28	35	
0	37	38	37	39	
0	29	35	37	44	
0	36	37	44	46	
0	38	40	46	49	
0	28	30	35	38	
0	21	27	26	33	
0	29	36	29	37	
0	41	43	39	42	
0	28	34	21	28	
0	47	48	51	53	
0	44	45	42	44	
0	48	48	53	53	

Εικόνα 31: Αποτελέσματα γενετικού αλγορίθμου.

8 Συμπέρασμα

8.1 Σύνοψη

Η αβεβαιότητα των έργων είναι ένα σημαντικό μειονέκτημα του προγραμματισμού τους. Εξαιτίας αυτής είναι αρκετά δύσκολο να μην υπάρχουν διαφοροποιήσεις μεταξύ του αρχικού χρονοπρογράμματος που κατασκευάζει ο διαχειριστής ενός έργου και του τελικού χρονοπρογράμματος με το οποίο εκτελείται το έργο. Αυτές οι διαφοροποιήσεις προκαλούν σημαντικά προβλήματα κατά την εκτέλεση του έργου και η επίλυσή τους έχει προβληματίσει πόλλους ερευνητές τα τελευταία χρόνια.

Παρουσιάστηκαν στην διάρκεια της εργασίας αναδραστικές μέθοδοι με τις οποίες μπορεί να αντιμετωπίσει ο διαχειριστής ενός έργου τυχόν διαφοροποιήσεις που μπορεί να συμβούν κατά την διάρκειά του. Οι μέθοδοι αυτές έχουν σαν στόχο την τροποποίηση του αρχικού χρονοπρογράμματος μετά από κάποια διαφοροποίηση έτσι ώστε να ξαναγίνει εφικτό και να μπορεί να συνεχίσει η εκτέλεση του έργου. Υπάρχουν διάφορες μέθοδοι που μπορεί να χρησιμοποιήσει ο διαχειριστής του έργου, όπως το σύστημα διακλάδωσης, του κάτω φράγματος, του σειριακού συστήματος παραγωγής χρονοπρογραμμάτος και του γενετικού αλγορίθμου. Από αυτές τις μεθόδους υλοποιήθηκε η μεταευρετική μέθοδος του γενετικού αλγορίθμου, που βασίζεται στην επιλογή της καλύτερης από μια ομάδα επιλύσεων, που προκύπτουν με βάση τους κανόνες της γενετικής επιστήμης.

Το πρόβλημα που ζητήθηκε να λυθεί περιέχει πολλαπλά είδη πόρων και τρόπων συσχέτισης δραστηριοτήτων. Όμως το πρόβλημα αυτό είναι εκθετικώς ανάλογο του μεγέθους του έργου και δύσκολα μπορεί να επιλυθεί. Για αυτό προτάθηκε μία μέθοδος μετασχηματισμού του σε ένα απλούστερο της μορφής RCPSP. Σε αυτή τη μορφή υπάρχουν μόνο ανανεώσιμοι πόροι για το έργο, διότι αυτοί οι πόροι είναι συνήθως σταθεροί κατά την διάρκεια του έργου και αλλαγές στις απαιτήσεις των δραστηριοτήτων προκαλούν μη εφικτά χρονοπρογράμματα σε αντίθεση με τους μη ανανεώσιμους που μπορεί να υπάρχουν σε αφθονία κατά την διάρκεια του έργου. Επιπλέον, διατηρήθηκε μόνο ο τρόπος συσχέτισης τέλους – αρχής (finish – start), αφού είναι ο συχνότερα χρησιμοποιούμενος τρόπος.

Ο γενετικός αλγόριθμος που υλοποιήθηκε λαμβάνει σαν στοιχεία ένα νέο πλασματικό έργο, που αποτελείται μόνο από τις δραστηριότητες του αρχικού έργου που δεν έχουν ολοκληρωθεί. Καταλήγει σε μία βέλτιστη λύση για το νέο χρονοπρόγραμμα, όπου δεν απέχει πολύ από το αρχικό και έχει χρόνο λήξης του έργου όσο πιο σύντομο γίνεται.

Ο αλγόριθμος υλοποιήθηκε στην γλώσσα C#. Δημιουργήθηκε και μια διεπαφή χρήστη – μηχανής στο περιβάλλον του MS Project για καλύτερη διαχείρηση των δεδομένων του έργου και πιο ευπαρουσίαστα αριθμητικά δεδομένα.

Από τα πειράματα που έγιναν στην ομάδα έργων J301 της PSPLIB παρατηρήθηκε ότι ο αλγόριθμος δίνει αποτελέσματα με μικρές διαφορές στους χρόνους τέλους του έργου και με μικρές διαφορές από το αρχικό χρονοπρόγραμμα με συνήθης εξαίρεση δραστηριότητες με μεγάλες ανάγκες σε πόρους, που κατά πάσα πιθανότητα ανήκουν στον κρίσιμο δρόμο του έργου και άρα οποιαδήποτε διαφοροποίηση θα προκαλούσε μεγάλες αλλαγές σε αυτές.

8.2 Περαιτέρω έρευνα

Για καλύτερη επαλήθευση του αλγορίθμου μπορεί σε μετέπειτα χρόνο γίνουν πειράματα σε άλλες ομάδες έργων της PSPLIB, καθώς και πειράματα σε πραγματικά έργα των οποίων υπάρχουν τα αρχεία MPP. Αυτή η περαιτέρω πειραματική διαδικασία μπορεί να οδηγήσει σε καλύτερη βελτιστοποίηση της παραμέτρου ***FIT_MULTIPLIER*** με περισσότερα δεκαδικά ψηφία.

Επιπλέον, μπορεί να εισαγθεί ένας αλγόριθμος επίλυσης προβλημάτων με μη-ανανεώσιμους πόρους ή με πολλαπλούς τρόπους εκτέλεσης των δραστηριοτήτων και να επαληθευτούν τα αποτελέσματα του γενετικού αλγορίθμου πάνω σε αυτά τα προβλήματα. Φυσικώς, ο γενετικός αλγόριθμος χρειάζεται να μετατραπεί σε μορφή που τα γονίδιά του να αντιστοιχούν σε παραμέτρους του νέου προβλήματος που τώρα πρέπει να επιλυθούν.

Τέλος, η διεπαφή χρήστη – μηχανής μπορεί να βελτιωθεί με την άμεση παρουσίαση των αποτελεσμάτων της αναδραστικής μεθόδους στο περιβάλλον του MS-PROJECT, αντί για απλά νούμερα σε μία στήλη του. Επίσης, μπορεί να διερευνήθουν άλλου είδους αρχεία που χρησιμοποιούνται για τα δεδομένα έργων πέρα του αρχείου RCP και να κατασκευάστει ο τρόπος εισαγωγής αυτών των αρχείων στο περιβάλλον του MS-PROJECT.

Βιβλιογραφία

ARTIGUES, C., MICHELON, P. & REUSSER, S. 2003. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research,* 149**,** 249-267.

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

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

BRUNI, M. E., BERALDI, P., GUERRIERO, F. & PINTO, E. 2011. A heuristic approach for resource constrained project scheduling with uncertain activity durations. *Computers & Operations Research,* 38**,** 1305-1318.

