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

Χρονοπρογραμματισμός Έργων με χρήση Γενετικών Αλγορίθμων

Μελέτη χρονοπρογραμματισμού έργων υπό περιορισμένους πόρους και διακριτή σχέση χρόνου-κόστους

Εκπόνηση: Εμμανουήλ Καλοειδάς

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

9/18/2012

Ευχαριστίες

Η περάτωση της παρούσης διπλωματικής εργασίας σηματοδοτεί το τέλος των σπουδών μου στη σχολή των Μηχανολόγων Μηχανικών. Δράττομαι της ευκαιρίας να ευχαριστήσω τους συμφοιτητές και τους φίλους μου που μου στάθηκαν κατά τη διάρκεια των σπουδών μου αλλά και τους καθηγητές μου, οι οποίοι πέραν από τις τεχνικές γνώσεις που μου παρείχαν, με βοήθησαν να αναπτύξω τον τρόπο σκέψης μου. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κύριο Κωνσταντίνο Κηρυττόπουλο με τον οποίο είχα άριστη συνεργασία και βοήθεια όποτε χρειαζόμουν το οτιδήποτε. Επίσης την κυρία Ρόκου Έλενα για την καθοδήγηση, την υπομονή και εμπιστοσύνη που μου έδειξε κατά τη διάρκεια της εργασίας και που δίχως την βοήθειά της η ολοκλήρωση της θα ήταν αδύνατη.

Τέλος, και πάνω από όλα θα ήθελα να ευχαριστήσω την οικογένειά μου για όλα όσα μου έχει προσφέρει αυτά τα χρόνια και για την ψυχολογική υποστήριξη που μου παρέχει.

ΠΕΡΙΕΧΟΜΕΝΑ

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

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.

Εμμανουήλ Καλοειδάς

Έποψη

Ο προγραμματισμός έργων είναι μία σημαντική διαδικασία στη διαχείριση (project management) των έργων. Ένα έργο αποτελείται από μία ομάδα δραστηριοτήτων. Οι δραστηριότητες έχουν σχέσεις προτεραιότητας, εκτιμώμενη διάρκεια και μπορεί να έχουν και άλλες μεταβλητές, όπως το κόστος. Κάθε δραστηριότητα απαιτεί συγκεκριμένη ποσότητα πόρων για την ολοκλήρωσή της και η συνολική ποσότητα των πόρων είναι περιορισμένη. Επίσης κάθε δραστηριότητα μπορεί να εκτελεσθεί με πολλαπλούς τρόπους, δηλαδή με συγκεκριμένη διάρκεια εκτέλεσης και ποσότητα πόρων. Η σχέση χρόνου-κόστους σε μία δραστηριότητα περιγράφει τη μεταβολή της διάρκειας περάτωσης της δραστηριότητας όταν σε αυτή μεταβληθεί: α) η ποσότητα ενός μη ανανεώσιμου πόρου (π.χ. κόστος), β) ενός ή περισσοτέρων ανανεώσιμων πόρων (π.χ. πλήθος απασχολούμενων εργατών) ή γ) το είδος (διαφορετική απόδοση) και η ποσότητα των απασχολούμενων ανανεώσιμων πόρων.

Η σχέση κόστους-χρόνου στα δίκτυα των έργων είναι αντικείμενο εκτενούς έρευνας από τότε που αναπτύχθηκε η μέθοδος κρισίμου δρόμου (CPM) στα τέλη της δεκαετίας του 50’. Στη διακριτή μορφή του προβλήματος, είναι γενικά αποδεκτό ότι η σχέση αυτή αναπαριστάται από μία γνησίως φθίνουσα συνάρτηση, δηλαδή, η επίσπευση μιας δραστηριότητας είναι δυνατή με τη χορήγηση περισσότερων ή καλύτερης απόδοσης πόρων με συνέπεια μεγαλύτερο κόστος.

Μοντελοποιήθηκε το παραπάνω προβλήματος και στη συνέχεια έγινε η επιλογή και η προσαρμογή στο πρόβλημα ενός κατάλληλου εξελικτικού αλγορίθμου για τη βελτιστοποίηση της σχέσης κόστους-χρόνου ενός έργου. Επίσης ο αλγόριθμος τροποποιήθηκε έτσι ώστε να ενσωματώνει δεδομένα από πραγματικά έργα και να δίνει λύσεις σχετικά με το χρονικό προγραμματισμό. Ο λειτουργία του αλγορίθμου μπορεί να παρουσιαστεί ως ένα μαύρο κουτί που δέχεται ως εισόδους τις πληροφορίες ενός έργου και έχει ως έξοδο το χρονικό του προγραμματισμό. Για τον έλεγχο της καλής λειτουργίας του αλγορίθμου χρησιμοποιήθηκε ένα έργο κτηματογράφησης χρησιμοποιείται ως ένα πρακτικό παράδειγμα.

1 Εισαγωγή

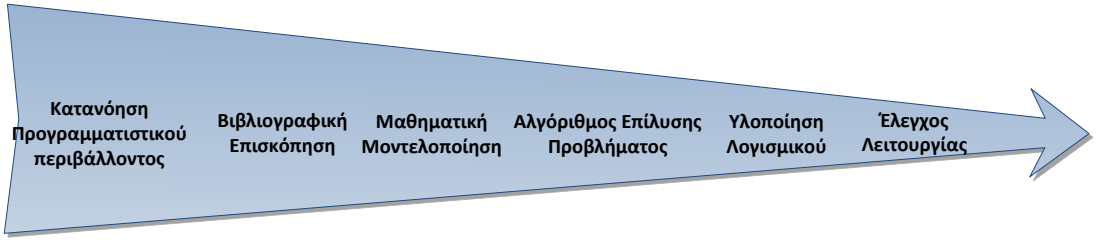
Η παρούσα διπλωματική εργασία έχει ως θέμα τον χρονικό προγραμματισμό έργων υπό περιορισμένους πόρους. Αυτή η κατηγορία προβλημάτων καθορίστηκε από τον Davis το 1973 ως «*η μέθοδος προγραμματισμού μιας σειράς διεργασιών με περιορισμένο ανά χρονική περίοδο αριθμών πόρων κατά τη διάρκεια εκτέλεσης του έργου με στόχο να ελαχιστοποιηθεί η διάρκειά του» (1973)* και το συγκεκριμένο θέμα αναφέρεται στη βιβλιογραφία ως Resource-Constrained Project Scheduling Problem, ή σύντομα RCPSP. Το πρόβλημα αυτού του τύπου έχει αποδειχθεί ότι αποτελεί πρόβλημα τύπου NP-Hard. Καθώς το πρόβλημα αυτό είναι ερευνηθεί εκτενώς, επεκταθήκαμε στη μελέτη του προβλήματος προγραμματισμού δραστηριοτήτων υπό περιορισμένους πόρους και πολλαπλούς τρόπους εκτέλεσης (Multi-Mode Resource-Constrained Project Scheduling Problem-MRCPSP). Το πρόβλημα αυτό αποτελεί γενικευμένη εκδοχή του γενικού προβλήματος RCPSP, όπου κάθε δραστηριότητα μπορεί να εκτελεσθεί με έναν από τους πολλούς τρόπους, δηλαδή με έναν από τους πολλούς συνδυασμούς διάρκειας και απαιτούμενων πόρων. Συγκεκριμένα εμβαθύναμε στη μελέτη της σχέσης κόστους - χρόνου σε διακριτή μορφή σε συνδυασμό με το παραπάνω πρόβλημα. Η διακριτή σχέση χρόνου-κόστους αναφέρεται στην βιβλιογραφία ως *Discrete Time-Cost Trade Off Resource Constrained Project Scheduling Problem* (DTCTP). Στο πρόβλημα αυτό η διάρκεια εκτέλεσης των δραστηριοτήτων είναι μία διακριτή, γνησίως φθίνουσα συνάρτηση του μοναδικού μη ανανεώσιμου πόρου που τους αντιστοιχεί. Το πρόβλημα αυτό ερευνήθηκε πρώτα από τους Hindelang και Muth. Για την επίλυση αυτών των προβλημάτων χρησιμοποιήθηκε η μεθοδολογία των γενετικών αλγορίθμων, οι οποίοι προσαρμόζονται πολύ καλά σε προβλήματα ανάθεσης δραστηριοτήτων με συγκεκριμένη σειρά. Οι Γενετικοί Αλγόριθμοι (Genetic Algorithms) ανήκουν στην κατηγορία συστημάτων επίλυσης προβλημάτων που είναι ευρύτερα γνωστή με τον όρο Εξελικτικοί Αλγόριθμοι (Evolutionary Algorithms). Ο καθηγητής John Holland θεμελίωσε τους Γενετικούς Αλγορίθμους ως μεθόδους βελτιστοποίησης στις αρχές του 1970 με τους συνεργάτες του στο Πανεπιστήμιο του Michigan.

Παρακάτω θα παρουσιαστούν τα κεφάλαια της διπλωματικής και μια σύντομη περιγραφή τους. Το 2^{ο} κεφάλαιο αναφέρεται στη μέθοδο της έρευνας. Το τρίτο κεφάλαιο της εργασίας εστιάζει στη συγκέντρωση των βασικών ορισμών για τη διαχείριση έργων, των ορισμών των προβλημάτων και των μαθηματικών μοντελοποιήσεων τους και τις διαφόρους μεθόδους επίλυσης των προβλημάτων

αυτών. Στο επόμενο κεφάλαιο παρουσιάζει τους αλγόριθμους που αναπτύχθηκαν και το πρόβλημα που λύνουν. Στο πέμπτο κεφάλαιο παρουσιάζεται ο αλγόριθμος επίλυσης του προβλήματος και τα βασικά του χαρακτηριστικά. Παρουσιάζεται το χρωμόσωμα, η μέθοδος για την παραγωγή του αρχικού πληθυσμού και οι βασικοί κανόνες που την διέπουν. Στο έκτο κεφάλαιο, γίνεται ο σχεδιασμός του συστήματος. Το πρώτο υποκεφάλαιο αφορά την αντικειμενική συνάρτηση και πως καταστρώθηκε, ενώ στο δεύτερο γίνεται εκτενής αναφορά στο γενετικό αλγόριθμο και σε όλα τα χαρακτηριστικά του. Το επόμενο κεφάλαιο αφορά την ανάλυση του συστήματος, όπου παρουσιάζεται ο τρόπος που λειτουργεί ο αλγόριθμός, τα δεδομένα που δέχεται ως εισόδους καθώς και τα αποτελέσματα που έχει ως εξόδους. Γίνεται σύγκριση με αποτελέσματα από τη βιβλιογραφία. Στο όγδοο κεφάλαιο ασχολείται με την πρακτική εφαρμογή του αλγορίθμου σε πραγματικά προβλήματα προγραμματισμού έργων. Ο σχεδιασμός της διεπαφής του αλγορίθμου και του προγράμματος Microsoft Project αναλύεται, όπως και τα βήματα για την εφαρμογή του. Στο τέλος του κεφαλαίου αναλύονται τα αποτελέσματα από τη χρήση του αλγορίθμου σε ένα πραγματικό πρόβλημα κτηματογράφησης και συγκρίνονται με τα αποτελέσματα από την εφαρμογή του στο Microsoft Project. Ολοκληρώνοντας, θα παρουσιαστούν τα συνολικά συμπεράσματα της έρευνας, η αξιολόγηση του προτεινόμενου αλγορίθμου και θα δοθούν κατευθύνσεις για περαιτέρω μελέτη.

2 Μέθοδος Έρευνας

Τα βήματα εργασίας τα οποία ακολουθήθηκαν για την εκπόνηση της εργασίας παρουσιάζονται στην εικόνα 1. Αρχικά κατανοήθηκε το προγραμματιστικό περιβάλλον. Εφόσον αποκτήθηκε μία σχετική γνώση με αυτό, το επόμενο βήμα ήταν η βιβλιογραφική επισκόπηση του θέματος της διπλωματικής. Συγκεκριμένα εντοπίστηκε το υπό μελέτη πρόβλημα, οι προτεινόμενες λύσεις στη βιβλιογραφία, καθώς και η συγκριτική μελέτη υφιστάμενων τρόπων μοντελοποίησης και αλγορίθμων επίλυσης. Έπειτα μοντελοποιήθηκε το πρόβλημα στη μαθηματική του μορφή και σχεδιάστηκε ο αλγόριθμος επίλυσης του προβλήματος. Τα τελευταία βήματα ήταν η ανάλυση του λογισμικού και ο έλεγχος λειτουργίας του.



ΕΙΚΟΝΑ 1 ΒΗΜΑΤΑ ΈΡΓΑΣΙΑΣ

3 Βιβλιογραφική Επισκόπηση

3.1 Προγραμματισμός & Διοίκηση Έργων

3.1.1 Έργο

Η έννοια του έργου στη διαχείριση έργων είναι πολύ σαφώς καθορισμένη. Σύμφωνα με το PMBOK (PMI, 2008), ο ορισμός του έργου είναι ο εξής:

«*Το έργο αποτελεί ένα εγχείρημα, το οποίο έχει πεπερασμένη αρχή και τέλος. Το έργο τελιώνει όταν έχουν επιτευχθεί οι στόχοι που έχουν τεθεί ή όταν κρίνεται ότι δεν μπορούν να επιτευχθούν πλέον. Ο σκοπός του έργου είναι η δημιουργία ενός μοναδικού προϊόντος, μιας υπηρεσίας ή ενός αποτελέσματος. Ένα έργο έχει προσωρινό χαρακτήρα δίχως αυτό να σημαίνει ότι έχει μικρή χρονική διάρκεια. Το προϊόν, η υπηρεσία ή το αποτέλεσμα παράγονται από το έργο με σκοπό να διαρκέσουν πολύ περισσότερο από την διάρκεια του έργου.*

Κάθε έργο χαρακτηρίζεται από τη μοναδικότητά του. Το προϊόν, η υπηρεσία ή το αποτέλεσμα που δημιουργούνται από το έργο μπορεί να είναι επαναλαμβανόμενα αλλά αυτή η επανάληψη δεν αλλάζει τη θεμελιώδη μοναδικότητα των εργασιών του έργου. Ένα έργο δεν έχει εκτελεσθεί ποτέ στο παρελθόν και δεν θα εκτελεσθεί ποτέ ξανά στο μέλλον. Κατά την εξέλιξη των εργασιών για την περάτωση του έργου θα συμβούν απρόσμενα και τυχαία γεγονότα τα οποία θα το επηρεάσουν».

Από τα παραπάνω προκύπτει ότι ένα έργο έχει τα εξής χαρακτηριστικά:

- Είναι μοναδικό
- Έχει ως σκοπό την εκπλήρωση στόχων
- Έχει αρχή και τέλος
- Έχει προσωρινή φύση
- Περιορίζεται οικονομικά

Ένα έργο μπορεί να δημιουργήσει:

- Ένα αυτοτελές προϊόν ή ένα στοιχείο που να εμπεριέχεται σε ένα άλλο προϊόν.
- Δυνατότητες για να εκτελείται μια υπηρεσία (π.χ., μια επιχειρηματική λειτουργία που υποστηρίζει την παραγωγή ή τη διανομή), ή

- Ένα αποτέλεσμα, όπως ένα έγγραφο ή ένα συμπέρασμα

Το κύριο χαρακτηριστικό των έργων είναι το ότι έχουν συγκεκριμένη διάρκεια. Αυτή η συγκεκριμένη διάρκεια αποτελεί τον κύκλο ζωής του έργου (Project Life Cycle). Ο Κύκλος Ζωής Έργου (Project Life Cycle) αναφέρεται σε μία λογική ακολουθία φάσεων για την επίτευξη των σκοπών ή στόχων του έργου. Ανεξάρτητα από το αντικείμενο ή την πολυπλοκότητά του, κάθε έργο διέρχεται από μία σειρά φάσεων κατά τη διάρκεια της ζωής του. Τυπικά ο Κύκλος Ζωής Έργου αποτελείται από τέσσερις βασικές φάσεις, όπως παρουσιάζονται στην Εικόνα 2.

ΕΙΚΟΝΑ 2 ΚΥΚΛΟΣ ΖΩΗΣ ΕΡΓΟΥ *(PROJECT MANAGEMENT INSTITUTE, 2008)*

Το πρώτο στάδιο, δηλαδή η σύλληψη της ιδέας, είναι αποτέλεσμα που προκύπτει για την ικανοποίηση μιας ανάγκης. Στη συνέχεια γίνεται ο σχεδιασμός του έργου όπου μελετούνται τα χαρακτηριστικά του έργου, όπως τα οικονομικά, κοινωνικά, περιβαλλοντολογικά στοιχεία του. Σε αυτό το στάδιο γίνονται και οι απαραίτητες μελέτες όπως η κοστολόγηση, η ανάλυση διαδικασιών, χρονοδιαγράμματα και τεχνικές μελέτες. Κατά την υλοποίηση του έργου γίνεται η παρακολούθηση του έργου και ακολουθείται ο σχεδιασμός που έχει γίνει νωρίτερα. Το τελευταίο στάδιο αφορά την διαδικαστική πλευρά της διαχείρισης του έργου. Σε αυτό γίνεται ο ποιοτικός έλεγχος, μετρήσεις για την αποτελεσματικότητα των διαδικασιών και η προσαρμογή του έργου στα δεδομένα του πελάτη.