DEBLAERE, F., DEMEULEMEESTER, E. & HERROELEN, W. 2008. Exact and heuristic reactive planning procedures for multimode resource-constrained projects. *Available at SSRN 1288546*.

DEBLAERE, F., DEMEULEMEESTER, E. & HERROELEN, W. 2011. Reactive scheduling in the multi-mode RCPSP. *Computers & Operations Research,* 38**,** 63-74.

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. L. & HERROELEN, W. 2002. *Project scheduling : a research handbook,* Boston, Kluwer Academic Publishers.

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.

HARTMANN, S. 1998. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL),* 45**,** 733-750.

HARTMANN, S. & DREXL, A. 1998. Project scheduling with multiple modes: A comparison of exact algorithms. *Networks,* 32**,** 283-297.

HERROELEN, W. & LEUS, R. 2004. Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research,* 42**,** 1599-1620.

HERROELEN, W. & LEUS, R. 2005. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research,* 165**,** 289-306.

KE, H. & LIU, B. 2005. Project scheduling problem with stochastic activity duration times. *Applied Mathematics and Computation,* 168**,** 342-353.

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. & SPRECHER, A. 1997. PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European Journal of Operational Research,* 96**,** 205-216.

KORF, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence,* 27**,** 97-109.

KYRIAKIDIS, T. S., KOPANOS, G. M. & GEORGIADIS, M. C. 2012. MILP formulations for single- and multi-mode resource-constrained project scheduling problems. *Computers & Chemical Engineering,* 36**,** 369-385.

LAMBRECHTS, O., DEMEULEMEESTER, E. & HERROELEN, W. 2008. Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of scheduling,* 11**,** 121-136.

LOURENÇO, H. R., MARTIN, O. C. & STUTZLE, T. 2001. Iterated local search. *arXiv preprint math/0102188*.

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.

PROJECT MANAGEMENT INSTITUTE. 2013. A guide to the project management body of knowledge (PMBOK guide). 5th ed. Newtown Square, Pa.: Project Management Institute, Inc.

SADEH, N., OTSUKA, S. & SCHNELBACH, R. Predictive and reactive scheduling with the Micro-Boss production scheduling and control system. Proceedings, IJCAI-93 Workshop on Knowledge-Based Production Planning, Scheduling and Control, 1993.

SMITH, S. F. 1995. Reactive scheduling systems. *Intelligent scheduling systems.* Springer.

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

VAN DE VONDER, S., BALLESTÍN, F., DEMEULEMEESTER, E. & HERROELEN, W. 2007. Heuristic procedures for reactive project scheduling. *Computers & Industrial Engineering,* 52**,** 11-28.

VAN DE VONDER, S., DEMEULEMEESTER, E. & HERROELEN, W. 2008. Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research,* 189**,** 723-733.

WIESEMANN, W., KUHN, D. & RUSTEM, B. 2012. Multi-resource allocation in stochastic project scheduling. *Annals of Operations Research,* 193**,** 193-220.

Παράρτημα

Παράρτημα 1: Κώδικας S-SGS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Office.Interop.MSProject;

namespace MS_PROJECT_ADDIN
{
    class SGS
    {
        public static int[] algorithm(int sgs_rule, int[] genetic_sequence =
null)
        {
            Project project = Globals.ThisAddIn.Application.ActiveProject;

            SGS_variables sgs = new SGS_variables();
            sgs.Secg = new int[project.Tasks.Count];
            sgs.Dg = new int[project.Tasks.Count];
            sgs.Fg = new int[project.Tasks.Count];
            sgs.sj = new int[project.Tasks.Count];

            initialize(ref sgs);

            sgs.sj[0] = Convert.ToInt32(project.Tasks[1].Number19);
            sgs.Secg[0] = 1;
            sgs.Fg[0] = Convert.ToInt32(project.Tasks[1].Number20);


            for (int g = 1; g < project.Tasks.Count - 1; g++)
            {
                reset(project, ref sgs);
                sgs.Dg = schedule_availability(project, sgs);
                Rkt_populate(project, ref sgs);
                sgs.j    =    priority_selection(sgs_rule,    project,    sgs,
genetic_sequence);
                ESj_calculate(project, ref sgs);
                calculatable(project, ref sgs);
                for (int i = 0; i < project.Tasks.Count; i++)
                {
                    sgs.sj[project.Tasks.Count          -          1]          =
Math.Max(sgs.sj[project.Tasks.Count - 1], sgs.Fg[i]);
                }
            }
            return sgs.sj;
        }

        private static void initialize(ref SGS_variables sgs)
        {
            for (int i = 0; i < sgs.Dg.Length; i++)
            {
                sgs.Dg[i] = -1;
                sgs.Fg[i] = -1;
                sgs.Secg[i] = -1;
                sgs.sj[i] = -1;
            }
            sgs.ESj = -1;
            sgs.j = -1;
        }

        private static void reset(Project project, ref SGS_variables sgs)
```

```
        {
            for (int i = 0; i < project.Tasks.Count; i++)
            {
                sgs.Dg[i] = -1;
            }
            sgs.ESj = -1;
            sgs.j = -1;
            sgs.Rkt.Clear();
        }

        private    static    int[]    schedule_availability(Project    project,
SGS_variables sgs)
        {
            int[] Dg = new int[project.Tasks.Count];

            for (int i = 1; i <= project.Tasks.Count; i++)
            {
                if (sgs.Secg[i - 1] == 1)
                {
                    Dg[i - 1] = 0;
                }
                else
                {
                    for       (int       j       =       1;       j       <=
project.Tasks[i].PredecessorTasks.Count; j++)
                    {
                        if  (sgs.Secg[project.Tasks[i].PredecessorTasks[j].ID  -
1] == -1)
                        {
                            Dg[i - 1] = 0;
                            break;
                        }
                        else
                        {
                            Dg[i - 1] = 1;
                        }
                    }
                }
            }

            return Dg;
        }

        private  static  void  Rkt_populate(Project  project,  ref  SGS_variables
sgs)
        {
            bool already = false;
            for (int i = 0; i < project.Tasks.Count - 1; i++)
            {
                Rkt rkt = new Rkt();
                if (sgs.Secg[i] == 1)
                {
                    already = false;
                    foreach (Rkt t in sgs.Rkt)
                    {
                        if (t.Fg == sgs.Fg[i])
                        {
                            already = true;
                            break;
                        }
                    }
                    if (already == false)
```

```
                    {
                        rkt.Fg = sgs.Fg[i];
                        int[] resused = new int[project.Resources.Count];
                        for (int task = 1; task <= project.Tasks.Count; task++)
                        {
                            //if (sgs.sj[task - 1] == sgs.Fg[task - 1])
                            //{
                            //    if (sgs.sj[task - 1] == 0)
                            //    {
                            //                    for (int res = 0; res < project.Tasks[task].Assignments.Count; res++)
                            //        {
                            //                                int resource_id = project.Tasks[task].Assignments[res + 1].ResourceID;
                            //                        resused[resource_id - 1] = resused[resource_id - 1] + Convert.ToInt16(project.Tasks[task].Assignments[res + 1].Units);
                            //        }
                            //    }
                            //}
                            if (sgs.sj[task - 1] <= rkt.Fg && sgs.Fg[task - 1] > rkt.Fg)
                            {
                                for (int res = 0; res < project.Tasks[task].Assignments.Count; res++)
                                {
                                    int resource_id = project.Tasks[task].Assignments[res + 1].ResourceID;
                                    resused[resource_id - 1] = resused[resource_id - 1] + Convert.ToInt16(project.Tasks[task].Assignments[res + 1].Units);
                                }
                            }
                        }
                        for (int res = 0; res < project.Resources.Count; res++)
                        {
                            Resourceused used = new Resourceused();
                            used.resID = res + 1;
                            used.used = resused[res];
                            rkt.resused.Add(used);
                        }
                        sgs.Rkt.Add(rkt);
                    }
                }
            }
        }