Τα προβλήματα προγραμματισμού και οργάνωσης ενός έργου έχουν να κάνουν με το μεγάλο πλήθος των επιμέρους δραστηριοτήτων, από την εκτέλεση και τη πολυπλοκότητα των οποίων εξαρτάται η ολοκλήρωση του ίδιου του έργου. Οι δραστηριότητες αυτές συνδέονται μεταξύ τους με τεχνολογικές, φυσικές, οικονομικές ή άλλες σχέσεις προτεραιότητας, οι οποίες εκφράζονται με συσχετίσεις μεταξύ των δραστηριοτήτων του τύπου «τέλους-αρχής», «αρχής-αρχής», «αρχής-τέλους» και «τέλους-τέλους», ενώ υπόκεινται σε διάφορους περιορισμούς, π.χ. λόγω διαθέσιμων πόρων ή υπάρχοντος θεσμικού πλαισίου, που πρέπει να ληφθούν υπόψη κατά τον προγραμματισμό τους. Τα παραπάνω προβλήματα αποκτούν ιδιαίτερη σημασία όταν συναντώνται σε έργα μεγάλης κλίμακας, κόστους κατασκευής, ιδιάζοντος ρόλου στην οικονομική και κοινωνική ζωή (όπως ένα λιμάνι ή ένα φράγμα) κλπ. Το ζητούμενο σε αυτές τις περιπτώσεις μπορεί να είναι η ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης του έργου, του συνολικού κόστους, του κόστους για ένα δεδομένο

συνολικό χρόνο ολοκλήρωσης, του χρόνου εκτέλεσης για ένα δεδομένο συνολικό κόστος, των πόρων που παραμένουν σε αδράνεια κ.α.

3.1.2 Διοίκηση Έργου

«*Η διοίκηση του έργου είναι η εφαρμογή της γνώσης, των δεξιοτήτων, των εργαλείων, και των διαδικασιών στις δραστηριότητές του, με σκοπό να επιτευχθούν οι στόχοι του. (Project Management Institute, 2008)*

Η διοίκηση έργου περιλαμβάνει τον σχεδιασμό, τον προγραμματισμό και τον έλεγχο των δραστηριοτήτων του έργου για την επίτευξη της απόδοσης, του κόστους και του χρόνου που έχουν τεθεί ως στόχος, χρησιμοποιώντας τους πόρους όσο το δυνατόν πιο αποδοτικά και αποτελεσματικά» (Project Management Institute, 2008).

Ο Σχεδιασμός περιλαμβάνει την ανάλυση του έργου σε δραστηριότητες, σε φάσεις, καθώς και την κοστολόγηση και τον χρονικό προγραμματισμό. Είναι μία από τις πιο κρίσιμες λειτουργίες για την καλή έκβαση του έργου. Αποτελείται από ενέργειες όπως η πρόβλεψη συντελεστών, παραγόντων (περιβαλλοντικών, οικονομικών, κοινωνικών), κινδύνων καθώς και των μεταβλητών που αφορούν το έργο.

Ο Προγραμματισμός στοχεύει στην ολοκλήρωση των δραστηριοτήτων του έργου στις ημερομηνίες που έχουν τεθεί κατά τον σχεδιασμό, όπως επίσης και στον υπολογισμό των ποσοτήτων των πόρων που απαιτούνται για την πραγματοποίηση των δραστηριοτήτων.

Ο Έλεγχος και συντονισμός είναι η φάση της διαχείρισης του έργου όπου παρουσιάζονται οι αποκλίσεις από τον αρχικό σχεδιασμό και γίνονται οι απαιτούμενες ενέργειες για την διόρθωση των παραγόντων που ευθύνονται έτσι ώστε να επιτευχθεί ο σκοπός. Ο συντονισμός είναι η λειτουργία που αφορά τη διευκόλυνση της ροής των δραστηριοτήτων, στην επικοινωνία μεταξύ των εργαζομένων του έργου αλλά και στην επίλυση προβλημάτων που μπορεί να προκύψουν από αντικρουόμενα συμφέροντα.

3.1.3 Χρονικός προγραμματισμός Έργων

Ο όρος «χρονοπρογραμματισμός» έχει ορισθεί ως *«…η κατανομή δεδομένων πόρων στη διάρκεια του χρόνου με σκοπό την ολοκλήρωση ενός συνόλου εργασιών*» (Baker,1974).

Ο χρονικός προγραμματισμός είναι μία διαδικασία που έχει σημαντικό ρόλο στις βιομηχανίες κατασκευών αλλά και σε αυτές της παροχής υπηρεσιών. Στο

σημερινό ανταγωνιστικό περιβάλλον της αγοράς, ο αποτελεσματικός προγραμματισμός εργασιών είναι σημαντικός παράγοντας στην επιβίωση μιας επιχείρησης. Οι εταιρείες πρέπει να ανταποκριθούν σε πολλές διορίες που τίθενται από τους πελάτες τους, αλλιώς θα χάσουν την αξιοπιστία τους. Ταυτόχρονα θα πρέπει να προγραμματίσουν τις ενέργειές τους έτσι ώστε να χρησιμοποιούν τους πόρους τους με αποτελεσματικό τρόπο.

Ο χρονικός προγραμματισμός περιλαμβάνει τις διαδικασίες εκείνες που απαιτούνται για να διασφαλιστεί η ολοκλήρωση του έργου στον προαποφασισμένο χρόνο, με το αρχικά ορισμένο κόστος και στην ζητούμενη ποιότητα (Project Management Institute, 2008). Οι διαδικασίες αυτές συνοψίζονται ως εξής:

- Καθορισμός των δραστηριοτήτων που απαρτίζουν το έργο.
- Σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων
- Καθορισμός των απαιτούμενων πόρων για την επίτευξη των στόχων
- Εκτίμηση της διάρκειας των δραστηριοτήτων του έργου
- Σχεδιασμός χρονοπρογράμματος
- Έλεγχος του έργου

Επομένως, κατά το χρονικό προγραμματισμό καθορίζεται η χρονική αλληλουχία των επί μέρους δραστηριοτήτων, η χρονική κατανομή του παραγωγικού δυναμικού και η ροή των υλικών που θα χρησιμοποιηθούν στο έργο. Ο χρονικός προγραμματισμός αφορά την δημιουργία χρονοπινάκων (ημερομηνίες έναρξης, ημερομηνίες ολοκλήρωσης, διάρκειες δραστηριοτήτων). Τα χρονοπρογράμματα είναι βασικά εργαλεία εργασίας για τον προγραμματισμό, την αξιολόγηση και τον έλεγχο του προγράμματος. Το σύνολο των χρονοπινάκων αποτελεί το πρόγραμμα του έργου.

3.2 Δομικά Στοιχεία Προβλήματος Προγραμματισμού Έργων

3.2.1 Πόροι

Για την εκτέλεση των δραστηριοτήτων απαιτούνται πόροι. Στον προγραμματισμό του έργου λαμβάνονται αποφάσεις σχετικά με το τι πόροι θα χρησιμοποιηθούν, ποιά θα είναι οι ποσότητα αυτών των πόρων για το συνολικό έργο, καθώς και την ποσότητα που απαιτεί κάθε δραστηριότητα για να εκτελεσθεί μεμονωμένα.

Οι πόροι μπορούν να ταξινομηθούν με διάφορους τρόπους. Σύμφωνα με τον Blazewicz (1986), μπορούν να διακριθούν τρεις κύριες κατηγορίες πόρων, κάθε μία από τις οποίες μπορεί να περιέχει διαφορετικού τύπου πόρων (π.χ. εργατικό δυναμικό, μηχανές), ως εξής:

- Ανανεώσιμοι πόροι
- Μη ανανεώσιμοι πόροι
- Διπλά περιορισμένοι πόροι (doubly-constrained resources)

Οι ανανεώσιμοι πόροι είναι διαθέσιμοι στο ίδιο επίπεδο σε κάθε χρονική περίοδο (π.χ. σταθερό εργατικό δυναμικό). Ο περιορισμός τίθεται μόνο στην ανώτατη ποσότητα που μπορεί να διατεθεί σε κάθε περίοδο.

Η δεύτερη κατηγορία περιλαμβάνει πόρους που διατίθενται στην αρχή του έργου και αναλώνονται με την πάροδο του χρόνου. Το χρήματα είναι το καλύτερο παράδειγμα αυτής της κατηγορίας πόρων καθώς το συνολικό κόστος του έργου συχνά περιορίζεται σε ένα προκαθορισμένο ποσό. Παραδείγματα μη ανανεώσιμων πόρων είναι επίσης οι πρώτες ύλες και η ενέργεια.

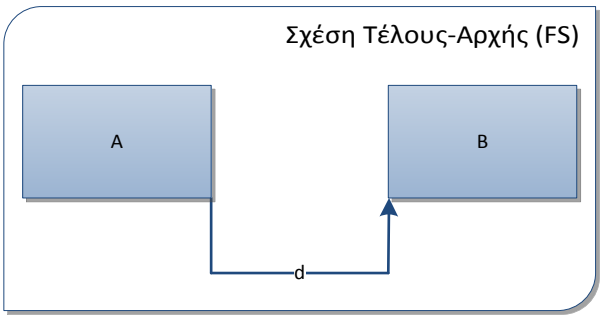
Στην τρίτη κατηγορία οι πόροι είναι διαθέσιμοι σε περιορισμένες ποσότητες σε κάθε περίοδο. Ωστόσο, η συνολική διαθεσιμότητά τους στη συνολική διάρκεια του έργου είναι επίσης περιορισμένη. Το κεφάλαιο με περιορισμένη ποσότητα ταμειακών εισροών ανά περίοδο και περιορισμένο συνολικό ποσό είναι ένα τυπικό παράδειγμα, όπως επίσης και οι διαθέσιμες ανθρωποώρες ανά ημέρα σε συνδυασμό με περιορισμένη διαθεσιμότητα ανθρωποωρών για όλο το έργο είναι ένα άλλο παράδειγμα.

3.2.2 Σχέσεις εξάρτησης-αλληλουχίας μεταξύ των δραστηριοτήτων (Generalized Precedence Relations)

Μεταξύ των δραστηριοτήτων που απαιτούνται για την υλοποίηση του έργου αναπτύσσονται τέσσερα είδη σχέσεων προτεραιότητας. Κάθε μία από αυτές συνδέεται με μία άλλη ορίζοντας ως *Lag* την χρονική καθυστέρηση που ισχύει.

1. Σχέση Τέλους – Αρχής(Finish to Start, FS):

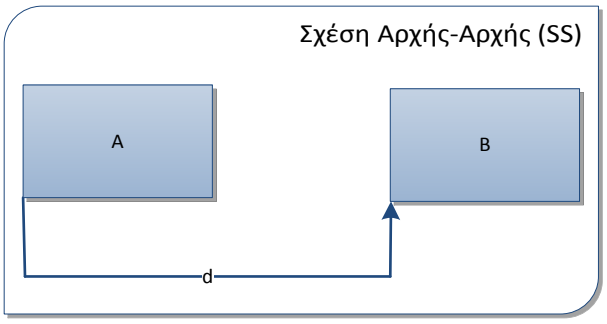
   Αποτελεί τη συνηθέστερη σχέση που εμφανίζεται στη διαχείριση έργων. Για την εκκίνηση της δραστηριότητας που έπεται (B) απαιτείται η ολοκλήρωση της προηγούμενης (A). Η αρχή της B, δηλαδή, καθορίζεται από το τέλος της A. Όταν η καθυστέρηση μεταξύ τους (d) είναι 0, η σχέση αντιστοιχεί στην σχέση προηγείται – έπεται του δικτύου C.P.M.



ΕΙΚΟΝΑ 3 ΣΧΕΣΗ ΤΕΛΟΥΣ – ΑΡΧΗΣ

2. Σχέση Αρχής – Αρχής (Start to Start, SS):

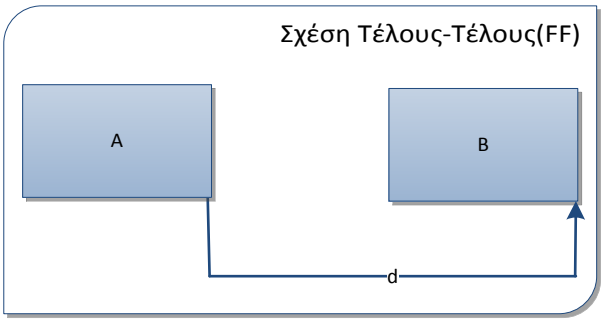
   Δύο δραστηριότητες (A) και (B) συνδέονται με σχέση αρχής – αρχής όταν και μόνο όταν η δραστηριότητα (B) μπορεί να ξεκινήσει d χρονικές μονάδες μετά την έναρξη της (A). Η λογική πίσω από την επιλογή αυτής της σχέσης είναι πως οι απαιτήσεις και οι περιορισμοί για την εκκίνηση μιας δραστηριότητας πληρούνται πριν την ολοκλήρωση του συνόλου της προκατόχου της.



ΕΙΚΟΝΑ 4 ΣΧΕΣΗ ΑΡΧΗΣ – ΑΡΧΗΣ

3. Σχέση Τέλους – Τέλους (Finish to Finish, FF):

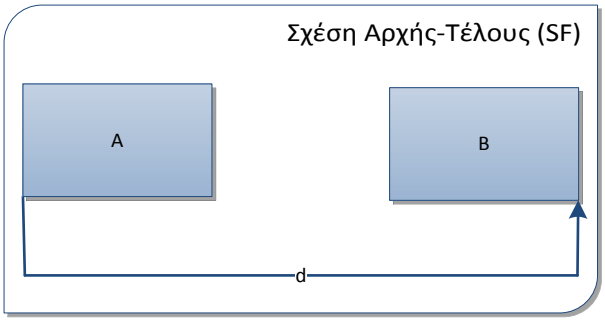
Σύμφωνα με αυτή τη σχέση, η ολοκλήρωση της διαδόχου δραστηριότητας (B) προϋποθέτει την ολοκλήρωση της προκατόχου της (A). Κατά αυτόν τον τρόπο επιτρέπεται η ταυτόχρονη εκτέλεση των δύο δραστηριοτήτων ή ακόμα και η εκκίνηση της δραστηριότητας πριν από αυτή της προκατόχου, όσο δεν παραβιάζεται η συνθήκη για τη λήξη τους. Αυτό συμβαίνει μόνο και μόνο όταν η (B) μπορεί να τελειώσει d χρονικές μονάδες μετά την ολοκλήρωση της (A).



ΕΙΚΟΝΑ 5 ΣΧΕΣΗ ΤΕΛΟΥΣ – ΤΕΛΟΥΣ

4. Σχέση Αρχής – Τέλους (Start to Finish, SF):

Δύο δραστηριότητες (A) και (B) συνδέονται με σχέση αρχής – τέλους όταν και μόνο όταν η δραστηριότητα (B) μπορεί να τελειώσει d χρονικές μονάδες μετά την έναρξη της (A). Εδώ, η λήξη της διαδόχου (B) καθορίζεται από την έναρξη της προκατόχου (A).



ΕΙΚΟΝΑ 6 ΣΧΕΣΗ ΑΡΧΗΣ – ΤΕΛΟΥΣ

3.2.3 Είδη Χρονοπρογραμμάτων

Πλήρες πρόγραμμα (complete schedule) ονομάζεται ένα πρόγραμμα στο οποίο έχουν καθοριστεί οι χρόνοι έναρξης όλων των εργασιών.

Εφικτό πρόγραμμα (feasible schedule) ονομάζεται ένα πρόγραμμα στο οποίο έχουν καθοριστεί οι χρόνοι έναρξης (ή οι χρόνοι ολοκλήρωσης) των δραστηριοτήτων που το απαρτίζουν και τηρούνται όλοι οι κανόνες προτεραιότητας και όλοι οι περιορισμοί για τους πόρους.

Εάν σε ένα πρόγραμμα μπορούμε να μεταθέσουμε μια δραστηριότητα έτσι ώστε να ξεκινήσει νωρίτερα και αν το πρόγραμμα που προκύπτει είναι εφικτό, τότε ονομάζουμε την διαδικασία αυτή *τοπική μετατόπιση αριστερά* (local left shift). Εάν σε ένα πρόγραμμα δεν μπορεί να γίνει η παραπάνω διαδικασία τότε ονομάζεται **ημι-ενεργό πρόγραμμα** (semi-active schedule).

Εάν είναι δυνατή η μετατόπιση προς τα αριστερά μιας δραστηριότητας αλλά παραβιάζοντας τους περιορισμούς των πόρων τότε η διαδικασία αυτή ονομάζεται *γενική αριστερή μετατόπιση* (global left shift).

Τα προγράμματα στα οποία δεν μπορούν να εφαρμοστούν ούτε γενικές ούτε τοπικές αριστερές μετατοπίσεις ονομάζονται **ενεργά προγράμματα** (active schedules).

Πρόγραμμα μη-καθυστέρησης (non-delay schedule) είναι κάθε εφικτό πρόγραμμα στο οποίο δεν υπάρχει χρονική περίοδος κατά την οποία μία δραστηριότητα που ανήκει σε αυτήν και τηρεί όλους τους περιορισμούς, δεν έχει προγραμματιστεί. Υπάρχει περίπτωση σε κάποιο πρόγραμμα μη-καθυστέρησης να μην υπάρχει βέλτιστη λύση.

3.2.4 Συμπίεση μιας Δραστηριότητας (Crashing)

Η συμπίεση δεν έχει νόημα για μια δραστηριότητα για την οποία δεν μπορούμε να επηρεάσουμε τη χρονική της εξέλιξη, όπως για παράδειγμα η διαδικασία πήξης του σκυροδέματος. Στις πιο πολλές περιπτώσεις όμως, ο χρόνος που απαιτείται για μια εργασία θα εξαρτάται από την ένταση με την οποία εκτελείται αυτή. Αν δηλαδή σε μια δραστηριότητα ασχοληθούν περισσότερα άτομα ή με πιο μεγάλη ένταση ή με περισσότερα εργαλεία, τότε θα επηρεαστεί η διάρκεια της δραστηριότητας. Οι βασικές μορφές των διαρκειών των δραστηριοτήτων είναι οι **Κανονικές (normal)** και οι **Συμπιεσμένες (crashed).**

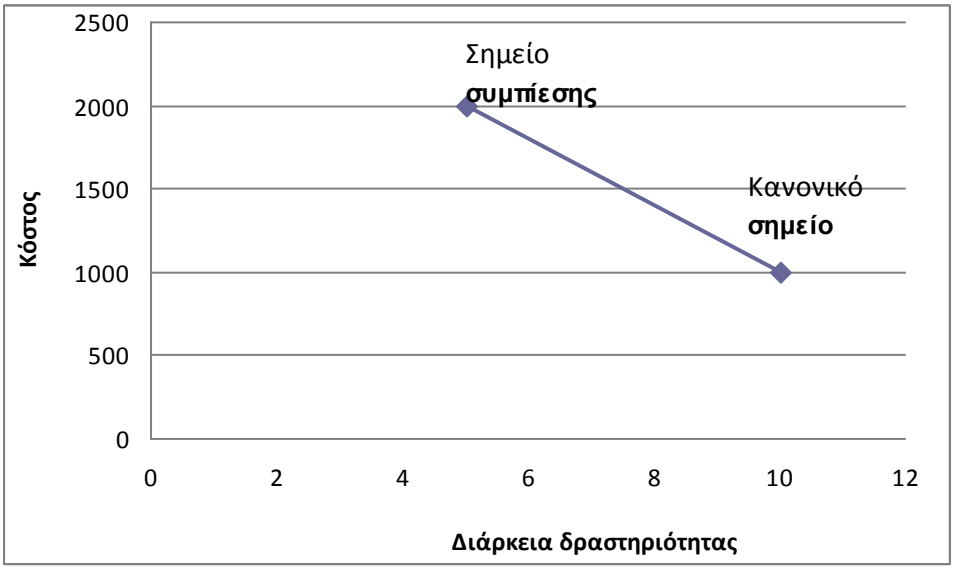
Η συμπίεση είναι η διαδικασία βάσει της οποίας η διάρκεια μιας δραστηριότητας συντομεύεται με την προσθήκη πόρων και την καταβολή πρόσθετου άμεσου κόστους[1]. Ένα συμπιεσμένο πρόγραμμα περιλαμβάνει δραστηριότητες που εκτελούνται πιο γρήγορα με αποτέλεσμα να διατίθενται σε αυτό σε απόλυτο αριθμό ανά περίοδο περισσότεροι πόροι.

Η συμπίεση της διάρκειας μιας δραστηριότητας συνήθως επιφέρει αύξηση στο άμεσο κόστος της. Για να χειριστούμε σωστά αυτή την αναλογία θα πρέπει να είναι γνωστό το κόστος για κάθε δραστηριότητα και η διάρκεια που απαιτείται για την εκτέλεσή της και στις δύο μορφές. Ένα παράδειγμα ακολουθεί για την κατανόηση της συμπίεσης.

[1] Η διάκριση μεταξύ άμεσου και έμμεσου κόστους γίνεται στην παράγραφο 6.1

ΠΙΝΑΚΑΣ 1 ΣΧΕΣΗ ΧΡΟΝΟΥ-ΚΟΣΤΟΥΣ

Δραστηριότητα	Κόστος	Διάρκεια Εκτέλεσης (Απλή)	Συμπιεσμένη Διάρκεια(Crashed)
A	1000€	10	-
A	2000€	-	5



ΕΙΚΟΝΑ 7 ΧΑΡΑΚΤΗΡΙΣΤΙΚΗ ΚΑΜΠΥΛΗ ΑΝΤΙΣΤΑΘΜΙΣΗΣ ΧΡΟΝΟΥ-ΚΟΣΤΟΥΣ

3.2.5 Σχέση Χρόνου-Κόστους στον Χρονοπρογραμματισμό

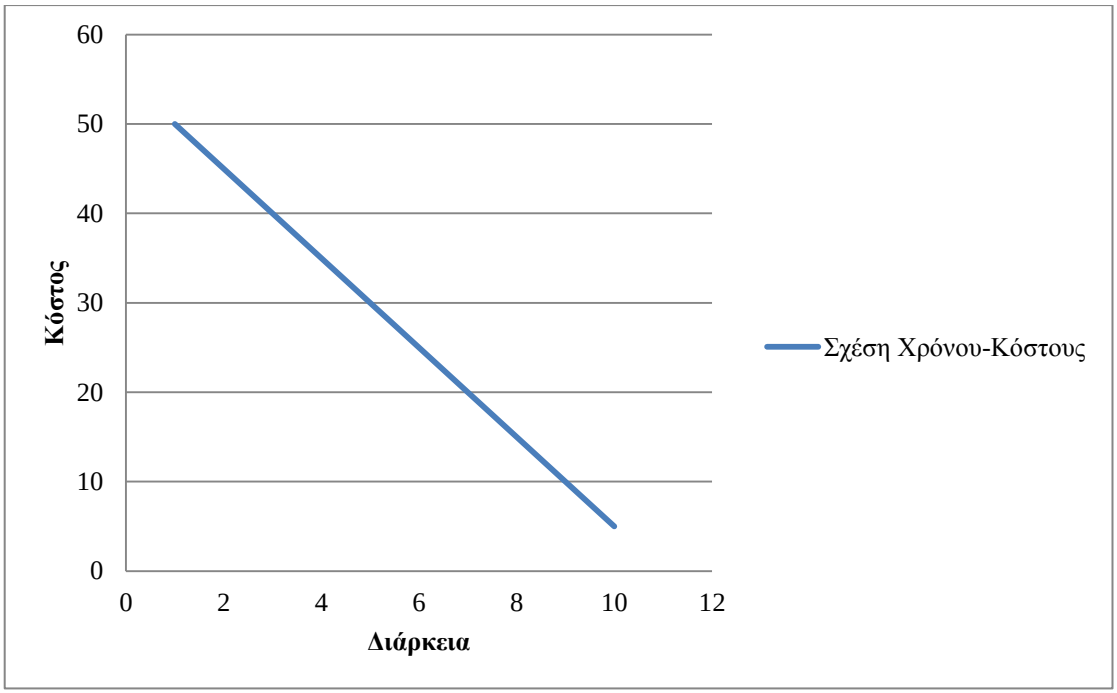
Οι ερευνητές της μεθόδου του κρίσιμου δρόμου (CPM) ήταν οι πρώτοι που κατάλαβαν ότι στα προβλήματα προγραμματισμού έργων στην πραγματική ζωή, οι δραστηριότητες μπορούν να εκτελεσθούν γρηγορότερα ή πιο αργά ανάλογα με την ποσότητα των πόρων που διατίθενται σε κάθε μία. Το κόστος είναι αυτό που αυξάνεται όταν η δραστηριότητα εκτελεσθεί γρηγορότερα.

Τα επιπλέον κόστη αντισταθμίζονται από τα κέρδη που προέρχονται από τη μείωση του χρόνου διάρκειας του έργου. Από την άλλη, εκτελώντας δραστηριότητες σε αργότερους ρυθμούς συνήθως επιτυγχάνεται μειωμένο κόστος που προέρχεται από την μείωση στην ποσότητα των πόρων που είναι απαραίτητοι για την εκτέλεσή της αλλά συγχρόνως αυξάνεται η συνολική διάρκεια του έργου και αυτό ισοδυναμεί αρκετές φορές στην πραγματική ζωή σε κόστη που προέρχονται από ποινές λόγω καθυστέρησης παράδοσης του έργου.

Δεδομένης της ημερομηνίας παράδοσης του έργου, ο στόχος είναι να αποφασισθούν οι διάρκειες των δραστηριοτήτων και η σειρά που θα εκτελεσθούν έτσι ώστε να ελαχιστοποιηθεί το συνολικό κόστος του έργου.

Το κόστος αυτό είναι το άθροισμα του άμεσου και του έμμεσου κόστους, το οποίο προέρχεται από τη διάρκεια του έργου. Το έμμεσο κόστος περιλαμβάνει επίβλεψη και άλλα συνήθη γενικά έξοδα, τους τόκους για τη σωρευτική επένδυση του έργου, ποινές για την ολοκλήρωση του έργου μετά από μια ορισμένη ημερομηνία, δώρα (bonus) για την έγκαιρη ολοκλήρωση του έργου.

Το πρόβλημα της σχέσης χρόνου-κόστους έχει ερευνηθεί προ δεκαετιών με την προϋπόθεση της συνεχής σχέσης τους. Η σχέση αυτή μπορεί να παρουσιασθεί ως μία συνάρτηση $y = f(x)$, όπου η ανεξάρτητη μεταβλητή x είναι η διάρκεια της δραστηριότητας και η εξαρτημένη y είναι το άμεσο (direct) κόστος. Η συνάρτηση f, αρχικά, θεωρήθηκε γραμμική.

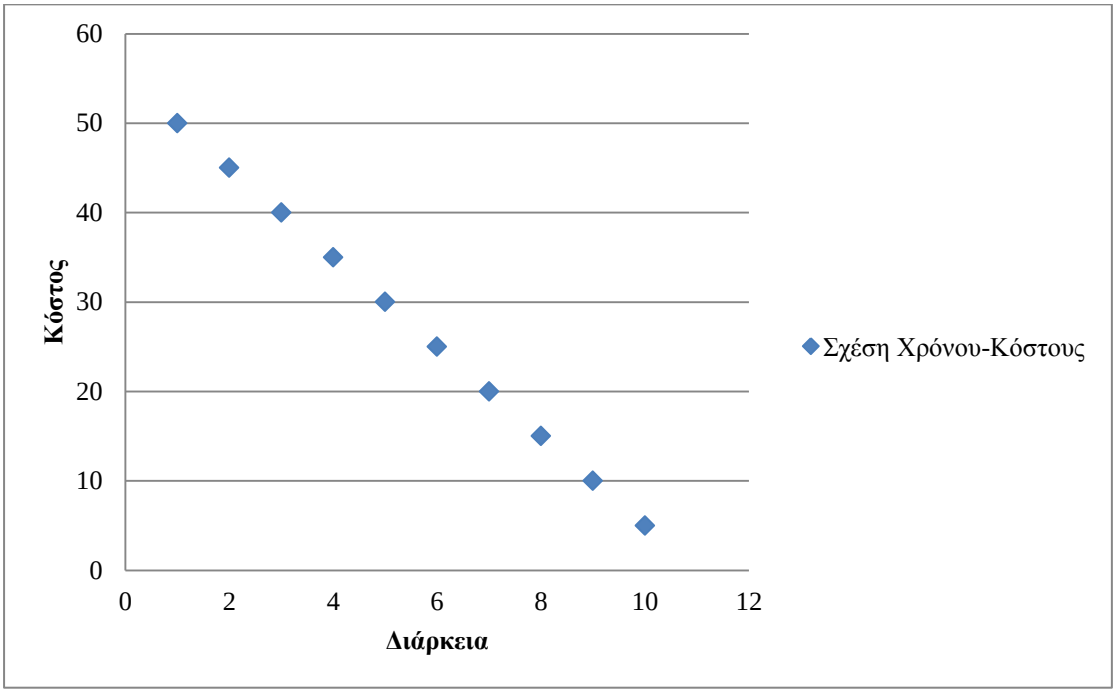


ΕΙΚΟΝΑ 8 ΣΥΝΕΧΗΣ ΣΧΕΣΗ ΧΡΟΝΟΥ-ΚΟΣΤΟΥΣ

Αντιθέτως, στη βιβλιογραφία η υπόθεση όπου είναι διακριτά τα σημεία με τα οποία μπορούν να εκτελεσθούν οι δραστηριότητες, είναι σχετικά πρόσφατη, παρά το γεγονός ότι η διακριτή μορφή της σχέσης χρόνου-κόστους θεωρείται πιο ρεαλιστικό μοντέλο προσέγγισης των πραγματικών έργων. Το πρόβλημα αυτό ερευνήθηκε παραπάνω όταν απεδείχθη ότι είναι τύπου NP-Hard. Αναλυτικές μέθοδοι(Hindelang and Muth, 1973), ευρετικές (Demeulemeester et al., 1996, Vanhoucke, 2005,

Erenguc et al., 2001)και μετά-ευρετικές έχουν προταθεί για διάφορες μεταβλητές του προβλήματος. Καμία από τις αναλυτικές μεθόδους δεν μπορεί να λύσει μεγάλα και δύσκολα προβλήματα, δηλαδή με πολλές δραστηριότητες και με πολλούς τρόπους εκτέλεσης σε κάθε δραστηριότητα.

Αρκετή σημασία δόθηκε στο διακριτό πρόβλημα στα τελευταία χρόνια, καθώς είναι πιο ρεαλιστικό μοντέλο των πραγματικών έργων. Το πρόβλημα αυτό είναι strongly NP-Hard και για αυτό η εύρεση βέλτιστων λύσεων ήταν πολύ δύσκολη στο παρελθόν.



ΕΙΚΟΝΑ 9 ΔΙΑΚΡΙΤΗ ΣΧΕΣΗ ΧΡΟΝΟΥ-ΚΟΣΤΟΥΣ

Ο Παναγιωτακόπουλος (Panagiotakopoulos, 1977) έδειξε ότι η διάρκεια των δραστηριοτήτων και το κόστος που τους αντιστοιχεί συσχετίζονται. Στη πραγματικότητα, στα σημερινά έργα οι υπεύθυνοι πάντα πρέπει να σκέφτονται τη σχέση ανάμεσα στο συνολικό κόστος και στη συνολική διάρκεια του έργου. Αρκετές φορές για επίτευξη της μείωσης του κόστους, πρέπει να θυσιάσουν τη διάρκεια, καθυστερώντας το έργο. Άλλες φορές, πρέπει να τροποποιήσουν το χρονοδιάγραμμα του έργου για το ολοκληρώσουν γρηγορότερα, έχοντας ως αποτέλεσμα την αύξηση του κόστους. Για παράδειγμα, η πρόσληψη περισσότερων εργατών μπορεί να επιταχύνει την εκτέλεση του έργου έτσι ώστε να ολοκληρωθεί γρηγορότερα από την ημερομηνία παράδοσής του, όμως το συνολικό κόστος θα αυξηθεί. Έτσι είναι λογικό για τους υπεύθυνους του έργου να βρουν ένα πρόγραμμα

στο οποίο η ολοκλήρωσή του θα απαιτεί ελάχιστο κόστος, και συγχρόνως, θα ικανοποιεί τους χρονικούς περιορισμούς (Elmaghraby and Kamburowski, 1992).

3.3 Προγραμματισμός Έργων υπό Περιορισμένους Πόρους

3.3.1 Ορισμός Προβλήματος (RCPSP)

Το πρόβλημα προγραμματισμού δραστηριοτήτων όταν υπάρχουν περιορισμένοι διαθέσιμοι πόροι και σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, με σκοπό την ελαχιστοποίηση της διάρκειας του έργου αναφέρεται στην βιβλιογραφία ως πρόβλημα προγραμματισμού δραστηριοτήτων υπό περιορισμένους πόρους (Resource-Constrained Project Scheduling Problem) (RCPSP) (Gen and Cheng, 2000).

Αυτή η κατηγορία προβλημάτων καθορίστηκε από τον Davis το 1973 ως «*η μέθοδος προγραμματισμού μιας σειράς διεργασιών με περιορισμένο ανά χρονική περίοδο αριθμών πόρων κατά τη διάρκεια εκτέλεσης του έργου με στόχο να ελαχιστοποιηθεί η διάρκειά του» (Davis, 1973).*

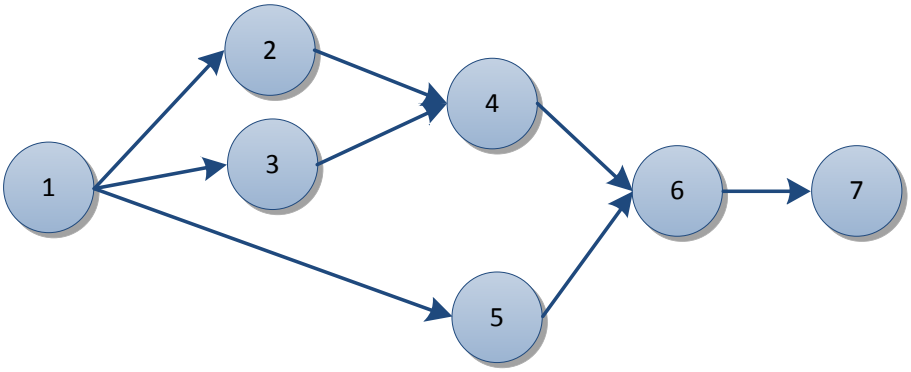
Αναλυτικά το πρόβλημα αυτό μπορεί να παρουσιαστεί ως εξής: Το έργο αποτελείται από $j = 1, \dots, J$ δραστηριότητες. Οι δραστηριότητες $j = 0$ και $j = J + 1$ είναι πλασματικές, δεν έχουνε διάρκεια και δεν απαιτούνται πόροι για την εκτέλεσή τους. Οι δραστηριότητες αυτές συμβολίζουν την αρχή και το τέλος του έργου αντίστοιχα. Η χρονική διάρκεια που χρειάζεται μια δραστηριότητα, έστω j, για να ολοκληρωθεί είναι δεδομένη και συμβολίζεται ως d_j (duration). Προκειμένου να ξεκινήσει μια δραστηριότητα j πρέπει να έχουν ολοκληρωθεί όλες οι προαπαιτούμενες δραστηριότητες της. Το σύνολο των προαπαιτούμενων δραστηριοτήτων i της δραστηριότητας j συμβολίζονται ως P_j (Predecessor). Όταν ξεκινήσει μια δραστηριότητα δεν επιτρέπεται να διακοπεί. Κάθε δραστηριότητα απαιτεί ένα συγκεκριμένο αριθμό πόρων για την ολοκλήρωσή της. Ο χρόνος ολοκλήρωσης της εκτέλεσης κάθε δραστηριότητας θα συμβολίζεται ως FT_j (Finish Time). Το σύνολο των δραστηριοτήτων που εκτελούνται στην ίδια περίοδο συμβολίζεται ως A_t όπου $FT_j - d_j + 1 \leq t \leq FT_j$. Υπάρχουν K ανανεώσιμοι πόροι που χρησιμοποιούνται στο πρόβλημα και κάθε ένας από αυτούς υπάρχει σε διαθέσιμη ποσότητα ίση με R_k. Για την ολοκλήρωση κάθε δραστηριότητας j απαιτείται ποσότητα του πόρου k ίση με r_{jk} . Στο γενικό πρόβλημα όλες οι

παράμετροι είναι ακέραιοι αριθμοί και μη μηδενικοί. Το πρόγραμμα προκύπτει από τον προγραμματισμό της έναρξης S_j των δραστηριοτήτων $j = 0, \dots, J + 1$, όπου $S_j = FT_j - d_j$. Ο στόχος είναι να προκύψει ένα πρόγραμμα με όσο το δυνατόν μικρότερο χρόνο ολοκλήρωσης. Η δομή του έργου μπορεί να απεικονιστεί από ένα δίκτυο δραστηριοτήτων-κόμβων $G = (V, E)$, όπου οι κόμβοι που ανήκουν στο σύνολο V απεικονίζουν τις δραστηριότητες ενώ τα βέλη ανήκουν στο σύνολο E και αναπαριστούν τις σχέσεις προτεραιότητας. Αυτή η λογική απεικόνισης των δραστηριοτήτων ονομάζεται δίκτυο δραστηριοτήτων στους κόμβους, στην αγγλική ορολογία AON - Activity On Nodes και είναι η πιο συχνά χρησιμοποιούμενη. Αυτή η απεικόνιση θα χρησιμοποιηθεί για την εργασία μας (Hartmann and Briskorn, 2010).