        #region Priorities

        private static int priority_selection(int sgs_rule, Project project, SGS_variables sgs, int[] genetic_sequence)
        {
            int j = -1;

            switch (sgs_rule)
            {
                    //EBST 1 : Earliest Baseline Starting Time with weights
                case 1:
                    j = ebst1_rule(project, sgs);
                    break;

                    //EBST 2 : Earliest Baseline Starting Time without weights
```

```
                case 2:
                    j = ebst2_rule(project, sgs);
                    break;

                    //LST : Latest Starting Time
                case 3:
                    j = lst_rule(project, sgs);
                    break;

                    //LW : Largest Weight
                case 4:
                    j = lw_rule(project, sgs);
                    break;

                    //LAN : Lowest Activity(Task) Number
                case 5:
                    j = lan_rule(project, sgs);
                    break;

                    //Random : Random activity number (Normal - Gaussian
Distribution)
                case 6:
                    j = random_rule(project, sgs);
                    break;

                    //Genetic algorithm
                case 7:
                    j = genetic_rule(sgs, genetic_sequence);
                    break;
            }

            return j;
        }

        private static int ebst1_rule(Project project, SGS_variables sgs)
        {
            int j = -1;
            int min_base = 100000;
            int task_id = -1;
            int task_weight = -1;

            foreach (Task task in project.Tasks)
            {
                if (sgs.Dg[task.ID - 1] == 1)
                {
                    if (Convert.ToInt32(task.Number17) != min_base)
                    {
                        min_base              =              Math.Min(min_base,
Convert.ToInt32(task.Number17));
                        if (min_base == Convert.ToInt32(task.Number17))
                        {
                            task_id = task.ID;
                            task_weight = Convert.ToInt32(task.Number16);
                        }
                    }
                    else if (Convert.ToInt32(task.Number17) == min_base)
                    {
                        if (Convert.ToInt32(task.Number16) > task_weight)
                        {
                            min_base            =            Math.Min(min_base,
Convert.ToInt32(task.Number17));
                            if (min_base == Convert.ToInt32(task.Number17))
```

```
                            {
                                task_id = task.ID;
                                task_weight = Convert.ToInt32(task.Number16);
                            }
                        }
                    }
                }
            }

            j = task_id - 1;

            return j;
        }

        private static int ebst2_rule(Project project, SGS_variables sgs)
        {
            int j = -1;
            int min_base = 100000;
            int task_id = -1;

            foreach (Task task in project.Tasks)
            {
                if (sgs.Dg[task.ID - 1] == 1)
                {
                    if (Convert.ToInt32(task.Number17) != min_base)
                    {
                        min_base              =              Math.Min(min_base,
Convert.ToInt32(task.Number17));
                        if (min_base == Convert.ToInt32(task.Number17))
                        {
                            task_id = task.ID;
                        }
                    }
                }
            }

            j = task_id - 1;

            return j;
        }

        private static int lst_rule(Project project, SGS_variables sgs)
        {
            int j = -1;



            return j;
        }

        private static int lw_rule(Project project, SGS_variables sgs)
        {
            int j = -1;
            int max_weight = -1;
            int task_id = -1;

            foreach (Task task in project.Tasks)
            {
                if (sgs.Dg[task.ID - 1] == 1)
                {
                    max_weight              =              Math.Max(max_weight,
Convert.ToInt32(task.Number16));
```

```
                    if (max_weight == Convert.ToInt32(task.Number16))
                    {
                        task_id = task.ID;
                    }
                }
            }

            j = task_id - 1;

            return j;
        }

        private static int lan_rule(Project project, SGS_variables sgs)
        {
            int j = -1;

            j = Array.IndexOf(sgs.Dg, 1);

            return j;
        }

        private static int random_rule(Project project, SGS_variables sgs)
        {
            int j = -1;
            int count = 0;
            for (int i = 0; i < project.Tasks.Count; i++)
            {
                if (sgs.Dg[i] == 1)
                    count++;
            }
            int[] eligible_tasks = new int[count];
            int q = 0;
            for (int i = 0; i < project.Tasks.Count; i++)
            {
                if (sgs.Dg[i] == 1)
                {
                    eligible_tasks[q] = i;
                    q++;
                }
            }

            int task_id = -1;
            int eligible_tasks_chance = Randomizer.random(1, (count * 25 + 1));
            decimal                   elibigle_tasks_decimal                   =
decimal.Divide(eligible_tasks_chance, 25);
            task_id = Convert.ToInt32(Math.Ceiling(elibigle_tasks_decimal));
            j = eligible_tasks[task_id];

            return j;
        }

        private    static    int    genetic_rule(SGS_variables    sgs,    int[]
genetic_sequence)
        {
            int j = -1;
            int task_id = -1;

            for (int i = 0; i < genetic_sequence.Length; i++)
            {
                task_id = genetic_sequence[i];
                if (sgs.Dg[task_id - 1] == 1)
                {
```

```
                    j = task_id - 1;
                    break;
                }
            }

            return j;
        }

        #endregion

        private static void ESj_calculate(Project project, ref SGS_variables
sgs)
        {
            int predecessor_id = -1;

            foreach (TaskDependency dependency in project.Tasks[sgs.j +
1].TaskDependencies)
            {
                if (dependency.To.ID == project.Tasks[sgs.j + 1].ID)
                {
                    predecessor_id = dependency.From.ID;
                    if (sgs.ESj < (sgs.Fg[predecessor_id - 1] + (dependency.Lag
/ 480)))
                    {
                        sgs.ESj = (sgs.Fg[predecessor_id - 1] + (dependency.Lag
/ 480));
                    }
                }
            }

            //for (int task = 1; task <= project.Tasks.Count; task++)
            //{
            //              for (int successor = 1; successor <=
project.Tasks[task].SuccessorTasks.Count; successor++)
            //    {
            //          if (project.Tasks[task].SuccessorTasks[successor].ID ==
project.Tasks[sgs.j + 1].ID)
            //        {
            //            if (sgs.ESj < sgs.Fg[task - 1])
            //            {
            //                sgs.ESj = sgs.Fg[task - 1];
            //            }
            //        }
            //    }
            //}
        }

        private static void calculatable(Project project, ref SGS_variables
sgs)
        {
            for (int i = 0; i < sgs.Rkt.Count; i++)
            {
                bool safe = false;
                if ((sgs.sj[sgs.j] > sgs.Rkt[i].Fg && sgs.Rkt[i].Fg >= sgs.ESj)
|| (sgs.sj[sgs.j] == -1 && sgs.Rkt[i].Fg >= sgs.ESj))
                {
                    int resources_feasible = 0;
                    for (int t = 0; t < sgs.Rkt.Count; t++)
                    {
                        int count = 0;
                        if (sgs.Rkt[t].Fg >= sgs.Rkt[i].Fg && sgs.Rkt[t].Fg <
sgs.Rkt[i].Fg + (project.Tasks[sgs.j + 1].Duration / 480))
```

```
                        {
                            for (int res = 0; res < project.Tasks[sgs.j +
1].Assignments.Count; res++)
                            {
                                int resource_id = project.Tasks[sgs.j +
1].Assignments[res + 1].ResourceID;
                                string resource_name = project.Tasks[sgs.j +
1].Assignments[res + 1].ResourceName;
                                if (sgs.Rkt[t].resused[resource_id - 1].used +
project.Tasks[sgs.j + 1].Assignments[res + 1].Units <=
project.Resources[resource_name].MaxUnits)
                                {
                                    count++;
                                }
                            }
                        }
                        else if (sgs.Rkt[t].Fg < sgs.Rkt[i].Fg)
                        {
                            resources_feasible++;
                        }
                        else if (sgs.Rkt[t].Fg >= sgs.Rkt[i].Fg +
(project.Tasks[sgs.j + 1].Duration / 480))
                        {
                            resources_feasible++;
                        }
                        if (count == project.Tasks[sgs.j +
1].Assignments.Count)
                        {
                            resources_feasible++;
                        }
                    }
                    if (resources_feasible == sgs.Rkt.Count)
                    {
                        safe = true;
                    }
                    else
                    {
                        safe = false;
                    }
                }
                if (safe == true)
                {
                    sgs.sj[sgs.j] = sgs.Rkt[i].Fg;
                    sgs.Fg[sgs.j] = sgs.sj[sgs.j] + (project.Tasks[sgs.j +
1].Duration / 480);
                    sgs.Secg[sgs.j] = 1;
                }
            }
        }
    }
}
```

Παράρτημα 2: Κώδικας Γενετικού αλγορίθμου

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Office.Interop.MSProject;

namespace MS_PROJECT_ADDIN
{
    class Genetic
    {
        public static void algorithm()
        {
            Project project = Globals.ThisAddIn.Application.ActiveProject;

            genetic_variables genetic_settings = new genetic_variables();
            genetic_settings = Properties.Settings.Default.GeneticSettings;

            List<chromosome> parents = new List<chromosome>();
            List<chromosome> children = new List<chromosome>();
            List<chromosome> generation = new List<chromosome>();

            System.Diagnostics.Stopwatch stopwatch = new
System.Diagnostics.Stopwatch();
            stopwatch.Start();

            initialize(genetic_settings, project, ref parents);

            chromosome best_current = new chromosome();

            for (int g = 1; g <= genetic_settings.generations; g++)
            {
                crossover(genetic_settings, parents, ref children);
                mutation(genetic_settings, ref children);
                compact_population(genetic_settings, parents, children, ref
generation);
                foreach (chromosome chromosome in generation)
                {
                    chromosome.fitness = fitness_calculation(project,
chromosome);
                }
                selection(genetic_settings, ref parents, generation);
                best_current = parents[0];
            }

            Secondary_functions.sj_Fg_populate(SGS.algorithm(7,
best_current.tasks_id_genes));

            stopwatch.Stop();
            project.Tasks[1].Number16 = stopwatch.ElapsedMilliseconds;

            Secondary_functions.save_file("R_Genetic");
        }

        private static void initialize(genetic_variables settings, Project
project, ref List<chromosome> parents)
        {
            bool duplicate_id = true;

            for (int i = 0; i < settings.population; i++)
            {
```

```
                chromosome parent = new chromosome();
                parent.tasks_id_genes = new int[project.Tasks.Count - 2];

                for (int j = 0; j < parent.tasks_id_genes.Length; j++)
                {
                    do
                    {
                        int task_id = Randomizer.random(2,
project.Tasks.Count);

                        if (parent.tasks_id_genes.Contains(task_id) == false)
                        {
                            parent.tasks_id_genes[j] = task_id;
                            duplicate_id = false;
                        }
                        else
                        {
                            duplicate_id = true;
                        }
                    } while (duplicate_id == true);
                }

                parents.Add(parent);
            }
        }

        #region Crossover Methods

        private static void crossover(genetic_variables settings,
List<chromosome> parents, ref List<chromosome> children)
        {
            children.Clear();
            switch (settings.crossover_method)
            {
                    //One Point Crossover
                case 1:
                    one_point_crossover(parents, ref children);
                    break;

                    //Two Point Crossover
                case 2:
                    two_point_crossover(parents, ref children);
                    break;
            }
        }

        private static void one_point_crossover(List<chromosome> parents, ref
List<chromosome> children)
        {
            int father_id = -1;
            int mother_id = -1;
            int q = -1;

            for (int i = 0; i < (parents.Count / 2); i++)
                 {
                father_id = 2 * i;
                mother_id = (2 * i) + 1;

                q = Randomizer.random(0,
parents[father_id].tasks_id_genes.Length - 1);

                chromosome son = new chromosome();
```

```
                chromosome daughter = new chromosome();

                son.tasks_id_genes                    =                    new
int[parents[father_id].tasks_id_genes.Length];
                daughter.tasks_id_genes                  =                    new
int[parents[mother_id].tasks_id_genes.Length];

                for (int j = 0; j < q; j++)
                {
                    son.tasks_id_genes[j]                                      =
parents[father_id].tasks_id_genes[j];
                    daughter.tasks_id_genes[j]                                 =
parents[mother_id].tasks_id_genes[j];
                }

                int task_location = -1;

                for (int j = q; j < parents[father_id].tasks_id_genes.Length;
j++)
                {
                    task_location = viable_task(son, parents[mother_id]);
                    son.tasks_id_genes[j]                                      =
parents[mother_id].tasks_id_genes[task_location];

                    task_location = viable_task(daughter, parents[father_id]);
                    daughter.tasks_id_genes[j]                                 =
parents[father_id].tasks_id_genes[task_location];
                }

                children.Add(son);
                children.Add(daughter);
                    }
        }

        private static void two_point_crossover(List<chromosome> parents, ref
List<chromosome> children)
        {
            int father_id = -1;
            int mother_id = -1;
            int q1 = -1;
            int q2 = -1;

            for (int i = 0; i < (parents.Count / 2); i++)
            {
                father_id = 2 * i;
                mother_id = (2 * i) + 1;

                q1                    =                   Randomizer.random(0,
parents[father_id].tasks_id_genes.Length - 1);
                q2          =          Randomizer.random(q1          +          1,
parents[father_id].tasks_id_genes.Length);

                chromosome son = new chromosome();
                chromosome daughter = new chromosome();

                son.tasks_id_genes                    =                    new
int[parents[father_id].tasks_id_genes.Length];
                daughter.tasks_id_genes                  =                    new
int[parents[mother_id].tasks_id_genes.Length];

                for (int j = 0; j < q1; j++)
                {
```

```
                    son.tasks_id_genes[j]                                       =
parents[father_id].tasks_id_genes[j];
                    daughter.tasks_id_genes[j]                                  =
parents[mother_id].tasks_id_genes[j];
                }

                int task_location = -1;

                for (int j = q1; j < q2; j++)
                {
                    task_location = viable_task(son, parents[mother_id]);
                    son.tasks_id_genes[j]                                       =
parents[mother_id].tasks_id_genes[task_location];

                    task_location = viable_task(daughter, parents[father_id]);
                    daughter.tasks_id_genes[j]                                  =
parents[father_id].tasks_id_genes[task_location];
                }

                for (int j = q2; j < parents[father_id].tasks_id_genes.Length;
j++)
                {
                    task_location = viable_task(son, parents[father_id]);
                    son.tasks_id_genes[j]                                       =
parents[father_id].tasks_id_genes[task_location];

                    task_location = viable_task(daughter, parents[mother_id]);
                    daughter.tasks_id_genes[j]                                  =
parents[mother_id].tasks_id_genes[task_location];
                }

                children.Add(son);
                children.Add(daughter);
            }
        }

        private static int viable_task(chromosome kid, chromosome parent)
        {
            int task_location = -1;
            int count = 0;

            for (int i = 0; i < parent.tasks_id_genes.Length; i++)
            {
                count = 0;
                for (int j = 0; j < kid.tasks_id_genes.Length; j++)
                {
                    if (parent.tasks_id_genes[i] != kid.tasks_id_genes[j])
                    {
                        count++;
                    }
                }
                if (count == kid.tasks_id_genes.Length)
                {
                    task_location = i;
                    break;
                }
            }

            return task_location;
        }

        #endregion
```

```
        private static void mutation(genetic_variables settings, ref
List<chromosome> children)
        {
            foreach (chromosome child in children)
            {
                int chromosome_mutation_chance = Randomizer.random(0, 101);
                if (chromosome_mutation_chance >=
settings.mutation_chance_task)
                {
                    int gene_1 = Randomizer.random(0,
child.tasks_id_genes.Length);
                    int gene_2 = Randomizer.random(0,
child.tasks_id_genes.Length);
                    if (gene_1 != gene_2)
                    {
                        int temp_gene_1_value = -1;
                        int temp_gene_2_value = -1;
                        temp_gene_1_value = child.tasks_id_genes[gene_1];
                        temp_gene_2_value = child.tasks_id_genes[gene_2];
                        child.tasks_id_genes[gene_2] = temp_gene_1_value;
                        child.tasks_id_genes[gene_1] = temp_gene_2_value;
                    }
                }
            }
        }

        private static bool feasibility_check(Project project, chromosome
child)
        {
            bool feasible = true;
            int predecessor_count = -1;
            int task_id = -1;
            int count = 0;

            for (int i = 0; i < child.tasks_id_genes.Length; i++)
            {
                count = 0;
                task_id = child.tasks_id_genes[i];
                predecessor_count =
project.Tasks[task_id].PredecessorTasks.Count;

                foreach (Task predecessor in
project.Tasks[task_id].PredecessorTasks)
                {
                    for (int j = 0; j < i; j++)
                    {
                        if (predecessor.ID == child.tasks_id_genes[j])
                        {
                            count++;
                        }
                    }
                }
                if (count != predecessor_count)
                {
                    feasible = false;
                    break;
                }
            }

            return feasible;
        }
```

```
        private static void compact_population(genetic_variables settings
,List<chromosome> parents, List<chromosome> children, ref List<chromosome>
generation)
        {
            generation.Clear();
            generation.AddRange(parents);
            generation.AddRange(children);
        }

        private static void selection(genetic_variables settings, ref
List<chromosome> parents, List<chromosome> generation)
        {
            parents.Clear();
            List<chromosome> ordered_generation = generation.OrderBy(chromosome
=> chromosome.fitness).ToList<chromosome>();
            for (int i = 0; i < settings.generations; i++)
            {
                parents.Add(ordered_generation[i]);
            }
        }

        private static int fitness_calculation(Project project, chromosome
chromosome)
        {
            int[] sj = SGS.algorithm(7, chromosome.tasks_id_genes);

            Double fitness = sj.Max();
            Double fit_multiplier = 1;

            for (int i = 0; i < sj.Length; i++)
            {
                if (sj[i] >= project.Tasks[i + 1].Number17 + ((project.Tasks[i
+ 1].Duration / 480) / 2) || sj[i] <= project.Tasks[i + 1].Number17 -
((project.Tasks[i + 1].Duration / 480) / 2))
                {
                    fit_multiplier += 0.2;
                }
            }

            fitness *= fit_multiplier;

            return Convert.ToInt32(fitness);
        }
    }
}
```