Στον πίνακα 2 και στην εικόνα 10 παρουσιάζεται ένα παράδειγμα έργου με 6 δραστηριότητες $V = (1,2,3,4,5,6)$, τις σχέσεις προτεραιότητας $E = \{(1,2), (1,3), (2,4), (3,4), (1,5), (4,6), (5,6)\}$ και το δίκτυο δραστηριοτήτων-κόμβων του. Το σύνολο τω κόμβων αποτελούν τις δραστηριότητες του έργου, και οι ακμές του υποδηλώνουν σχέσεις τέλους-αρχής.

ΠΙΝΑΚΑΣ 2 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ-ΣΧΕΣΕΙΣ ΑΛΛΗΛΟΥΧΙΑΣ

Δραστηριότητα	Προαπαιτούμενες Δραστηριότητες
1	-
2	1
3	1
4	2,3
5	1
6	4,5
7	-



ΕΙΚΟΝΑ 10 ΔΙΚΤΥΟ ΔΡΑΣΤΗΡΙΟΤΗΤΩΝ-ΚΟΜΒΩΝ ΕΡΓΟΥ

Σύμφωνα με τους παραπάνω συμβολισμούς το πρόβλημα μπορεί να μοντελοποιηθεί ως εξής:

$$Min\ FT_J \qquad (1)$$

Υπό τους περιορισμούς:

$$FT_i \leq FT_j - d_j \quad j = 2, \dots, J,\ i \in P_j, \qquad (2)$$

$$\sum_{j \in A_t} r_{jk} \leq R_k \qquad \forall\, k \in K,\ t = 1, \dots T, \qquad (3)$$

$$FT_j \geq 0 \qquad j = 1, \dots, J+1. \qquad (4)$$

Η αντικειμενική συνάρτηση (Εξ. 1) εκφράζει τον στόχο για ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης του έργου, δηλαδή ζητούμενη είναι η όσο το δυνατόν συντομότερη εκτέλεση της τελευταίας δραστηριότητάς του έργου, σύμφωνα με τους περιορισμούς προτεραιότητας και πόρων. Η εξίσωση (Εξ.2) εκφράζει τους περιορισμούς προτεραιότητας, όπου για λόγους ευκολίας των συμβόλων κάθε δραστηριότητα j έχει προαπαιτούμενες δραστηριότητες με αναγνωριστικό μικρότερο από αυτήν δηλαδή κάθε δραστηριότητα j έχει προαπαιτούμενες δραστηριότητες i, με $j > i$. Ο περιορισμός (Εξ.3) αποσκοπεί στο να εξασφαλίσει ότι σε κάθε περίοδο A_t το σύνολο των χρησιμοποιούμενων πόρων είναι μικρότερο από το ανώτατο όριο R_k για κάθε ομάδα πόρων, ενώ ο περιορισμός (4) υποδεικνύει ότι οι χρόνοι ολοκλήρωσης των διεργασιών είναι μη αρνητικοί αριθμοί (Kolisch, 1996).

3.3.2 Ορισμός Προβλήματος (MRCPSP)

Στο κλασσικό πρόβλημα προγραμματισμού δραστηριοτήτων υπό περιορισμένους πόρους (Resource-Constrained Project Scheduling Problem) (RCPSP), οι δραστηριότητες του έργου πρέπει να προγραμματιστούν έτσι ώστε να ελαχιστοποιηθεί η συνολική διάρκεια του έργου. Υπόψη πρέπει να ληφθούν οι τεχνολογικοί περιορισμοί που ορίζουν τη σειρά προτεραιότητας καθώς και η διαθεσιμότητα των ανανεώσιμων πόρων. Όταν αρχίσει να εκτελείται μια δραστηριότητα δεν μπορεί να διακοπεί.

Το πρόβλημα που θα παρουσιαστεί σε αυτήν την παράγραφο αναφέρεται στην βιβλιογραφία ως πρόβλημα προγραμματισμού δραστηριοτήτων υπό περιορισμένους πόρους και πολλαπλούς τρόπους εκτέλεσης (Multi-Mode Resource-Constrained Project Scheduling Problem) (MRCPSP). Το πρόβλημα αυτό αποτελεί γενικευμένη εκδοχή του γενικού προβλήματος RCPSP, όπου κάθε δραστηριότητα μπορεί να εκτελεσθεί με έναν από τους πολλούς τρόπους, δηλαδή με έναν από τους πολλούς συνδυασμούς διάρκειας και απαιτούμενων πόρων.

Κάθε τρόπος εκτέλεσης έχει διαφορετικό αντίκτυπο στη διάρκεια της δραστηριότητας, στο κόστος, καθώς και στην απαιτούμενη ποσότητα πόρων που απαιτούνται για την εκτέλεσή της. Διαφορετικοί τρόποι εκτέλεσης δημιουργούν σχέσης μεταξύ της διάρκειας και της ποσότητας πόρων (σχέση χρόνου/πόρων), μεταξύ της διάρκειας και του κόστους (σχέση χρόνου/κόστους) και μεταξύ της ποσότητας και του συνδυασμού των πόρων (σχέση πόρων/πόρων).

Το ζητούμενο είναι να βρεθεί ο τρόπος εκτέλεσης και ο χρόνος έναρξης κάθε δραστηριότητας έτσι ώστε ελαχιστοποιηθεί η συνολική διάρκεια του έργου βάσει των περιορισμών προτεραιότητας και ποσότητας πόρων. Εφόσον το πρόβλημα MRCPSP είναι γενικευμένο πρόβλημα του RCPSP, είναι και αυτό πρόβλημα τύπου NP-Hard.

Αναλυτικά το πρόβλημα μπορεί να παρουσιασθεί ως εξής: Θεωρούμε ένα έργο το οποίο αποτελείται από ένα σύνολο δραστηριοτήτων J . Οι δραστηριότητες j$= 0$ και j$= J + 1$ είναι πλασματικές, δεν έχουν διάρκεια και δεν απαιτούνται πόροι για την εκτέλεσή τους. . Η δομή του έργου μπορεί να απεικονιστεί από ένα δίκτυο δραστηριοτήτων-κόμβων $G = (V, E)$, όπου οι κόμβοι V απεικονίζουν τις δραστηριότητες ενώ τα βέλη E τις σχέσεις προτεραιότητας. Οι δραστηριότητες αυτές συμβολίζουν την αρχή και το τέλος του έργου αντίστοιχα. Κάθε δραστηριότητα j μπορεί να εκτελεσθεί με M_j τρόπους. Το σύνολο $M_j = 1, \dots, J + 1$ περιλαμβάνει για

κάθε δραστηριότητα j τον αριθμό των τρόπων εκτέλεσής τους ενώ ο αριθμός m δίνει τον τρόπο εκτέλεσης από τις δυνατές επιλογές σε κάθε δραστηριότητα. Κάθε τρόπος εκτέλεσης αναδεικνύει την ποσότητα των ανανεώσιμων και μη ανανεώσιμων πόρων που απαιτούνται και τη διάρκεια εκτέλεσης της δραστηριότητας. Όταν επιλεγεί ένας τρόπος εκτέλεσης δεν μπορεί να αλλάξει και όταν αρχίσει να εκτελείται η δραστηριότητα δεν μπορεί να διακοπεί Η χρονική διάρκεια που χρειάζεται η δραστηριότητα j να εκτελεσθεί με τον τρόπο εκτέλεσης m είναι δεδομένη και συμβολίζεται ως d_{jm} (duration). Υπάρχουν K^{ρ} ανανεώσιμοι πόροι που χρησιμοποιούνται στο πρόβλημα και κάθε ένας από αυτούς υπάρχει σε διαθέσιμη ποσότητα ίση με R_k^{ρ}. Επίσης υπάρχουν K^{v} μη ανανεώσιμοι πόροι και κάθε ένας από αυτούς υπάρχει σε διαθέσιμη ποσότητα ίση με R_k^{v}. Κάθε δραστηριότητα j που εκτελείται με τον τρόπο εκτέλεσης m χρησιμοποιεί r_{ρ}^{jmk} ποσότητα του ανανεώσιμου πόρου $k \in K^{\rho}$ και r_{v}^{jmk} ποσότητα του μη ανανεώσιμου πόρου $k \in K^{v}$. Στο γενικό πρόβλημα όλες οι παράμετροι είναι ακέραιοι αριθμοί και μη μηδενικοί. Εάν ο αριθμός μη ανανεώσιμων πόρων είναι μεγαλύτερος από 2 (K^{v}>=2) και υπάρχουν παραπάνω από ένας τρόπος εκτέλεσης για κάθε δραστηριότητα τότε το πρόβλημα είναι τύπου NP-Complete. (Kolisch and Drexl, 1997) (Hartmann, 2001) (Peteghem and Vanhoucke, 2010) (De Reyck and Herroelen, 1999)

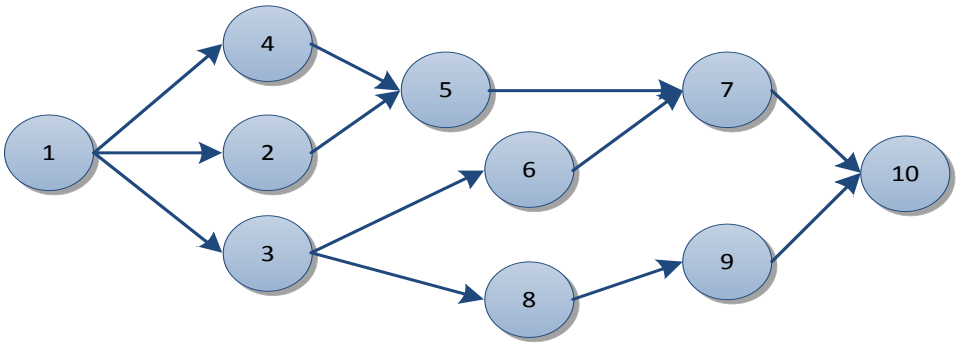
Στον παρακάτω πίνακα έχουμε ένα παράδειγμα ενός έργου που αποτελείται από 10 δραστηριότητες. Στο πρόβλημα αυτό χρησιμοποιείται ένας ανανεώσιμος και ένας μη ανανεώσιμος πόρος. Για κάθε δραστηριότητα βλέπουμε τους πιθανούς τρόπους εκτέλεσης και την αναλογία των πόρων που χρησιμοποιείται σε κάθε έναν από αυτούς.

ΠΙΝΑΚΑΣ 3 ΠΑΡΑΔΕΙΓΜΑ ΕΡΓΟΥ – ΔΕΔΟΜΕΝΑ ΠΡΟΒΛΗΜΑΤΟΣ MRCPSP

Δραστηριότητα j	Τρόπος Εκτέλεσης m_i	Διάρκεια Εκτέλεσης d_j	Ποσότητα Ανανεώσιμου Πόρου r_{ρ}^{jmk}	Ποσότητα Μη Ανανεώσιμου Πόρου r_{v}^{jmk}
1	1	0	0	0
2	1	1	4	2
	2	3	3	1
3	1	4	3	3
	2	2	4	5

4	1	5	5	3
	2	**3**	**6**	**5**
5	1	2	4	1
	2	**1**	**5**	**4**
6	1	4	3	1
	2	**2**	**5**	**2**
7	1	2	0	1
	2	**3**	**1**	**1**
8	1	1	2	3
	2	**3**	**1**	**1**
9	1	5	3	1
	2	**2**	**6**	**4**
10	1	0	0	0

.



ΕΙΚΟΝΑ 11 ΔΙΚΤΥΟ ΔΡΑΣΤΗΡΙΟΤΗΤΩΝ ΚΟΜΒΩΝ

Σύμφωνα με τους παραπάνω συμβολισμούς το πρόβλημα μπορεί να οριστεί μαθηματικά ως εξής:

$$Min\ FT_J \qquad (1)$$

Υπό τις παρακάτω προϋποθέσεις:

$$FT_i \leq FT_j - d_j \quad j = 2, \dots, J,\ i \in P_j, \qquad (2)$$

$$\sum_{j \in A_t} r_{jmk}^{\rho} \leq R_k^{\rho} \qquad k = 1, \dots, K^{\rho}, t = 1, \dots T, m \in M_j \qquad (3)$$

$$\sum_{j \in A_t} r_{jmk}^{v} \leq R_k^{v} \qquad k = 1, \dots, K^{v}, t = 1, \dots T, m \in M_j \qquad (4)$$

$$FT_j \geq 0 \qquad j = 1, \dots, J+1. \qquad (5)$$

Όπως και στο κλασσικό πρόβλημα η αντικειμενική συνάρτηση είναι η ελαχιστοποίηση του συνολικού χρόνου εκτέλεσης του έργου, δηλαδή η όσο το δυνατόν συντομότερη εκτέλεση της τελευταίας δραστηριότητάς του, σύμφωνα με τους περιορισμούς προτεραιότητας. Η δεύτερη εξίσωση προκύπτει από τους περιορισμούς προτεραιότητας. Οι περιορισμοί (3) και (4) εξασφαλίζουν ότι σε κάθε περίοδο A_t το σύνολο των χρησιμοποιούμενων ανανεώσιμων και μη ανανεώσιμων πόρων αντίστοιχα είναι μικρότερο από το ανώτατο όριο R_k^{ρ} και R_k^{v} για κάθε ομάδα πόρων. Με τον περιορισμό (5) ο χρόνος ολοκλήρωσης κάθε δραστηριότητας είναι μη αρνητικός. (Peteghem and Vanhoucke, 2010)

Μία διαφορετική μοντελοποίηση μπορεί να παρουσιασθεί ως εξής και είναι αυτή που θα χρησιμοποιηθεί παρακάτω για την παρουσίαση της σχέσης χρόνου-κόστους σε διακριτή μορφή.

$$x_{jm} = \begin{cases} 1, \text{εάν η δραστηριότητα } j \text{ εκτελείται με τον τρόπο εκτέλεσης } m \\ 0, \text{διαφορετικά} \end{cases}$$

$$\sum_{m \in M_j} x_{jm} = 1 \qquad (1)$$

$$Min\ FT_J \qquad (2)$$

Υπό τις παρακάτω προϋποθέσεις:

$$FT_i \leq FT_j - \sum_{m \in M_j} x_{jm} * d_{jm} \qquad j = 2, \dots, J,\ i \in P_j, \qquad (3)$$

$$\sum_{j \in A_t} \sum_{m \in M_j} x_{jm} * r_{jmk}^{\rho} \leq R_k^{\rho}$$
$$k = 1, \dots, K^{\rho}, t = 1, \dots T, m \in M_j \qquad (4)$$

$$\sum_{j \in A_t} \sum_{m \in M_j} x_{jm} * r_{jmk}^{v} \leq R_k^{v}$$
$$k = 1, \dots, K^{v}, t = 1, \dots T, m \in M_j \qquad (5)$$

$$FT_j \geq 0 \qquad j = 1, \dots, J+1 \qquad (6)$$

$$A_t = \{FT_j - d_j < t \leq FT_j\} \qquad (7)$$

3.3.3 Ορισμός Προβλήματος (DTCTP)

Το πρόβλημα προγραμματισμού δραστηριοτήτων όταν υπάρχουν περιορισμένοι διαθέσιμοι πόροι, σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων και της σχέσης χρόνου-κόστους σε διακριτή μορφή αναφέρεται στην βιβλιογραφία ως *Discrete Time-Cost Trade Off Resource Constrained Project Scheduling Problem* (DTCTP). Στο πρόβλημα αυτό η διάρκεια εκτέλεσης των δραστηριοτήτων είναι μία διακριτή, γνησίως φθίνουσα συνάρτηση του μοναδικού μη ανανεώσιμου πόρου που τους αντιστοιχεί. Το πρόβλημα αυτό ερευνήθηκε πρώτα από τους Hindelang και Muth. Για την επίλυση του προβλήματος πολλές από τις γνωστές μέθοδοι χρησιμοποιήθηκαν. Αναλυτικές μέθοδοι (Hindelang and Muth, 1973), αλγόριθμοι branch and bound. (Demeulemeester et al., 1996) (Erenguc et al., 2001).

Στη συνέχεια αναπτύχθηκαν και μετά-ευρετικοί αλγόριθμοι προσαρμοσμένοι στο πρόβλημα DTCTP χρησιμοποιώντας σειριακή μέθοδο παραγωγής χρονοπρογραμμάτων για την εύρεση ενός εφικτού προγράμματος παραγωγής. Οι Elmaghraby και Kamburowski ήταν οι πρώτοι που εισήγαγαν τις ποινές (penalty) και τις αμοιβές(bonus) στην αντικειμενική συνάρτηση, δίνοντας στην επίλυση του προβλήματος πιο ρεαλιστική σημασία. (Elmaghraby and Kamburowski, 1992) Η μείωση (crashing) της διάρκειας των δραστηριοτήτων εισήχθηκε στο πρόβλημα από τους Ann T και Erenguc SS, (Ahn and Erenguc, 1998) όπου σε κάθε δραστηριότητα ανάλογα με τον τρόπο εκτέλεσής της επιλέγεται και η διάρκειά της. Σε κάθε διάρκεια εφαρμόζεται μία μείωση (crashing) ανάλογα με κάποιον συντελεστή που επιλέγεται και αυτή η μείωση έχει επίδραση, προφανώς, στο κόστος της δραστηριότητας.