Παράρτημα 3: Τυχαιοποιητής

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

namespace MS_PROJECT_ADDIN
{
    class Randomizer
    {
        private static readonly Random getrandom = new Random();
        private static readonly object syncLock = new object();
        public static int random(int min, int max)
        {
            lock (syncLock) // synchronize
            {
                return getrandom.Next(min, max);
            }
        }
    }
}
```

Παράρτημα 4: Εισαγωγή δεδομένων από RCP αρχείο

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Text.RegularExpressions;
using Microsoft.Office.Interop.MSProject;

namespace MS_PROJECT_ADDIN
{
    class Input_File
    {
        public static void RCPinput(string input)
        {
            Project project = Globals.ThisAddIn.Application.ActiveProject;

            StreamReader sr = new StreamReader(input);
            string FileContent = sr.ReadToEnd();
            string[] fileLines = Regex.Split(FileContent, "\r\n");
            string[] constants = Regex.Split(fileLines[0], @"\s\s\s\s\s");
            string[] avail = Regex.Split(fileLines[1], @"\s\s\s\s\s");
            int actnrs = Convert.ToInt32(constants[0]);
            int resnrs = Convert.ToInt32(constants[1]);
            for (int i = 1; i <= resnrs; i++)
            {
                project.Resources.Add(i);
                project.Resources[i].MaxUnits = Convert.ToInt32(avail[i - 1]);
            }
            for (int i = 1; i <= actnrs; i++)
            {
                string[]    lineparts    =    Regex.Split(fileLines[i    +    1],
@"\s\s\s\s\s");
                project.Tasks.Add(i);
                project.Tasks[i].Duration = Convert.ToInt32(lineparts[0])*480;

                int k = 0;

                for (int j = 1; j <= resnrs; j++)
                {
                    if (Convert.ToInt32(lineparts[j]) != 0)
                    {
                        k = k + 1;
                        project.Tasks[i].Assignments.Add(ResourceID:j);
                        project.Tasks[i].Assignments[k].Units                    =
Convert.ToInt32(lineparts[j]);
                    }
                }
            }
            for (int i=1;i<=actnrs;i++)
            {
                string[]    lineparts    =    Regex.Split(fileLines[i    +    1],
@"\s\s\s\s\s");
                int successors = 0;
                successors = Convert.ToInt32(lineparts[resnrs + 1]);
                if(successors != 0)
                {
                    for (int k = 1; k <= successors; k++)
                    {
                        int succID = 0;
                        succID = Convert.ToInt16(lineparts[k + resnrs + 1]);
```

```
                        project.Tasks[i].LinkSuccessors(project.Tasks[succID]);
                    }
                }
            }

            project.Tasks[1].Milestone = true;
            project.Tasks[1].Name = "Start";
            project.Tasks[actnrs].Milestone = true;
            project.Tasks[actnrs].Name = "Finish";

            sr.Close();

            Secondary_functions.save_file("R_RCP",  save_mydocs:  false,  input:
true, input_file: input);
        }
    }
}
```