Τρεις μορφές του προβλήματος DTCTP έχουν εξετασθεί στη βιβλιογραφία. Το χρονικό πρόβλημα (deadline problem), το πρόβλημα κόστους (budget problem) και το efficiency πρόβλημα. Στο πρώτο, με δεδομένο το συνδυασμό χρόνου-κόστους για κάθε δραστηριότητα και τη συνολική διάρκεια του έργου, κάθε δραστηριότητα αντιστοιχίζεται σε έναν δυνατό τρόπο εκτέλεσης έτσι ώστε να ελαχιστοποιηθεί το συνολικό κόστος του έργου. Αντίστοιχα στο δεύτερο ελαχιστοποιείται η συνολική διάρκεια του έργου δεδομένου του συνολικού διαθέσιμου προϋπολογισμού. Το τελευταίο έχει ως σκοπό τη δημιουργία ενός αποδοτικού μείγματος κόστους και συνολικής διάρκειας του έργου.

Το μαθηματικό μοντέλο που ανταποκρίνεται στις απαιτήσεις του προβλήματος MRCPSP και DTCTP παρουσιάζεται παρακάτω και αναφέρεται ως MRC-DTCTP:

Εισάγεται ένας νέος παράγοντας, ο y_j , ο οποίος είναι μια μεταβλητή απόφασης.

$$y_j = \begin{cases} 1, \text{εάν η δραστηριότητα } j \text{ εκτελείται στη συμπιεσμένη διάρκειά της} \\ 0, \text{διαφορετικά} \end{cases}$$

$$x_{jm} = \begin{cases} 1, \text{εάν η δραστηριότητα } j \text{ εκτελείται με τον τρόπο εκτέλεσης } m \\ 0, \text{διαφορετικά} \end{cases}$$

$$Min\ F_c \qquad (1)$$

Υπό τις παρακάτω προϋποθέσεις:

$$\sum_{m \in M_j} x_{jm} = 1 \qquad (2)$$

$$FT_i \leq FT_j - \sum_{m \in M_j} x_{jm} * d_{jm}$$

$$\forall (i, j) \in E, d_{jm} \in \{nd_{jm}, ed_{jm}\}, \qquad (3)$$

$$\sum_{j \in A_t} \sum_{m \in M_j} (x_{jm} * r_{jmk}) \leq R_k , \qquad k = 1, \ldots, K \qquad (4)$$

$$A_t = \{FT_j - d_j + 1 \leq t \leq FT_j\}, \qquad (5)$$

$$FT_j \geq 0, \qquad j = 1, \ldots, J+1 \qquad (6)$$

$$FT_J \leq T_{max}. \qquad (7)$$

Στην παραπάνω μοντελοποίηση, η αντικειμενική συνάρτηση (1) ελαχιστοποιεί το συνολικό κόστος του έργου[2]. Η συνάρτηση αυτή αναλύεται στη παράγραφο 5.1.1. Με τον περιορισμό (2), κάθε δραστηριότητα μπορεί να εκτελεσθεί μόνο με έναν τρόπο. Ο περιορισμός (3) τηρεί τις σχέσεις αλληλουχίας και με nd_{jm} συμβολίζεται η κανονική διάρκεια της δραστηριότητας j που εκτελείται με τον τρόπο m, ενώ με ed_{jm} συμβολίζεται η συμπιεσμένη διάρκεια της δραστηριότητας j που εκτελείται με τον τρόπο m. Ο περιορισμός (4) δείχνει ότι σε κάθε περίοδο $[t-1, t]$ και για κάθε τύπο πόρου k, η ποσότητα των ανανεώσιμων πόρων που απαιτείται για τις

[2] Ο τρόπος με τον οποίο υπολογίζεται το συνολικό κόστους F_c του έργου αναλύεται στην παράγραφο 6.1.1.

δραστηριότητες που είναι σε εκτέλεση δεν μπορεί να υπερβεί τη μέγιστη διαθεσιμότητα. Όπου r_{jmk} είναι η ποσότητα του τύπου ανανεώσιμων πόρων k που χρειάζεται η δραστηριότητα j όταν εκτελείται με τον τρόπο m. Σε αυτό το πρόβλημα, όπως έχει προαναφερθεί, υπάρχουν πολλαπλοί ανανεώσιμοι πόροι αλλά μόνο ένας μη ανανεώσιμος, το κόστος. Τέλος ο περιορισμός (5) δείχνει ότι ο χρόνος ολοκλήρωσης κάθε δραστηριότητας είναι μη αρνητικός και ο περιορισμός (7) αναγκάζει το έργο να τελειώσει γρηγορότερα από το άνω όριο που έχει τεθεί T_{max}.

3.4 Μέθοδοι Επίλυσης Προβλήματος

Οι τεχνικές επίλυσης των RCPSPs κατατάσσονται σε δύο βασικές κατηγορίες: τις αναλυτικές προσεγγίσεις (exact procedures) και τις ευρετικές μεθόδους (heuristics). Η χρήση των μεθόδων της πρώτης κατηγορίας οδηγεί στη βέλτιστη λύση. Παρουσιάζει όμως το μειονέκτημα των μεγάλων υπολογιστικών χρόνων, κυρίως σε προβλήματα με μεγάλο αριθμό δραστηριοτήτων. Για το λόγο αυτό οι αναλυτικές μέθοδοι που έχουν χρησιμοποιηθεί για την επίλυση των RCPSP κυρίως περιορίζονται στην περίπτωση του RCPSP όπου υπάρχει μόνο ένας τρόπος εκτέλεσης για κάθε δραστηριότητα (single-mode case) και είναι:

• ο δυναμικός προγραμματισμός

• ο ακέραιος 0-1 προγραμματισμός και

• οι τεχνικές περιορισμού και διακλάδωσης (**branch and bound techniques**).

Οι ευρετικές μέθοδοι αν και είναι υπολογιστικά πολύ ταχύτερες, σε σχέση με τις βέλτιστες προσεγγίσεις, δεν παρέχουν πάντοτε τη βέλτιστη λύση. Ευρετικές μέθοδοι που έχουν χρησιμοποιηθεί για την επίλυση των *RCPSP* είναι:

• η παραγωγή χρονοπρογράμματος με βάση κανόνες προτεραιότητας (**priority-based scheduling**)

• ευρετικές μέθοδοι που βασίζονται σε τεχνικές Περιορισμού και Διακλάδωσης (**Branch and Bound**)

• η μέθοδος διαζευκτικών τόξων (**disjunctive arc concepts**)

• μετά-ευρετικές τεχνικές (**metaheuristic techniques**), όπως η μέθοδος προσομοιωμένης ανόπτησης (**simulated annealing**), η αναζήτηση ταμπού (**Tabu Search**) και οι γενετικοί αλγόριθμοι (**genetic algorithms**).

3.4.1 Αναλυτικές Μέθοδοι (Exact Methods)

Δυναμικός Προγραμματισμός (dynamic programming)

Στη μέθοδο του δυναμικού προγραμματισμού θα μπορούσε να δοθεί και η ονομασία «επαναλαμβανόμενη» (recursive) ή «πολυσταδιακή» (multistage) δεδομένου ότι ερμηνεύει τα προβλήματα βελτιστοποίησης ως πολυσταδιακές διαδικασίες απόφασης (multistage decision processes). Αυτό σημαίνει ότι το πρόβλημα είναι διαιρεμένο σε έναν αριθμό επιπέδων και σε κάθε επίπεδο απαιτείται μία απόφαση που θα επηρεάσει τις αποφάσεις που θα γίνουν στα επόμενα επίπεδα. Η μέθοδος αυτή είναι εκθετικής υπολογιστικής πολυπλοκότητας. Για παράδειγμα, για την επίλυση ενός μικρού προβλήματος είναι επαρκής ο αλγόριθμος δυναμικού προγραμματισμού, ο οποίος χρειάζεται έξι δευτερόλεπτα. Ο χρόνος εκτέλεσης επεκτείνεται στα 450 δευτερόλεπτα με μόλις 16 εργασίες και αυτή η εκθετική αύξηση τον κάνει ακατάλληλο για πραγματικά προβλήματα που αποτελούνται από εκατοντάδες εργασίες.

Αλγόριθμος Διακλάδωση και Περιορισμός (Branch and Bound)

Ο αλγόριθμος Branch and Bound (**Διακλάδωση και Περιορισμός**) είναι ένας γενικός αλγόριθμος για την εύρεση βέλτιστων λύσεων των διαφόρων προβλημάτων βελτιστοποίησης, κυρίως σε διακριτή και συνδυαστική βελτιστοποίηση. Η μέθοδος προτάθηκε για πρώτη φορά από τους A. H. Land and A. G. Doig το 1960.(Land and Doig, 1960)

«*Έχουμε ένα πεπερασμένο αριθμό εφικτών λύσεων* $\mathcal{S}$ *και ένα κριτήριο* $\gamma: \mathcal{S} \to R$. *Στόχος είναι να βρούμε τα S* που ανήκουν στο* $\mathcal{S}$ *έτσι ώστε γ(S*) =* $min_{S\mathcal{E}S}\, \gamma(S)$ *»* (Blazewicz, 2007).

Ο αλγόριθμος Branch and Bound (**Διακλάδωση και Περιορισμός**) βρίσκει τις λύσεις S*, έμμεσα, πραγματοποιώντας μερική μόνο απαρίθμηση (implicit enumeration[3]). Το σύνολο των εφικτών λύσεων $\mathcal{S}$ διασπάται σε υποσύνολα και κάθε ένα από αυτά αποτελεί μία ομάδα λύσεων του αντίστοιχου υποπροβλήματος στο γενικό πρόβλημα. Σε κάθε στάδιο του αλγορίθμου, ένα υποσύνολο επιλέγεται και γίνεται έλεγχος για την εύρεση της καλύτερης λύσης.

[3] Ο αλγόριθμος Branch and Bound είναι μέθοδος μερικής απαρίθμησης καθώς θεωρεί δεδομένες λύσεις του προβλήματος χωρίς να τις υπολογίζει.

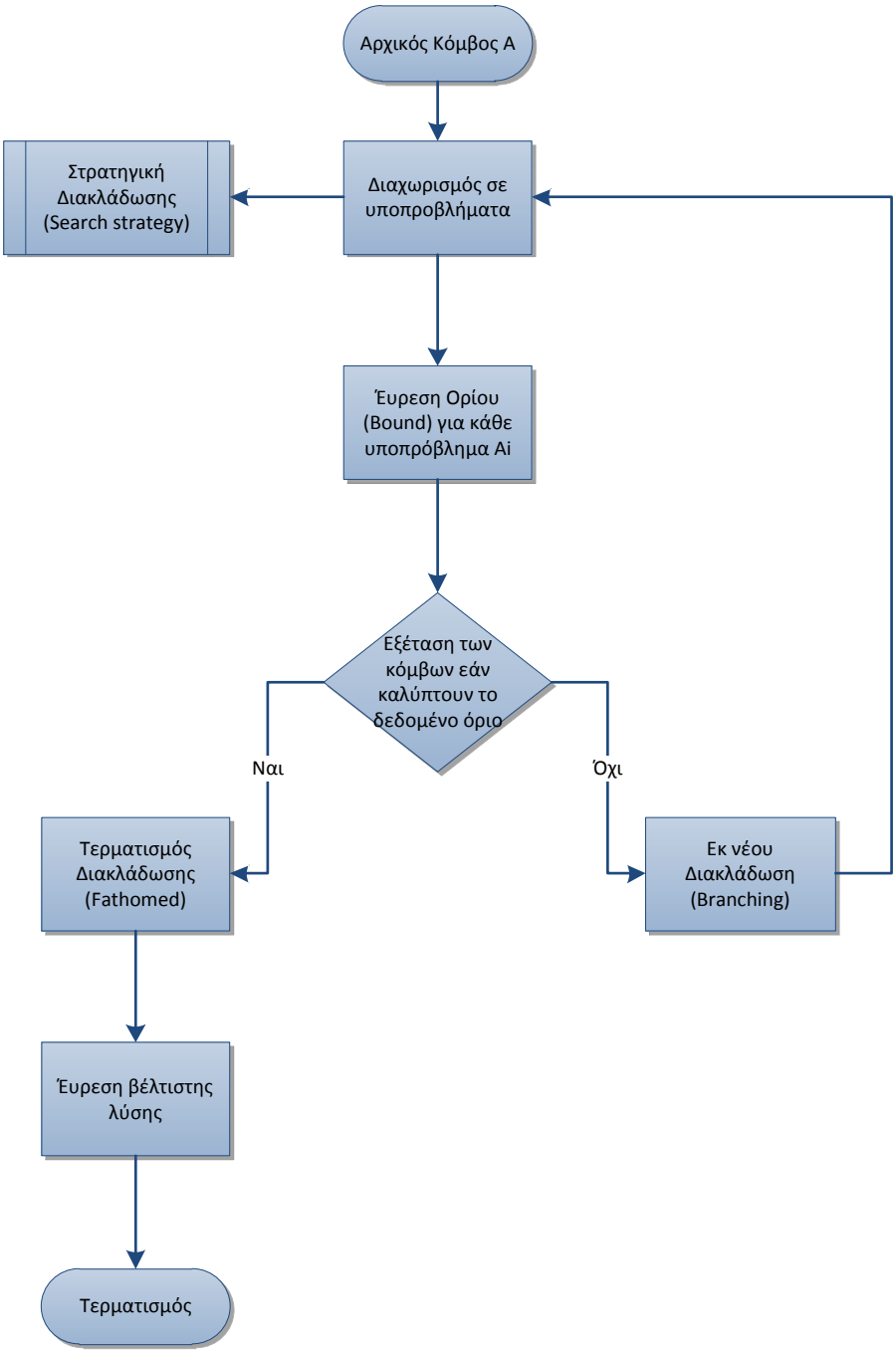
Ο αλγόριθμος αυτός αποτελείται από δύο διαδικασίες, την διακλάδωση (branch) και τον περιορισμό (bound). Η πρώτη διαδικασία είναι υπεύθυνη για τον διαχωρισμό του προβλήματος σε δύο ή περισσότερα υποπροβλήματα. Η δεύτερη διαδικασία υπολογίζει ένα κατώτατο όριο (lower bound) και με βάση αυτό δέχεται εάν ένα υποπρόβλημα μπορεί να συνεχίσει στην επόμενη φάση ή όχι. Τα όρια αυτά χρησιμοποιούνται για την επιλογή των καλύτερων λύσεων και εμμέσως των υποπροβλημάτων και την απόρριψη αυτών που δεν θα μπορούσαν να έχουν ικανοποιητικές λύσεις. Σε κάθε στάδιο του αλγορίθμου, κάποιο υποσύνολο εξετάζεται εάν πληροί το κριτήριο του κατώτατου ορίου. Εάν σε κάποιο υποσύνολο η τιμή της αντικειμενικής συνάρτησης για την λύση του γ(S) είναι μικρότερη του κατώτατου ορίου τότε δεν διαχωρίζεται πλέον αυτό το υποσύνολο. Το υποσύνολο λέγεται πως είναι αξιολογημένο (fathomed).

Όταν ο αλγόριθμος αρχίζει να εκτελείται αναζητείται μια αρχική λύση του προβλήματος (trial solution). Σε κάθε διακλάδωση του προβλήματος συγκρίνεται αυτή η λύση με την λύση της διακλάδωσης και εάν δεν είναι καλύτερη αποκόπτει τους κλάδους του δέντρου που δίνουν σίγουρα μεγαλύτερη τιμή στο αποτέλεσμα. Έτσι, η αναζήτηση δε διασχίζει αυτά τα μέρη του δέντρου και η όλη διαδικασία γίνεται συντομότερη.

Ο τρόπος με τον οποίο αναζητείται η βέλτιστη λύση του προβλήματος είναι επιλογή μίας εκ των δύο στρατηγικών αναζήτησης. Η πρώτη ονομάζεται αναζήτηση πρώτα σε βάθος (depth first search, DFS, backtracking) και η δεύτερη αναζήτηση πρώτα των βέλτιστων (best first search, BFS, jumptracking).

Η διαδικασία της διακλάδωσης μπορεί να παρουσιαστεί ως ένα δέντρο (search tree). Στο πρώτο επίπεδο (πρώτο κόμβο) βρίσκεται το αρχικό πρόβλημα και σε κάθε επόμενο επίπεδο βρίσκονται συγκεκριμένα υποπροβλήματα του προβλήματος του προηγούμενου επιπέδου.

Τα βήματα του αλγορίθμου παρουσιάζονται συνοπτικά στο διάγραμμα ροής της εικόνας 12.



ΕΙΚΟΝΑ 12 ΑΛΓΟΡΙΘΜΟΣ BRANCH AND BOUND

Οι περισσότεροι αλγόριθμοι branch and bound χρησιμοποιώντας μία τεχνική χαλάρωσης των περιορισμών των πόρων και υπολογίζοντας την μέγιστη διάρκεια του έργου (longest path) από το γράφο βρίσκουν το κατώτατο όριο-φράγμα (lower bound). Στην συνέχεια, κατά κανόνα γίνεται χρήση κριτηρίων-περιορισμών για να αποκλεισθούν οι κόμβοι-λύσεις που δεν οδηγούν σε βέλτιστη λύση.

Για την υλοποίηση του αλγορίθμου branch and bound και οι αποφάσεις που πρέπει να παρθούν είναι:

- Η διαδικασία διακλάδωσης.
- Η στρατηγική αναζήτησης.
- Η διαδικασία περιορισμού, ή
- Τα κριτήρια που πρέπει να ισχύουν στο πρόβλημα και εάν δεν ικανοποιούνται τότε οι λύσεις αποκλείονται.

Εν κατακλείδι, ο αλγόριθμος branch and bound βρίσκει βέλτιστη λύση όταν το πρόβλημα μας είναι μικρό (λίγες δραστηριότητες). Αλλιώς όταν το πρόβλημα είναι αυξημένης πολυπλοκότητας η χρήση του καθίσταται ασύμφορη λόγω του υπολογιστικού κόστους, δηλαδή του χρόνου, και της υπολογιστικής μνήμης που απαιτείται.

3.4.2 Ευρετικοί Αλγόριθμοι

Ενώ οι αναλυτικοί μέθοδοι βρίσκουν πάντα την βέλτιστη λύση, εάν υπάρχει, οι ευρετικοί αλγόριθμοι οδηγούν κάποιες φορές στη βέλτιστη λύση, αλλά τις περισσότερες φορές βρίσκουν μία καλή λύση. Η διαφορά των δύο μεθόδων είναι ότι οι δεύτεροι απαιτούν πολύ λιγότερο χρόνο από ότι οι πρώτοι.

Σύμφωνα με τον Blazewicz (1983) το πρόβλημα RCPSP, το οποίο είναι εξειδίκευση του προβλήματος job-shop, είναι τύπου NP-hard (δύσκολου) προβλήματος. Για αυτό τον λόγο οι ευρετικοί αλγόριθμοι είναι απαραίτητοι για την επίλυση αυτού του είδους προβλημάτων. Συχνά στην πραγματικότητα όπου τα έργα αποτελούνται από πολλές δραστηριότητες, οι υπεύθυνοι διαχείρισης του έργου δεν έχουν την πολυτέλεια για την αναζήτηση της βέλτιστης λύσης όταν αυτό απαιτεί πολύ χρόνο. Έτσι συμβιβάζονται με μία καλή λύση η οποία είναι διαθέσιμη άμεσα.

Μέθοδοι διαζευκτικών τόξων (Disjunctive arc based methods)

Η βασική ιδέα των μεθόδων αυτών είναι η επέκταση των σχέσεων προήγησης (το σύνολο των συζευκτικών τόξων - conjunctive arcs) προσθέτοντας διαζευκτικά βέλη (disjunctive arcs) (Kolisch and Hartmann, 1999). Ο στόχος είναι να εξαλειφθούν τα «απαγορευμένα» σύνολα δραστηριοτήτων, δηλαδή τα σύνολα των ανεξάρτητων δραστηριοτήτων οι οποίες δεν μπορούν να δρομολογηθούν ταυτόχρονα λόγω περιορισμένων πόρων. Έτσι προκύπτει το συντομότερο εφικτό χρονοπρόγραμμα και από άποψη περιορισμών πόρων και από άποψη σχέσεων προτεραιότητας.

Μέθοδοι Παραγωγής Χρονοπρογραμμάτων (Schedule Generation Schemes)

Ο Kelley (James E. Kelley and Walker, 1959) ήταν ο πρώτος που εισήγαγε την μέθοδο του Schedule Generation Scheme (SGS) και έκτοτε πολλοί ερευνητές έχουν ασχοληθεί με αυτή και έχουν δημιουργήσει ευρετικούς αλγορίθμους που βασίζονται σε αυτή. Αρχικά παρουσιάζεται η μέθοδος παραγωγής χρονοπρογραμμάτων (SGS) η οποία είναι ο πυρήνας για την δημιουργία των ευρετικών αλγορίθμων. Υπάρχουν δύο είδη μεθόδων παραγωγής χρονοπρογραμμάτων. Η σειριακή (serial) και η παράλληλη (parallel) μέθοδος.

Αυτές οι μέθοδοι παράγουν ένα εφικτό (feasible) χρονοπρόγραμμα ξεκινώντας από το μηδέν. Για τον προγραμματισμό όλων των δραστηριοτήτων χρησιμοποιούν ένα ημιτελές πρόγραμμα (partial schedule), δηλαδή ένα πρόγραμμα στο οποίο έχουν προγραμματιστεί ένα μέρος του συνόλου των δραστηριοτήτων.

- **Σειριακή Μέθοδος Παραγωγής Προγράμματος (Serial Schedule Generation Scheme)**

Η σειριακή μέθοδος παραγωγής προγράμματος περιλαμβάνει j=1,…,J στάδια. Στο καθένα από αυτά υπολογίζονται οι δραστηριότητες που είναι υποψήφιες να επιλεγούν προς χρονικό προγραμματισμό. Αυτές οι δραστηριότητες ανήκουν πλέον σε ένα καινούργιο υποσύνολο. Από αυτό το υποσύνολο επιλέγεται μια δραστηριότητα και προγραμματίζεται. Υπάρχουν πολλά κριτήρια επιλογής για το ποια δραστηριότητα θα προγραμματιστεί, τα οποία θα αναλυθούν αργότερα. Το σύνολο επιλογής περιλαμβάνει επομένως τις δραστηριότητες των οποίων οι προαπαιτούμενες έχουν προγραμματιστεί και συγχρόνως δεν παραβιάζουν τους περιορισμούς πόρων και σχέσεων προτεραιότητας. Αυτή η μέθοδος χρησιμοποιήθηκε για την υλοποίηση του προγράμματός μας οπότε και θα αναλυθεί διεξοδικά σε παρακάτω κεφάλαιο.

- **Παράλληλη Μέθοδος Παραγωγής Προγράμματος (Parallel Schedule Generation Scheme)**

Στην παράλληλη μέθοδο παραγωγής χρονοπρογράμματος οι δραστηριότητες κατηγοριοποιούνται σε τρεις ομάδες. Στην πρώτη ομάδα ανήκουν οι δραστηριότητες που επιλέχθηκαν να προγραμματιστούν και έχουν ολοκληρωθεί, ενώ στην δεύτερη αυτές που έχουν επιλέχθηκαν να προγραμματιστούν και βρίσκονται κατά τη διάρκεια εκτέλεσής τους. Οι δύο πρώτες ομάδες αναφέρονται ως complete set C_n και active

set A_n αντίστοιχα. Επίσης όπως και στη σειριακή μέθοδο υπάρχει το decision set D_n, έτσι υπάρχει και εδώ, μόνο που τώρα σε κάθε στάδιο ανήκουν οι δραστηριότητες των οποίων οι προαπαιτούμενες έχουν προγραμματιστεί και έχουν ολοκληρωθεί (Kolisch, 1996).

Σύγκριση για το ποια μέθοδος αποδίδει καλύτερα έχει γίνει από τον Valls (1992). Η έκβαση της έρευνας δεν είχε συγκεκριμένο αποτέλεσμα για το ποιο είναι πιο αποδοτικό. Αντίθετα, οι Alvarez-Valdes and Tamarit (1989) έφτασαν στο συμπέρασμα ότι η παράλληλη μέθοδος παραγωγής χρονοπρογράμματος οδηγεί σε καλύτερα αποτελέσματα αλλά μπορεί πιο εύκολα να εγκλωβιστεί σε ένα τοπικό ακρότατο και να χάσει τη βέλτιστη (Kolisch, 1996).

Ωστόσο πρέπει να αναφερθεί ότι ένα πρόγραμμα το οποίο προγραμματίζεται σύμφωνα με τη σειριακή μέθοδο και οποιοδήποτε κανόνα προτεραιότητας είναι ενεργό πρόγραμμα, ενώ ένα πρόγραμμα το οποίο προγραμματίζεται ακολουθώντας την παράλληλη μέθοδο και οποιονδήποτε κανόνα προτεραιότητας ανήκει στα χρονικά προγράμματα μη-καθυστέρησης. (Kolisch, 1996).

- **Οπισθοδρομικός Προγραμματισμός (Backward Planning)**

Εκτός από τις δύο βασικές μεθόδους παραγωγής χρονοπρογραμμάτων υπάρχουν και άλλες όπως η Backward Planning κατά την οποία ο προγραμματισμός ξεκινά από την πλασματική δραστηριότητα του τέλους του έργου και συνεχίζει μέχρι τον προγραμματισμό της αρχικής (Demeulemeester and Herroelen, 2002).

- **Αμφίδρομος Προγραμματισμός (Bidirectional Planning)**

Αυτή η μέθοδος συνδυάζει και την σειριακή ή παράλληλη μέθοδο και τον οπισθοδρομικό προγραμματισμό χρονοπρογράμματος. Για την υλοποίηση της μεθόδου χρησιμοποιούνται δύο λίστες προτεραιότητας, μία για κάθε είδος προγραμματισμού. (Demeulemeester and Herroelen, 2002)

Παραγωγή Χρονοπρογράμματος με βάση κανόνες προτεραιότητας (Priority Rule Based Heuristics)

Οι ευρετικές μέθοδοι που βασίζονται σε κανόνες προτεραιότητας συνδυάζουν τους κανόνες προτεραιότητας και τις μεθόδους παραγωγής χρονοπρογραμμάτων και καταλήγουν σε ένα ή περισσότερα εφικτά χρονοπρογράμματα εκτέλεσης των δραστηριοτήτων ενός έργου.

Οι αλγόριθμοι αυτοί αποτελούνται λοιπόν από δύο κύρια χαρακτηριστικά:

1. Μέθοδο παραγωγής χρονοπρογραμμάτων
2. Κανόνα προτεραιότητας

Κανόνες Προτεραιότητας (Priority Rules)

Οι κανόνες προτεραιότητας είναι υπεύθυνοι για την επιλογή της δραστηριότητας που θα προγραμματιστεί από το σύνολο των δραστηριοτήτων που βρίσκεται στο Decision set. Συνολικά παραπάνω από 73 τέτοιοι κανόνες υπάρχουν στην βιβλιογραφία. Αυτοί μπορούν να κατηγοριοποιηθούν ανάλογα με κάποια κοινά τους χαρακτηριστικά. Οι κυριότερες κατηγορίες είναι οι εξής σύμφωνα με τον Lawrence (1985):

- Κανόνες προτεραιότητας βάσει δραστηριοτήτων: Οι πληροφορίες που χρειάζονται για να δημιουργηθεί η λίστα προτεραιότητας σχετίζεται απευθείας με τη δραστηριότητα και μόνο. Σε αυτήν την κατηγορία τρεις είναι οι βασικοί κανόνες, δύο εκ των οποίων σχετίζονται με τη διάρκεια των δραστηριοτήτων.
    1. Μικρότερη διάρκεια εκτέλεσης (shortest processing time) (SPT)
    2. Μεγαλύτερη διάρκεια εκτέλεσης (longest processing time) (LPT)
    3. Τυχαία ανάθεση τιμών στη λιστα (random) (RND)
- Κανόνες βάση του Δικτύου (Network)

Βασίζονται στη σειρά προτεραιότητας των δραστηριοτήτων σε ένα έργο. Δεν χρησιμοποιούν πληροφορίες σχετικά με τους πόρους.

1. Περισσότεροι άμεσοι επιτυχόντες δραστηριότητες. Όσοι περισσότεροι τόσο μεγαλύτερη τιμή έχει η δραστηριότητα στη λίστα (most immediate successors) (MIS),
2. Περισσότερες συνολικά δραστηριότητες που έπονται (most total successors) (MTS).
3. Οι δραστηριότητες με τις λιγότερες σχέσεις αλληλουχίας-εξάρτησης (least non-related jobs) (LNRJ).

4. Οι δραστηριότητες με τη μεγαλύτερη διάρκεια συμπεριλαμβανομένω των άμεσων δραστηριοτήτων που έπονται (greatest rank positional weight) (GRPW).

❖ Κανόνες που στοχεύουν στην εξυπηρέτηση κατά προτεραιότητα των εργασιών της κρίσιμης διαδρομής (Critical path based priority rules).

1. Η δραστηριότητα με τη μικρότερη νωρίτερη έναρξη (earliest start time) (EST).
2. Η δραστηριότητα με τη μικρότερη νωρίτερη λήξη (earliest finish time) (EFT).
3. Η δραστηριότητα με το μικρότερο χρόνο έναρξης (latest start time) (LST).
4. Η δραστηριότητα με το μικρότερο χρόνο λήξης (latest finish time) (LFT).
5. Η δραστηριότητα με το μικρότερο περιθώριο (slack) ξεκινά πρώτη (minimum slack) (MSLK)

❖ Κανόνες βάσει των πόρων. Αυτοί οι κανόνες παίρνουν υπόψη την αναγκαία ποσότητα των πόρων που χρειάζεται κάθε δραστηριότητα για να εκτελεσθεί.

1. Οι περισσότεροι πόροι πρώτοι. Οι εργασίες κατατάσσονται βάση της χρήσης ενός συγκεκριμένου πόρου. Η λογική του κανόνα είναι ότι οι σημαντικότερες δραστηριότητες απαιτούν περισσότερους πόρους (greatest resource demand) (GRD)
2. Η δραστηριότητα της οποίας οι απαιτήσεις πόρων αυτής και των άμεσών της προαπαιτούμενων δραστηριοτήτων προγραμματίζεται πρώτη. (greatest cumulative resource demand) (GCUMRDComposite κανόνες που χρησιμοποιούν πληροφορίες από τους παραπάνω κανόνες και δημιουργούν ένα μέσο όρο.

Ένας κανόνας προτεραιότητας αντιστοιχεί μια τιμή (value) (vj) σε κάθε δραστηριότητα και δηλώνει εάν θα επιλεγεί η δραστηριότητα με τη μεγαλύτερη ή τη μικρότερη τιμή. Στην περίπτωση που δυο ή περισσότερες δραστηριότητες έχουν την ίδια αξία συνήθως επιλέγεται η δραστηριότητα με το μικρότερο αύξοντα αριθμό. Στον Πίνακα 4 φαίνονται οι πιο γνωστοί κανόνες-κριτήρια απόδοσης προτεραιοτήτων.

ΠΙΝΑΚΑΣ 4 ΓΝΩΣΤΟΙ ΚΑΝΟΝΕΣ ΠΡΟΤΕΡΑΙΟΤΗΤΑΣ

Κανόνες Προτεραιότητας	Ερευνητές	V(j)	Στόχος
MTS	Alvarez-Valdez and Tamarit	$\|S_j\|$	μεγιστοποίηση
LST	Alvarez-Valdez and Tamarit	$LFT_J - d_j$	ελαχιστοποίηση
GRPW	Alvarez-Valdez and Tamarit	$d_j + \sum_{i \in S_j} d_i$	μεγιστοποίηση
WRUP	Ulusoy and Ozdamar	$0{,}7\|S_j\| + 0{,}3 \sum_{r \in R} k_{jr} / K_r$	μεγιστοποίηση
LFT	Davis and Patterson	LFT_J	ελαχιστοποίηση
MSLK	Davis and Patterson	$LFT_J - EFT_J$	ελαχιστοποίηση

Σε έρευνες που έχουν διεξαχθεί οι κανόνες προτεραιότητας που βρίσκονται στον πίνακα καταλαμβάνουν τις πρώτες θέσεις όσον αφορά την αποτελεσματικότητά τους. Ο κανόνας minimum slack - MSLK είναι ο καλύτερος από τους κανόνες προτεραιότητας διότι ελαχιστοποιεί την καθυστέρηση του έργου και τον χρόνο που μένουν αδρανείς οι πόροι.

Οι X-Pass Methods ή Priority rule based heuristics σύμφωνα με τους Hartmann και Kolisch συνδυάζουν τις μεθόδους παραγωγής χρονοπρογράμματος και κανόνες προτεραιότητας για την εύρεση μιας λύσης. Εάν χρησιμοποιώντας μία μέθοδο και ένα μόνο κανόνα παράγουν ένα πρόγραμμα λέγονται μοναδικού βήματος (X-pass Method) (X=1), ενώ εάν χρησιμοποιούν διαφορετικές μεθόδους ή κανόνες και παράγουν πολλαπλά προγράμματα σε διαδοχικές επαναλήψεις, λέγονται multi – pass methods (X>1). (Hartmann and Kolisch, 2000)

3.4.3 Μετά-Ευρετικοί Αλγόριθμοι

Μετά-ευρετική (metaheuristic) καλείται μια υπολογιστική μέθοδος βελτιστοποίησης η οποία στην πραγματικότητα δεν γνωρίζει τη φύση του προβλήματος που προσπαθεί να επιλύσει και για το λόγο αυτό μπορεί να προσαρμοστεί σε πολλά και διαφορετικά προβλήματα. Στα θετικά της μεθόδου

συγκαταλέγεται το γεγονός ότι δίνει καλύτερα αποτελέσματα σε σχέση με μία ευρετική μέθοδο, ενώ στα αρνητικά ότι χρειάζεται να αξιολογήσει πληθώρα λύσεων, αυξάνοντας ενδεχομένως τον υπολογιστικό χρόνο.