Παράρτημα 5: Εισαγωγή δεδομένων από MPP αρχείο

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Office.Interop.MSProject;

namespace MS_PROJECT_ADDIN
{
    class Convert_File
    {
        public static void convert()
        {
            Project                        original_project                        =
Globals.ThisAddIn.Application.ActiveProject;

            Globals.ThisAddIn.Application.Projects.Add();

            Project                          rcpsp_project                         =
Globals.ThisAddIn.Application.ActiveProject;

            List<Task_IDs_table> tasks_ids = new List<Task_IDs_table>();

            resource_selection(original_project, ref rcpsp_project);

            rcpsp_project.Tasks.Add("Start");
            rcpsp_project.Tasks["Start"].Duration = 0;
            rcpsp_project.Tasks["Start"].Start = original_project.ProjectStart;

            tasks_selection(original_project,     ref     rcpsp_project,     ref
tasks_ids);

            rcpsp_project.Tasks.Add("Finish");
            rcpsp_project.Tasks["Finish"].Duration = 0;
            rcpsp_project.Tasks["Finish"].Start                                  =
original_project.ProjectFinish;

            resource_assignment(original_project, ref rcpsp_project);

            tasks_linking(original_project, ref rcpsp_project, tasks_ids);

            Secondary_functions.column_creation();

            Secondary_functions.baseline_populate();

            Secondary_functions.save_file("Converted");
        }

        private  static  void  resource_selection(Project  original_project,  ref
Project rcpsp_project)
        {
            for (int i = 1; i <= original_project.Resources.Count; i++)
            {
                string resource_name = original_project.Resources[i].Name;

                if (original_project.Resources[i].Type == 0)
                {
                    rcpsp_project.Resources.Add(resource_name);
                    rcpsp_project.Resources[resource_name].MaxUnits            =
original_project.Resources[i].MaxUnits;
```

```
                }
            }
        }

        private static void tasks_selection(Project original_project, ref
Project rcpsp_project, ref List<Task_IDs_table> tasks_ids)
        {
            int count = 2;

            for (int i = 1; i <= original_project.Tasks.Count; i++)
            {
                Task_IDs_table id = new Task_IDs_table();
                string task_name = original_project.Tasks[i].Name;

                if (original_project.Tasks[i].OutlineChildren.Count == 0)
                {
                    rcpsp_project.Tasks.Add(task_name);

                    id.ID_preconvert = original_project.Tasks[i].ID;
                    id.ID_postconvert = count;
                    tasks_ids.Add(id);
                    count++;

                    rcpsp_project.Tasks[id.ID_postconvert].Manual = false;
                    rcpsp_project.Tasks[id.ID_postconvert].Duration            =
original_project.Tasks[i].Duration;
                    rcpsp_project.Tasks[id.ID_postconvert].Start               =
original_project.Tasks[i].Start;
                    rcpsp_project.Tasks[id.ID_postconvert].Finish              =
original_project.Tasks[i].Finish;
                    rcpsp_project.Tasks[id.ID_postconvert].PercentComplete     =
original_project.Tasks[i].PercentComplete;
                    rcpsp_project.Tasks[id.ID_postconvert].Baseline1Start      =
original_project.Tasks[i].Baseline1Start;
                    rcpsp_project.Tasks[id.ID_postconvert].Baseline1Finish     =
original_project.Tasks[i].Baseline1Finish;
                }
                else if (original_project.Tasks[i].PredecessorTasks.Count != 0
|| original_project.Tasks[i].SuccessorTasks.Count != 0)
                {
                    rcpsp_project.Tasks.Add(task_name);

                    id.ID_preconvert = original_project.Tasks[i].ID;
                    id.ID_postconvert = count;
                    tasks_ids.Add(id);
                    count++;

                    rcpsp_project.Tasks[id.ID_postconvert].Manual = false;
                    rcpsp_project.Tasks[id.ID_postconvert].Duration            =
original_project.Tasks[i].Duration;
                    rcpsp_project.Tasks[id.ID_postconvert].Start               =
original_project.Tasks[i].Start;
                    rcpsp_project.Tasks[id.ID_postconvert].Finish              =
original_project.Tasks[i].Finish;
                    rcpsp_project.Tasks[id.ID_postconvert].PercentComplete     =
original_project.Tasks[i].PercentComplete;
                    rcpsp_project.Tasks[id.ID_postconvert].Baseline1Start      =
original_project.Tasks[i].Baseline1Start;
                    rcpsp_project.Tasks[id.ID_postconvert].Baseline1Finish     =
original_project.Tasks[i].Baseline1Finish;
                }
            }
```

```
        }

        private static void resource_assignment(Project original_project, ref
Project rcpsp_project)
        {
            for (int i = 2; i < rcpsp_project.Tasks.Count; i++)
            {
                string task_name = rcpsp_project.Tasks[i].Name;

                if (original_project.Tasks[task_name].Assignments.Count != 0)
                {
                    int assign_count = 0;
                    for (int j = 1; j <=
original_project.Tasks[task_name].Assignments.Count; j++)
                    {
                        string resource_name =
original_project.Tasks[task_name].Assignments[j].ResourceName;

                        if
(original_project.Tasks[task_name].Assignments[j].ResourceType == 0)
                        {
                            assign_count++;
                            int resource_id =
rcpsp_project.Resources[resource_name].ID;
                            rcpsp_project.Tasks[i].Assignments.Add(ResourceID :
resource_id);

rcpsp_project.Tasks[i].Assignments[assign_count].Units =
original_project.Tasks[task_name].Assignments[j].Units;
                        }
                    }
                    rcpsp_project.Tasks[i].PercentComplete =
original_project.Tasks[task_name].PercentComplete;
                }
            }
        }

        private static void tasks_linking(Project original_project, ref Project
rcpsp_project, List<Task_IDs_table> tasks_ids)
        {
            foreach(Task task in rcpsp_project.Tasks)
            {
                if (task.Name != "Start" && task.Name != "Finish")
                {
                    Task_IDs_table id = new Task_IDs_table();
                    id = tasks_ids.Find(x => x.ID_postconvert == task.ID);

                    if
(original_project.Tasks[id.ID_preconvert].PredecessorTasks.Count == 0)
                    {
                        task.LinkPredecessors(rcpsp_project.Tasks["Start"]);
                        rcpsp_project.Tasks["Start"].Manual = false;
                        rcpsp_project.Tasks["Start"].Duration = 0;
                    }
                    else
                    {
                        dependencies(original_project, ref rcpsp_project,
tasks_ids);
                    }
                    if
(original_project.Tasks[id.ID_preconvert].SuccessorTasks.Count == 0)
                    {
```