Συστήματα Μυρμηγκιών (Ant System)

Η βελτιστοποίηση αποικιών μυρμηγκιών προσομοιώνει τη συμπεριφορά των μυρμηγκιών καθώς κινούνται για αναζήτηση της τροφής τους (Dorigo and Gambardella, 1997). Τα μυρμήγκια αφήνουν καθ'όλη τη διαδρομή τους, από τη φωλιά προς το σημείο προορισμού, μια ουσία που ονομάζεται φερομόνη, η οποία διεγείρει το ένστικτο της πείνας μέσω της οσμής. Η φερομόνη τοποθετείται στο μονοπάτι για να την ακολουθήσουν τα άλλα μυρμήγκια. Όλο και περισσότερη φερομόνη αφήνεται στο μονοπάτι και έτσι μοιάζει σαν μία κατάσταση στην οποία όλος ο πληθυσμός προσπαθεί να ακολουθήσει το συντομότερο μονοπάτι. Έχει ευρεία απήχηση στον επιστημονικό κύκλο επιτυγχάνοντας καλά αποτελέσματα στην επίλυση του RCPSP. (Merkle et al., 2002, Merkle and Middendorf, 2000)

Το μειονέκτημα αυτής της μεθόδου είναι ότι συγκλίνει πρόωρα στα τοπικά ελάχιστα. Μηχανισμοί που αποτρέπουν τον αλγόριθμο να αναζητούν λύσεις στα τοπικά ελάχιστα έχουν βρεθεί. Οι Stuzle και Hoos (St\ et al., 2000) πρότειναν μια συνάρτηση αξιολόγησης που λαμβάνει υπόψη το βαθμό σύγκλισης και την απόσταση από τη βέλτιστη λύση,

Μέθοδος Αναζήτησης με Απαγορευμένες Κινήσεις (Tabu Search)

Η ταμπού αναζήτηση (tabu search, **TS**) εισήχθη από το Fred Glover. Πρόκειται για μια μέθοδο τοπικής αναζήτησης, η οποία προσπαθεί να ξεπεράσει τον εγκλωβισμό σε τοπικά ελάχιστα της αντικειμενικής συνάρτησης. (Glover and Taillard, 1993)

Ο αλγόριθμος αυτός υπολογίζει όλες τις εφικτές λύσεις στο χώρο που ψάχνει και βρίσκει την βέλτιστη, δηλαδή αυτή με τη μικρότερη συνολική διάρκεια. Σε κάθε βήμα της επαναληπτικής διαδικασίας επιλέγεται μόνο μία λύση, αντίθετα με ότι συμβαίνει σε άλλους αλγορίθμους και μάλιστα μπορεί να επιλεγεί κάποια που είναι χειρότερη από την ήδη υπάρχουσα. Για να αποφευχθεί αυτό το φαινόμενο χρησιμοποιείται ένας μηχανισμός ο οποίος θυμάται ποιες λύσεις έχουν βρεθεί και ψάχνει σε κατευθύνσεις που ενδέχεται να βρεθούν καλύτερες λύσεις. Αυτό το φαινόμενο ονομάζεται κριτήριο φιλοδοξίας (inspiration criterion) και αποτελεί μαζί με τη δομή της γειτονιάς (neighborhood structure), τις κινήσεις (moves) και τη λίστα

ταμπού (tabu list) τα βασικά χαρακτηριστικά γνωρίσματα μιας ταμπού αναζήτησης. Με τον όροι κίνηση εννοούμε το πως βρίσκει ο αλγόριθμος μία γειτονική λύση δεδομένης μιας υπάρχουσας. Η λίστα ταμπού περιλαμβάνει ένα σύνολο απαγορευμένων κινήσεων, οι οποίες δεν επιτρέπεται να συμπεριληφθούν στη λύση που αναζητείται. Η λίστα αυτή ενημερώνεται διαρκώς με νέες εισόδους σε αυτήν ή και διαγραφές. Το κριτήριο φιλοδοξίας είναι μια προϋπόθεση, την οποία αν ικανοποιεί μια κίνηση τότε η τελευταία διαγράφεται από τη λίστα ταμπού (Demeulemeester and Herroelen, 2002).

Προσομοιωμένη Ανόπτηση (Simulated Annealing)

Η προσομοιωμένη ανόπτηση (simulated annealing – SA) είναι μια μέθοδος τοπικής αναζήτησης, η οποία αξιοποιεί την αναλογία που υπάρχει μεταξύ των προβλημάτων συνδυαστικής βελτιστοποίησης και των προβλημάτων της στατιστικής μηχανικής (statistical mechanics). Οι πρώτοι που παρατήρησαν την αναλογία ήταν οι S Kirpatrick, C Gelatti και M Vecchi (Kirkpatrick et al., 1983), οι οποίοι τη χρησιμοποίησαν ως μια μέθοδο για να ξεπεράσουν τον εγκλωβισμό σε τοπικά ελάχιστα.

Ο αλγόριθμος ξεκινά από μια λύση και αναζητά άλλες καλύτερες σε μια «γειτονιά» της αρχικής, το μέγεθος της οποίας ορίζεται από το χρήστη. Σε κάθε επανάληψη επιλέγεται τυχαία μια νέα λύση στη περιοχή της προηγούμενης, η οποία αξιολογείται και, αν προκύψει καλύτερη από την προηγούμενη, γίνεται αυτόματα αποδεκτή ενώ αν είναι χειρότερη αυτής γίνεται δεκτή με πιθανότητα η οποία είναι αύξουσα συνάρτηση της θερμοκρασίας και φθίνουσα συνάρτηση της επιδείνωσης στη συνάρτηση κόστους. Κατ' αυτόν τον τρόπο επιτυγχάνεται, από τη μία υψηλή εξερεύνηση στις αρχικές επαναλήψεις όπου η θερμοκρασία είναι υψηλή και ο αλγόριθμος μπορεί να ξεφύγει του εγκλωβισμού σε τοπικά ακρότατα και από την άλλη, εκμετάλλευση στα τελικά στάδια όσο η θερμοκρασία μειώνεται και ο αλγόριθμος καταλήγει στην περιοχή της βέλτιστης λύσης.

Γενετικοί Αλγόριθμοι

Οι γενετικοί αλγόριθμοι μπορούν να χρησιμοποιηθούν για οποιαδήποτε διαδικασία όπου αναζητείται βέλτιστη λύση προσομοιάζοντας τη φυσική εξελιγκτική διαδικασία επιλογής. Εκτενής ανάλυση των Γενετικών Αλγορίθμων γίνεται στη παράγραφο 5.2.

3.4.4 Επιπλέον Παράμετροι

Στο κλασσικό πρόβλημα RCPSP οι δραστηριότητες δεν μπορούν να διακοπούν. Ωστόσο στην πραγματικότητα υπάρχουν δραστηριότητες που διακόπτονται σε ένα χρονικό σημείο της εκτέλεσής τους και συνεχίζονται αργότερα. Αυτού του είδους το πρόβλημα αναφέρεται στη βιβλιογραφία ως preemptive resource constrained project scheduling problem.

3.4.5 Εναλλακτικοί στόχοι βελτιστοποίησης

Οι βασικοί στόχοι βελτιστοποίησης, δηλαδή ο στόχος της αντικειμενικής συνάρτηση είναι είτε η ελαχιστοποίηση της συνολικής διάρκειας του έργου, είτε η ελαχιστοποίηση του κόστους του έργου, είτε η δημιουργία καμπύλης με άξονες το κόστος και τη διάρκεια του έργου, για την εύρεση σημείου με βέλτιστο συνδυασμό κόστους-διάρκειας. Παρόλα αυτά, στο σύγχρονο κόσμο τα προβλήματα που πρέπει να επιλυθούν είναι πολύ πιο σύνθετα και οι στόχοι που θέτονται από την ομάδα έργου επηρεάζονται από πολλές παραμέτρους. Για αυτόν το λόγο οι ερευνητές έχουν εστιάσει και σε άλλους στόχους βελτιστοποίησης.

- **Στόχοι βελτιστοποίησης που βασίζονται στη διάρκεια**

Ο κύριος στόχος είναι η ελαχιστοποίηση της συνολικής διάρκειας του έργου. Με αυτή την προσέγγιση γίνονται εφικτοί και πολλοί άλλοι παράλληλοι στόχοι όπως η μείωση των πιθανοτήτων για καθυστέρηση της ολοκλήρωσης του έργου που ισοδυναμεί με πρόσθετα κόστη ή και ποινές, απελευθέρωση πόρων για μελλοντικά έργα κλπ.

- **Στόχοι βελτιστοποίησης που βασίζονται στους πόρους**

Οι στόχοι αυτοί αφορούν γνωστά προβλήματα όπως το πρόβλημα όπου η ποσότητα των ανανεώσιμων πόρων πρέπει υπολογιστεί έτσι ώστε να καλύπτουν τις ανάγκες του έργου τόσο για την εκπλήρωση των προθεσμιών, όσο και την ελαχιστοποίηση του κόστους που προέρχονται από την διαθεσιμότητα των πόρων. Τέτοια κόστη μπορεί να είναι υπερωρίες αν οι πόροι είναι εργατικό δυναμικό, κόστη αποθεματοποίησης, κόστη που προέρχονται από έκτακτες παραγγελίες κλπ.

Το πρόβλημα αυτό καθώς και το πρόβλημα της εξομάλυνσης πόρων (resource leveling problem), εντάσσονται στην ευρύτερη κατηγορία των προβλημάτων κατανομής πόρων (resource allocation problems). Η εξομάλυνση της χρήσης των πόρων συγκαταλέγεται στις κορυφαίες προκλήσεις που καλούνται να αντιμετωπίσουν οι διαχειριστές έργων, διότι οι αυξομειώσεις στην απασχόληση ενός

πόρου κατά κανόνα αυξάνουν το κόστος, πλήττουν την ποιότητα του παραδοτέου, και υπονομεύουν την έγκαιρη παράδοση του έργου.

- **Στόχοι βελτιστοποίησης που βασίζονται στην ποιότητα**

Αρκετοί ερευνητές έχουν ενσωματώσει στις αντικειμενικές συναρτήσεις τους τον παράγοντα της ποιότητας. Με την αύξηση της ποιότητας επιπλέον κέρδη μπορεί να επιτευχθούν. Αυτή η αύξηση προέρχεται συνήθως από αύξηση του κόστους του έργου και για αυτόν τον λόγο πρέπει να δημιουργηθεί ένας μηχανισμός που να ελαχιστοποιεί αυτήν την αναλογία.

ΕΙΚΟΝΑ 13 THE IRON TRIANGLE FOR SCOPE

- **Στόχοι βελτιστοποίησης που βασίζονται στη χρηματοοικονομική ανάλυση**

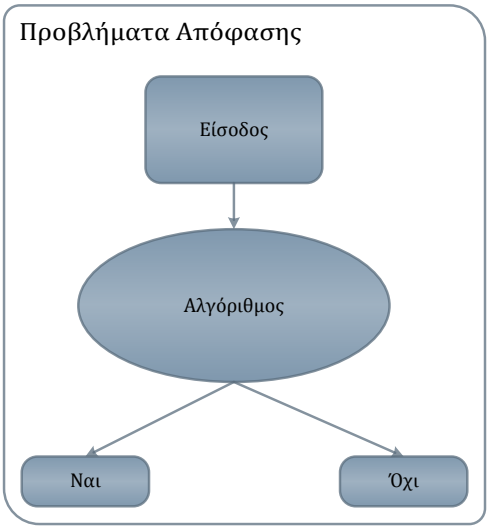
Αυτοί οι στόχοι συνδέονται με τις εξερχόμενες και εισερχόμενες ταμειακές ροές που δημιουργούνται κατά την εκτέλεση του έργου. Το αποτέλεσμα είναι η μεγιστοποίηση της καθαρής παρούσας αξίας (NPV) του έργου (Demeulemeester and Herroelen, 2002).

3.5 Ο χρονοπρογραμματισμός έργων υπό συνθήκες περιορισμένων πόρων ως NP-hard πρόβλημα

Γενικά, τα προβλήματα προγραμματισμού αποτελούν ευρύτερο πρόβλημα των συνδυαστικών προβλημάτων (combinatorial search problem). Τα συνδυαστικά προβλήματα Π μπορούν να οριστούν ως ένα ζεύγος (I, A), όπου I είναι μία περίπτωση του προβλήματος (π.χ. ένα πεπερασμένο σύνολο παραμέτρων) με συγκεκριμένες τιμές, και A είναι η απάντηση (λύση) στο πρόβλημα. Ανάμεσα στα

προβλήματα αυτά διακρίνονται δύο υποπροβλήματα: Τα **προβλήματα βελτιστοποίησης** και τα **προβλήματα απόφασης**.

Προβλήματα απόφασης (decision problems) είναι τα προβλήματα, όπου η απάντηση μπορεί να είναι ναι ή όχι .



ΕΙΚΟΝΑ 14 ΠΡΟΒΛΗΜΑΤΑ ΑΠΟΦΑΣΗΣ

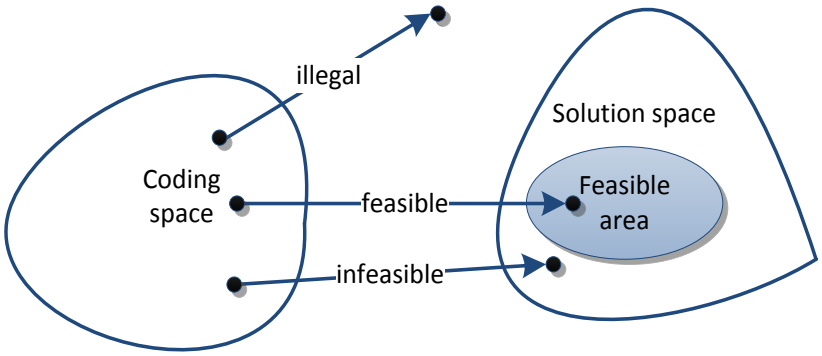
3.5.1 Προβλήματα βελτιστοποίησης

Στα μαθηματικά, ο όρος βελτιστοποίηση αναφέρεται στην επίλυση ενός προβλήματος στο οποίο, για μια συνάρτηση $f(x)$ αναζητείται με συστηματικό τρόπο, η τιμή εκείνη της μεταβλητής x για την οποία η συνάρτηση φ μεγιστοποείται ή ελαχιστοποιείται. Ένα πρόβλημα βελτιστοποίησης μπορεί αν ορισθεί ως εξής:

Για μια συνάρτηση $f: A \to R$ βρες το στοιχείο x^* του χώρου A για το οποίο $f(x^*) \leq f(x)$ για κάθε $x \in A$.

Γενικά, ένα πρόβλημα μεγιστοποίησης μπορεί να μεταφραστεί ως πρόβλημα ελαχιστοποίησης, εάν σε αυτό αντιστραφεί η συνάρτησή του. Οπότε όλα τα προβλήματα βελτιστοποίησης μπορούν να μεταφραστούν ως προβλήματα ελαχιστοποίησης. Ο χώρος A είναι το υποσύνολο του Ευκλείδειου χώρου R^n. Ο χώρος αυτός ονομάζεται σύνολο αναζήτησης (search space), και τα στοιχεία που τον απαρτίζουν ονομάζονται υποψήφιες λύσεις. Το υποσύνολο B του A, $B \subseteq A$, ονομάζεται σύνολο αποδεκτών λύσεων (feasible space) και τα στοιχεία που το απαρτίζουν αποδεκτές λύσεις (feasible solutions). Τα προβλήματα βελτιστοποίησης αποτελούνται από τη συνάρτηση f, η οποία ονομάζεται αντικειμενική συνάρτηση (objective function) ή συνάρτηση κόστους (cost function), και από ένα σύνολο

περιορισμών. Βέλτιστη λύση (optimal solution) ονομάζουμε την αποδεκτή λύση, που ελαχιστοποιεί την αντικειμενική μας συνάρτηση. Σημαντικό πρόβλημα που παρουσιάζουν οι διάφοροι μέθοδοι βελτιστοποίησης είναι ο εγκλωβισμός σε ένα σύνολο λύσεων που δεν είναι βέλτιστες.



ΕΙΚΟΝΑ 15 ΠΕΡΙΟΧΗ ΑΝΑΖΗΤΗΣΗΣ ΛΥΣΕΩΝ

3.5.2 Αλγόριθμος και πολυπλοκότητα

Τα προβλήματα τύπου P είναι τα προβλήματα απόφασης που μπορούν να επιλυθούν από την Μηχανή Τουρινγκ[4], χρησιμοποιώντας πολυωνυμική ποσότητα υπολογιστικού χρόνου.

Τα προβλήματα τύπου NP είναι τα προβλήματα απόφασης που μπορούν να επιλυθούν από την NDTM [5] σε πολυωνυμικό χρόνο. Παράδειγμα δύσκολων προβλημάτων (NP), τα οποία δεν μπορούν να λυθούν με τον παραδοσιακό τρόπο είναι τα NP (Polynomial time) προβλήματα. Υπάρχουν πολλά θέματα για τα οποία γνωρίζουμε γρήγορους (πολυωνυμικούς) αλγορίθμους αλλά και πολλά άλλα που δεν είναι δυνατό να λυθούν αλγοριθμικά. Για κάποια προβλήματα δε, έχει αποδειχθεί ότι δεν είναι δυνατό να επιλυθούν σε πολυωνυμικό χρόνο. Ένα πρόβλημα απόφασης

[4] Η μηχανή Τούρινγκ (Turing machine) είναι μια βασική αφηρημένη μηχανή που μεταχειρίζεται σύμβολα, η οποία, παρ' όλη την απλότητά της, μπορεί να προσαρμοστεί έτσι ώστε να προσομοιώσει τη λογική οποιουδήποτε αλγορίθμου. Οι μηχανές Τούρινγκ περιγράφηκαν το 1936 από τον Άλαν Τούρινγκ. Ενώ σχεδιάστηκαν για να είναι τεχνικά εφικτές, οι μηχανές Τούρινγκ δεν προορίζονταν να είναι πρακτική υπολογιστική τεχνολογία, αλλά ένα νοητό πείραμα για τα όρια των μηχανικών υπολογισμών. Έτσι, δεν κατασκευάστηκαν στην πραγματικότητα. Η μελέτη των αφηρημένων τους ιδιοτήτων φανερώνει πολλές αρχές της επιστήμης υπολογιστών και της θεωρίας πολυπλοκότητας.

[5] Η διαφορά μεταξύ στη ντετερμινιστική και μη ντετερμινιστική μηχανή (non-deterministic Turing machine) (NTM), Turing έγκειτε στους κανόνες που διέπουν τη λύση ενός προβλήματος. Η μη ντετερμινιστική μηχανή μπορεί να ορίζει περισσότερες από μία ενέργειες για μια δεδομένη κατάσταση, ενώ η ντετερμινιστική μηχανή ορίζει μόνο μία ενέργεια για κάθε κατάσταση.

είναι τύπου NP-πλήρες εάν είναι τύπου NP και κάθε πρόβλημα τύπου NP έχει λύση (απάντηση) “ναι” μόνο και μόνο όταν η λύση του NP-πλήρους είναι “ναι” επίσης. Υπάρχουν πολλά σημαντικά θέματα, για τα οποία είναι πολύ δύσκολο να βρεθεί μια λύση, αλλά μόλις βρεθεί, είναι εύκολο να ελεγχθεί. Αυτό το γεγονός οδήγησε στα NP-πλήρη προβλήματα (NP-complete). Το NP σημαίνει μη ντετερμινιστικό πολυωνυμικά και δηλώνει ότι είναι δυνατό να μαντέψουμε τη λύση (με κάποιο μη ντετερμινιστικό αλγόριθμο) και να την ελέγξουμε στη συνέχεια, και τα δύο σε πολυωνυμικό χρόνο. Έτσι, εάν είχαμε ένα μηχάνημα που μπορεί να μαντέψει, θα ήμασταν σε θέση να βρούμε μια λύση σε κάποιο λογικό χρόνο.