```
                        task.LinkSuccessors(rcpsp_project.Tasks["Finish"]);
                        rcpsp_project.Tasks["Finish"].Manual = false;
                        rcpsp_project.Tasks["Finish"].Duration = 0;
                    }
                }
            }
        }

        private static void dependencies(Project original_project, ref Project
rcpsp_project, List<Task_IDs_table> tasks_ids)
        {
            int lag = 0;
            int new_pred_id = 0;
            int old_task_id = 0;

            foreach (Task task in rcpsp_project.Tasks)
            {
                if (task.Name != "Start" && task.Name != "Finish")
                {
                    Task_IDs_table task_ids = new Task_IDs_table();
                    task_ids   =   tasks_ids.Find(x   =>   x.ID_postconvert   ==
task.ID);
                    old_task_id = task_ids.ID_preconvert;

                    foreach        (TaskDependency        dependency        in
original_project.Tasks[old_task_id].TaskDependencies)
                    {
                        if (dependency.From.ID != old_task_id)
                        {
                            Task_IDs_table pred_ids = new Task_IDs_table();
                            pred_ids  =  tasks_ids.Find(x  =>  x.ID_preconvert  ==
dependency.From.ID);
                            new_pred_id = pred_ids.ID_postconvert;

                            lag                                                 =
precedence_transmogrification(original_project, dependency);

task.TaskDependencies.Add(rcpsp_project.Tasks[new_pred_id],
PjTaskLinkType.pjFinishToStart, lag.ToString());
                        }
                    }
                }
            }
        }

        private      static      int      precedence_transmogrification(Project
original_project, TaskDependency dependency)
        {
            int new_lag = 0;

            if (dependency.Type == PjTaskLinkType.pjStartToStart)
            {
                new_lag        =        Convert.ToInt32(dependency.Lag        -
original_project.Tasks[dependency.From.ID].Duration);
            }
            else if (dependency.Type == PjTaskLinkType.pjStartToFinish)
            {
                new_lag        =        Convert.ToInt32(dependency.Lag        -
original_project.Tasks[dependency.From.ID].Duration                           -
original_project.Tasks[dependency.To.ID].Duration);
            }
            else if (dependency.Type == PjTaskLinkType.pjFinishToStart)
```

```
            {
                new_lag = Convert.ToInt32(dependency.Lag);
            }
            else if (dependency.Type == PjTaskLinkType.pjFinishToFinish)
            {
                new_lag        =        Convert.ToInt32(dependency.Lag        -
original_project.Tasks[dependency.To.ID].Duration);
            }

            return new_lag;
        }
    }
}
```

Παράρτημα 6: Ομάδα έργων J301 σε μορφή RCP αρχείου

```
J301_1
32     4
12     13     4      12
0      0      0      0      0      3      2      3      4
8      4      0      0      0      3      6      11     15
4      10     0      0      0      3      7      8      13
6      0      0      0      3      3      5      9      10
3      3      0      0      0      1      20
8      0      0      0      8      1      30
5      4      0      0      0      1      27
9      0      1      0      0      3      12     19     27
2      6      0      0      0      1      14
7      0      0      0      1      2      16     25
9      0      5      0      0      2      20     26
2      0      7      0      0      1      14
6      4      0      0      0      2      17     18
3      0      8      0      0      1      17
9      3      0      0      0      1      25
10     0      0      0      5      2      21     22
6      0      0      0      8      1      22
5      0      0      0      7      2      20     22
3      0      1      0      0      2      24     29
7      0      10     0      0      2      23     25
2      0      0      0      6      1      28
7      2      0      0      0      1      23
2      3      0      0      0      1      24
3      0      9      0      0      1      30
3      4      0      0      0      1      30
7      0      0      4      0      1      31
8      0      0      0      7      1      28
3      0      8      0      0      1      31
7      0      7      0      0      1      32
2      0      7      0      0      1      32
2      0      0      2      0      1      32
0      0      0      0      0      0
```

```
J301_2
32    4
14    10    11    14
0     0     0     0     0     3     2     3     4
2     0     0     2     0     1     10
10    0     8     0     0     3     5     9     15
7     0     0     7     0     2     7     13
1     8     0     0     0     2     6     17
3     0     0     0     8     3     13    16    18
6     0     0     5     0     3     8     11    19
2     0     9     0     0     1     27
10    0     0     0     2     2     14    28
5     1     0     0     0     2     12    22
9     2     0     0     0     2     21    26
4     0     0     6     0     1     24
4     0     5     0     0     1     25
2     0     0     10    0     1     25
4     0     0     0     10    1     21
10    10    0     0     0     2     20    26
10    0     0     7     0     3     21    26    29
7     5     0     0     0     1     23
10    0     0     0     7     1     25
7     0     0     3     0     1     31
2     0     7     0     0     1     31
1     10    0     0     0     2     27    29
2     8     0     0     0     1     31
2     0     0     0     8     1     29
2     0     0     0     7     1     30
9     7     0     0     0     1     27
1     0     0     0     1     1     30
4     8     0     0     0     1     30
7     0     3     0     0     1     32
8     0     0     8     0     1     32
9     0     6     0     0     1     32
0     0     0     0     0     0
```

```
J301_3
32    4
10    8    13    12
0    0    0    0    0    3    2    3    4
1    0    0    0    5    2    23    24
1    0    3    0    0    3    5    6    17
1    8    0    0    0    2    7    20
7    0    0    2    0    3    10    22    28
6    0    0    0    3    1    18
4    1    0    0    0    3    8    9    12
5    0    0    10    0    3    14    21    27
8    0    0    3    0    2    11    16
7    0    0    0    1    1    16
8    9    0    0    0    1    17
1    7    0    0    0    3    13    15    16
2    0    3    0    0    1    30
3    0    0    0    6    1    19
10    0    7    0    0    1    26
10    3    0    0    0    1    25
2    0    0    3    0    1    24
10    0    0    4    0    1    21
1    0    0    0    3    1    25
1    0    0    7    0    2    25    27
7    0    2    0    0    1    22
9    0    0    0    10    2    29    30
9    0    0    7    0    1    31
4    0    4    0    0    1    27
4    0    3    0    0    1    26
1    0    0    4    0    1    30
1    9    0    0    0    1    28
8    0    0    0    9    1    31
1    0    0    0    1    1    32
2    0    8    0    0    1    32
7    0    4    0    0    1    32
0    0    0    0    0    0
```

```
J301_4
32    4
7     11    11    15
0     0     0     0     0     3     2     3     4
3     7     0     0     0     3     6     14    23
10    0     8     0     0     1     29
10    0     5     0     0     1     5
7     0     0     10    0     1     8
6     0     0     0     5     3     7     15    19
9     0     0     0     4     3     13    20    24
7     0     0     4     0     3     9     10    22
9     0     0     0     10    1     26
4     0     0     0     4     3     11    12    26
4     0     0     0     9     1     17
9     0     0     3     0     2     20    29
7     0     9     0     0     3     16    18    21
9     0     0     0     9     1     27
5     0     5     0     0     2     16    25
7     0     0     0     9     1     28
1     0     2     0     0     1     29
7     0     0     0     3     1     31
2     0     0     0     10    1     26
9     0     7     0     0     1     30
2     0     0     3     0     1     25
7     0     2     0     0     1     23
3     0     0     0     4     2     25    31
10    0     0     10    0     1     31
1     0     0     0     1     1     28
4     5     0     0     0     1     30
4     0     0     7     0     1     28
1     0     0     1     0     1     30
10    0     6     0     0     1     32
9     0     0     0     6     1     32
8     0     0     0     10    1     32
0     0     0     0     0     0
```

```
J301_5
32    4
11    11    9     11
0     0     0     0     0     3     2     3     4
6     0     0     7     0     3     11    12    15
4     0     0     1     0     3     5     8     16
2     0     0     0     2     1     27
1     0     0     6     0     3     6     12    14
5     2     0     0     0     1     7
1     0     0     0     6     2     9     28
3     0     9     0     0     2     10    20
5     0     8     0     0     2     13    21
6     8     0     0     0     2     17    25
6     0     0     7     0     2     25    28
4     7     0     0     0     2     25    29
1     8     0     0     0     3     22    23    31
6     0     1     0     0     1     17
5     0     0     0     2     1     24
3     0     0     5     0     1     18
3     0     2     0     0     1     26
2     7     0     0     0     2     19    23
3     0     8     0     0     1     22
3     0     0     5     0     1     29
7     0     0     0     2     1     31
4     0     0     0     1     1     24
8     0     0     0     5     1     24
1     0     0     0     5     1     29
8     0     9     0     0     1     30
8     0     5     0     0     1     28
5     0     9     0     0     1     30
1     0     4     0     0     1     30
5     0     4     0     0     1     32
2     0     9     0     0     1     32
1     0     0     0     10    1     32
0     0     0     0     0     0
```

```
J301_6
32    4
12    10    10    12
0    0    0    0    0    3    2    3    4
10    0    0    0    4    3    5    7    8
1    0    0    0    10    1    11
9    4    0    0    0    2    6    16
3    6    0    0    0    2    15    23
1    3    0    0    0    2    10    12
7    0    4    0    0    3    9    14    25
1    0    0    0    2    1    13
4    10    0    0    0    1    24
10    0    0    0    2    1    22
6    0    0    10    0    3    14    16    24
2    0    0    0    6    2    13    21
3    0    7    0    0    3    17    24    30
1    0    0    3    0    1    18
3    0    0    0    6    2    16    29
1    0    0    10    0    1    19
3    0    0    0    7    1    18
10    0    0    0    9    2    20    31
1    0    6    0    0    1    28
3    5    0    0    0    1    26
4    0    3    0    0    1    28
2    8    0    0    0    1    28
4    1    0    0    0    1    27
2    3    0    0    0    2    26    31
4    0    9    0    0    1    30
6    0    0    0    7    1    29
9    0    0    0    7    1    30
2    0    0    0    5    1    31
1    0    0    9    0    1    32
1    0    0    9    0    1    32
9    0    0    4    0    1    32
0    0    0    0    0    0
```

J301_7
32 4
8 13 11 10
0 0 0 0 0 3 2 3 4
2 0 10 0 0 3 5 7 13
3 0 3 0 0 3 6 18 24
6 0 0 9 0 1 19
8 8 0 0 0 3 8 9 10
3 0 6 0 0 3 11 13 22
2 0 8 0 0 1 14
2 0 9 0 0 1 29
1 0 5 0 0 2 15 25
8 0 10 0 0 1 12
4 0 5 0 0 1 14
3 0 0 0 5 1 16
2 0 1 0 0 2 23 26
7 4 0 0 0 1 30
6 0 0 5 0 1 19
8 0 0 8 0 3 17 21 22
5 0 0 1 0 2 20 26
6 0 0 0 9 1 28
8 0 0 7 0 1 29
5 0 8 0 0 1 25
2 0 0 8 0 1 26
1 8 0 0 0 3 23 27 29
3 0 0 0 1 1 28
6 0 0 0 4 1 31
5 0 1 0 0 1 28
10 0 0 0 9 1 27
8 0 4 0 0 1 30
5 0 0 1 0 1 31
5 0 0 8 0 1 32
8 5 0 0 0 1 32
6 0 0 0 6 1 32
0 0 0 0 0 0

```
J301_8
32    4
12    14    12    10
0     0     0     0     0     3     2     3     4
8     10    0     0     0     3     9     17    25
3     0     4     0     0     3     5     10    11
6     0     5     0     0     3     8     15    19
9     0     4     0     0     3     6     7     13
8     0     0     6     0     1     15
6     0     0     0     8     2     30    31
4     0     0     0     5     2     12    14
5     0     8     0     0     3     18    22    23
4     0     0     0     1     2     12    14
5     8     0     0     0     2     13    29
7     10    0     0     0     1     27
1     1     0     0     0     1     28
10    0     8     0     0     1     16
1     0     0     4     0     1     28
5     0     0     0     3     1     18
4     0     0     0     9     2     19    20
10    0     0     10    0     1     21
3     0     0     3     0     1     26
5     0     0     5     0     1     26
10    0     2     0     0     1     29
1     0     0     5     0     1     27
1     0     2     0     0     1     24
3     3     0     0     0     1     27
10    0     8     0     0     1     26
1     0     0     0     3     1     28
1     0     1     0     0     1     29
7     0     0     5     0     1     31
8     0     10    0     0     1     32
9     0     0     7     0     1     32
2     0     6     0     0     1     32
0     0     0     0     0     0
```

```
J301_9
32    4
12    11    11    13
0    0    0    0    0    3    2    3    4
5    0    0    0    8    3    6    7    10
3    0    7    0    0    1    20
7    4    0    0    0    1    5
3    0    0    6    0    2    8    17
4    0    9    0    0    1    13
6    8    0    0    0    1    12
7    5    0    0    0    3    9    14    16
2    7    0    0    0    3    19    21    22
10    0    3    0    0    3    11    15    24
6    0    0    5    0    1    31
5    0    0    10    0    1    30
5    2    0    0    0    1    14
8    0    6    0    0    2    27    31
7    4    0    0    0    1    18
7    7    0    0    0    1    26
3    0    0    0    5    1    25
6    0    8    0    0    3    22    25    27
7    6    0    0    0    1    23
8    0    8    0    0    2    26    29
9    6    0    0    0    2    25    26
10    0    0    0    7    2    29    31
7    0    0    0    5    1    29
6    0    0    0    10    1    30
5    0    0    0    3    1    28
1    0    0    0    9    1    28
1    9    0    0    0    1    28
5    0    0    8    0    1    30
1    5    0    0    0    1    32
4    0    1    0    0    1    32
2    0    6    0    0    1    32
0    0    0    0    0    0
```

```
J301_10
32    4
11    12    9    9
0    0    0    0    0    3    2    3    4
4    1    0    0    0    3    5    7    20
3    0    0    0    2    1    22
10    0    0    0    4    1    18
10    0    0    1    0    3    6    9    10
5    0    10    0    0    1    30
1    0    3    0    0    3    8    11    21
7    3    0    0    0    1    30
4    0    0    0    1    3    13    14    16
5    0    0    0    6    1    19
7    0    0    0    6    1    12
10    0    3    0    0    2    17    25
2    0    7    0    0    2    21    23
4    0    0    2    0    2    15    27
5    0    5    0    0    1    28
5    0    0    0    1    1    29
4    0    0    0    4    1    24
3    0    0    7    0    2    19    23
7    0    0    0    3    1    31
6    8    0    0    0    1    23
2    0    0    2    0    1    24
5    0    0    8    0    2    26    27
8    10    0    0    0    2    26    31
3    0    0    7    0    1    28
3    5    0    0    0    2    27    28
2    8    0    0    0    1    30
10    0    0    5    0    1    31
3    0    0    0    9    1    29
5    8    0    0    0    1    32
5    7    0    0    0    1    32
1    5    0    0    0    1    32
0    0    0    0    0    0
```