Ένα πρόβλημα απόφασης δεν είναι υπολογιστικά δυσκολότερο να επιλυθεί από ένα πρόβλημα βελτιστοποίησης. Ένα πρόβλημα βελτιστοποίησης είναι NP-Hard (non-deterministic polynomial-time hard) όταν το αντίστοιχο πρόβλημα απόφασης είναι τύπου NP-πλήρες (Blazewicz, 2007).

4 Ορισμός του υπό μελέτη προβλήματος

Στην παρούσα εργασία αναπτύσσονται δύο μετά-ευρετικοί αλγόριθμοι για την επίλυση του προβλήματος χρονοπρογραμματισμού έργων. Οι μετά-ευρετικοί αλγόριθμοι αποτελούν μια νέα οικογένεια αλγορίθμων, οι οποίοι έχουν εφαρμοστεί με επιτυχία σε διάφορα προβλήματα βελτιστοποίησης.

Στόχος της διπλωματικής ήταν η βελτιστοποίηση χρονοπρογραμμάτων και η μελέτη της διακριτής σχέσης χρόνου-κόστους. Αρχικά μελετήθηκε το πρόβλημα προγραμματισμού δραστηριοτήτων υπό περιορισμένους πόρους και πολλαπλούς τρόπους εκτέλεσης (Multi-Mode Resource-Constrained Project Scheduling Problem) (MRCPSP). Το πρόβλημα αυτό αποτελεί γενικευμένη εκδοχή του γενικού προβλήματος RCPSP, όπου κάθε δραστηριότητα μπορεί να εκτελεσθεί με έναν από τους πολλούς τρόπους, δηλαδή με έναν από τους πολλούς συνδυασμούς διάρκειας και απαιτούμενων πόρων. Δημιουργήθηκε πρόγραμμα του οποίου τα αποτελέσματα συγκρίθηκαν τα βέλτιστα από τα ήδη υπάρχοντα που παρουσιάζονται στη βιβλιογραφία.

Η μελέτη του προβλήματος προγραμματισμού δραστηριοτήτων όταν υπάρχουν περιορισμένοι διαθέσιμοι πόροι, σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων και εξάρτηση μεταξύ χρόνου και κόστους σε διακριτή μορφή (DTCTP) στη βιβλιογραφία αναπτύσσεται ως πρόβλημα προγραμματισμού με έναν τρόπος εκτέλεσης (single-mode), παρόλα αυτά θεωρήθηκε σκοπιμότερο να μελετηθεί ως πρόβλημα με πολλαπλούς τρόπους εκτέλεσης (multi-mode) καθώς στα προβλήματα προγραμματισμού που συναντώνται στην πραγματική ζωή κάθε δραστηριότητα μπορεί να εκτελεσθεί με πολλαπλούς τρόπους που έχουν διαφορετικό συνδυασμό ποσότητας απαιτούμενων πόρων και διάρκεια. Δεδομένου ότι η διάρκεια μιας δραστηριότητας μπορεί να μειωθεί διαθέτοντας περισσότερους πόρους για την πραγματοποίησή της, άρα αυξάνοντας το άμεσο κόστος της, κατά το χρονοπρογραμματισμό ενός έργου ανακύπτει συχνά ανάγκη της επιλογής των κατάλληλων πόρων για την εκτέλεση των δραστηριοτήτων του έργου. Στην εργασία προτείνεται ένας αλγόριθμος για την κατασκευή της καμπύλης συσχέτισης χρόνου-κόστους, σε έργα με διακριτούς συνδυασμούς χρόνου-κόστους στις δραστηριότητες. Ο αλγόριθμος είναι ένας γενετικός αλγόριθμος, ο οποίος χρησιμοποιεί τη σειριακή μέθοδο παραγωγής χρονοπρογραμμάτων (SGS) για τη δημιουργία του αρχικού πληθυσμού. Έτσι μοντελοποιήθηκε το πρόβλημα σε MRC-DTCTP και έπειτα προσαρμόστηκε κατάλληλος γενετικός αλγόριθμος για τη βελτιστοποίηση της σχέσης χρόνου-κόστους ενός έργου.

Για τη μελέτη του κόστους δημιουργήθηκε μια συνάρτηση κόστους η οποία έχει ως αντικείμενο κόστους τις δραστηριότητες που απαρτίζουν το έργο και αναλύεται σε άμεσο και έμμεσο κόστος. Μέσω της συνάρτησης αυτής και της βελτιστοποίησης της συνολικής διάρκειας του χρόνου δημιουργούνται αποδοτικοί συνδυασμοί χρόνου-κόστους για κάθε έργο. Τέλος, ένα έργο κτηματογράφησης χρησιμοποιείται ως ένα πρακτικό παράδειγμα για να καταδείξει την πρακτικότητα και την αποδοτικότητα του αλγορίθμου.

5 Αλγόριθμος Επίλυσης Προβλήματος

5.1 Παρουσίαση Χρωμοσώματος

Εφόσον το MRC-DTCTP είναι συνδυασμός των προβλημάτων MRCPSP και DTCTP το χρωμόσωμα πρέπει να περιέχει τις εξής πληροφορίες:

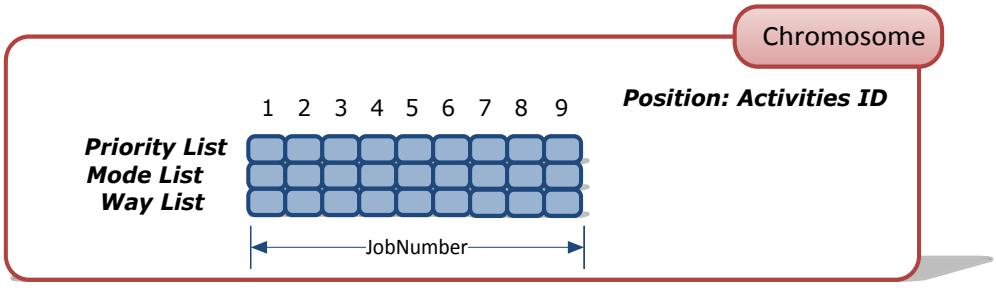
1. Τις τιμές προτεραιότητας (***Priority list***) (Gen and Cheng, 2000).
2. Τον τρόπο εκτέλεσης (***Mode list***) (Hartmann, 2001)
3. Αν η δραστηριότητα θα εκτελεσθεί σε μειωμένη διάρκεια ή όχι (***Way list***)

Και οι τρεις λίστες έχουν μέγεθος όσο και ο αριθμός των δραστηριοτήτων. Η πρώτη λίστα τα γονίδια παίρνουν τιμές v_j από το 1 μέχρι J. Η απόδοση τιμών γίνεται τυχαία. Όσο μεγαλύτερη τιμή έχει μια δραστηριότητα τόσο μεγαλύτερες πιθανότητες έχει να προγραμματιστεί πρώτη όταν επιλεγούν παραπάνω από μία δραστηριότητες να προγραμματιστούν στην επόμενη φάση.

Στη δεύτερη λίστα περιέχονται οι πληροφορίες m_j σχετικά με τον τρόπο εκτέλεσης κάθε δραστηριότητας. Σε αυτά τα γονίδια περιέχεται ο αριθμός που αναδεικνύει τον τρόπο που μπορεί να εκτελεσθεί μια δραστηριότητα ανάμεσα από τις δυνατές επιλογές.

Η τελευταία λίστα w_j περιέχει είτε τον αριθμό 0 είτε τον 1. Ο αριθμός 0 δείχνει ότι η δραστηριότητα θα εκτελεσθεί στην κανονική (normal) διάρκειά της, ενώ ο αριθμός 1 ότι θα εκτελεσθεί στην συμπιεσμένη (crashing) διάρκειά της.

Στόχος είναι να γίνει συγχρόνως συγχρόνως χειρισμός και των τριών λιστών, καθώς δεν υπάρχει καλός τρόπος εκτέλεσης μιας δραστηριότητας, για παράδειγμα, εάν δεν γνωρίζουμε πως αυτός επιδρά στον χρόνο εκτέλεσής της. Το χρωμόσωμα συνεπώς θα έχει την μορφή που παρουσιάζεται στην Εικόνα 16.

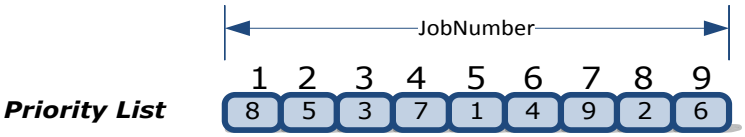


ΕΙΚΟΝΑ 16 ΔΟΜΗ ΧΡΩΜΟΣΩΜΑΤΟΣ

Το γονίδιο περιέχει δύο ειδών πληροφορίες: Τον τόπο (locus), δηλαδή τη θέση που είναι τοποθετημένο το γονίδιο στη δομή του χρωμοσώματος, και το αλληλόμορφο (allele), την τιμή, δηλαδή, που παίρνει το γονίδιο. Έτσι υπάρχουν δύο δυνατοί τρόποι να αναπαρασταθεί μια δραστηριότητα: μέσω του τόπου ή μέσω του αλληλόμορφου. Για την εργασία, ο τόπος χρησιμοποιείται για να δηλώσει το ID μιας δραστηριότητας, όπως φαίνεται στην Εικόνα 16 ενώ το αλληλόμορφο είναι η τιμή της αντίστοιχης δραστηριότητας, η οποία ανά λίστα εκφράζει την προτεραιότητα, τον τρόπο εκτέλεσης και το αν θα εκτελεσθεί σε κανονική ή συμπιεσμένη διάρκεια αντίστοιχα.

5.2 *Προγραμματισμός βάσει κανόνα προτεραιότητας (Priority Value Based Scheduling)*

Στην παρούσα εργασία χρησιμοποιήθηκε η κωδικοποίηση βάση τιμών προτεραιότητας. Η πρώτη λίστα του χρωμοσώματος περιλαμβάνει τιμές που συμβολίζουν τις τιμές προτεραιότητας που θα χρησιμοποιηθούν εάν προκύψουν δύο ή παραπάνω δραστηριότητες για προγραμματισμό στο *σύνολο απόφασης* (E_h).



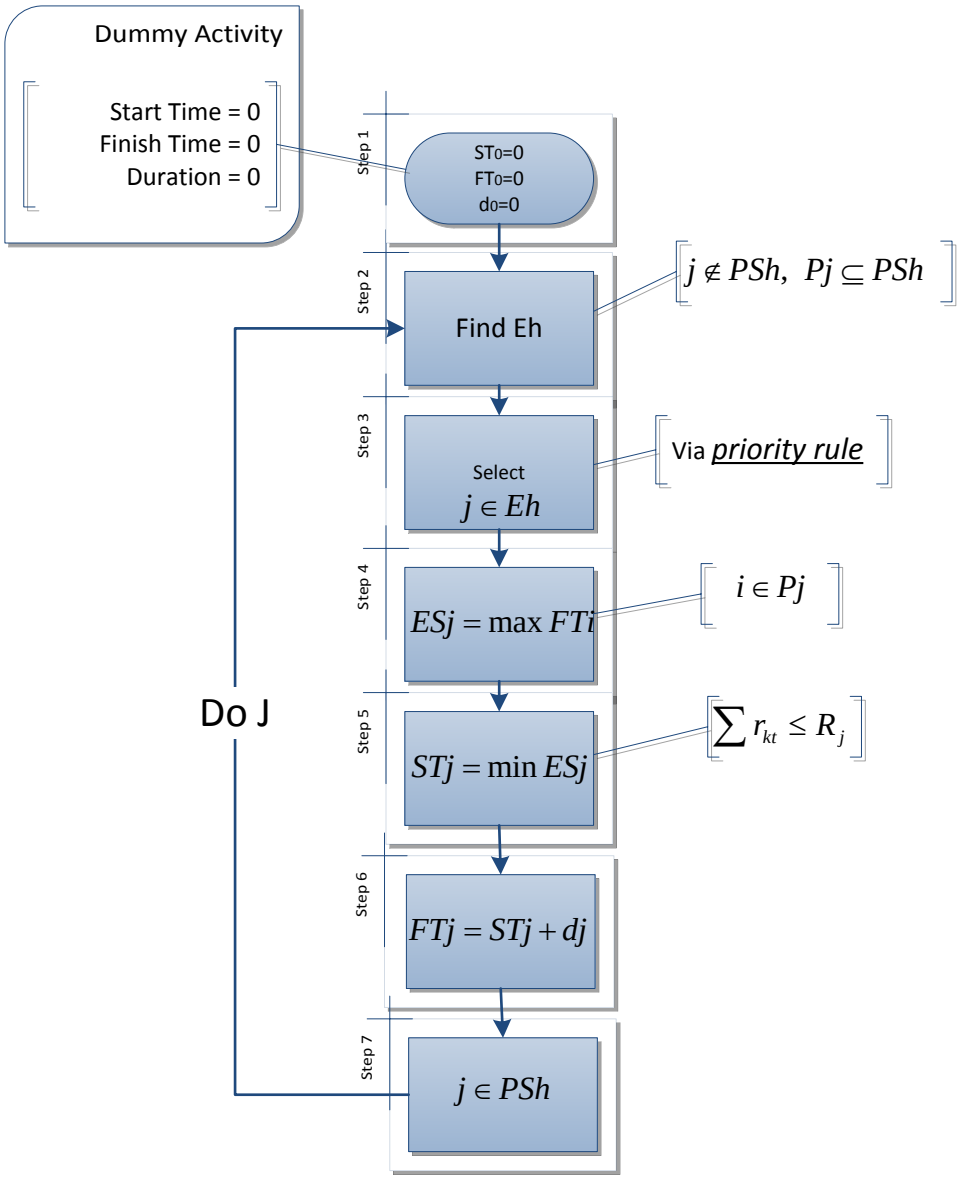
ΕΙΚΟΝΑ 17 ΛΙΣΤΑ ΠΡΟΤΕΡΑΙΟΤΗΤΑΣ

Για παράδειγμα, εάν οι δραστηριότητες 3 και 6 είναι έτοιμες για προγραμματισμό, από άποψη περιορισμών προτεραιότητας, τότε αυτή που θα προγραμματιστεί πρώτη είναι η 6, καθώς έχει μεγαλύτερη τιμή προτεραιότητας. Στην παρούσα εργασία χρησιμοποιήθηκε η σειριακή μέθοδος παραγωγής χρονοπρογράμματος, η οποία σύμφωνα με τον Kolisch (Kolisch, 1996) ανταποκρίνεται καλύτερα σε προβλήματα με πολλές δραστηριότητες, βάσει αυτής της αναπαράστασης. Επιπρόσθετα, η παράλληλη μέθοδος μπορεί να αποκλείσει τη βέλτιστη λύση ψάχνοντας σε λάθος χώρο του προβλήματος, ενώ με τη σειριακή μέθοδο πάντα θα βρίσκεται η βέλτιστη λύση.

5.3 *Σειριακή Μέθοδος Παραγωγής Χρονοπρογραμμάτων (Serial Schedule Generation Scheme)*

Αυτή η μέθοδος αποτελείται από τόσο βήματα όσα και οι δραστηριότητες που θέλουμε να προγραμματίσουμε. Επίσης οι δραστηριότητες χωρίζονται σε δύο ομάδες. Στη μία ομάδα ανήκουν οι δραστηριότητες που έχουν ήδη προγραμματιστεί

και επομένως αποτελούν το ημιτελές πρόγραμμα (PS_h– Partial Schedule) και στη δεύτερη ομάδα (E_h - Decision Set) ανήκουν οι δραστηριότητες των οποίων οι προαπαιτούμενες δραστηριότητες (predecessors- P_j) έχουν προγραμματιστεί, δηλαδή ανήκουν στο ημιτελές πρόγραμμα, και είναι σε σειρά να προγραμματιστούν. Σε κάθε βήμα μόνο μία δραστηριότητα προγραμματίζεται και αφού επιλεγεί ποιά θα είναι αυτή περνά από την δεύτερη ομάδα στην πρώτη, στο ημιτελές πρόγραμμα πλέον. Η επιλογή αυτή γίνεται βάση κάποιου κανόνα προτεραιότητας. Ο αλγόριθμος αυτός σταματά όταν έχουν προγραμματιστεί όλες οι δραστηριότητες, δηλαδή όταν η πρώτη ομάδα (decision set) είναι κενή. Παρακάτω φαίνεται η διαδικασία που περιγράφεται μέσω ενός διαγράμματος ροής (εικόνα 18). Πρώτα προγραμματίζεται η πλασματική δραστηριότητα και σηματοδοτεί την έναρξη του έργου. Στο επόμενο βήμα ο αλγόριθμος βρίσκει τις δραστηριότητες των οποίων οι προαπαιτούμενες δραστηριότητες έχουν προγραμματιστεί. Αυτές ανήκουν πλέον στο ημιτελές πρόγραμμα E . Μέσω ενός κανόνα προτεραιότητας επιλέγεται ποια θα προγραμματιστεί. Ο χρόνος νωρίτερης έναρξης (Early Start – ES) της ορίζεται ως ο χρόνος ολοκλήρωσης της δραστηριότητας i που ανήκει στο P_j, δηλαδή στο σύνολο με τις προαπαιτούμενες δραστηριότητές της, και έχει τον μεγαλύτερο χρόνο ολοκλήρωσης. Ο χρόνος έναρξής της (Start Time – ST) ορίζεται ως ο μικρότερος χρόνος νωρίτερης έναρξης όπου δεν υπάρχει παράβαση του περιορισμού της διαθεσιμότητας των πόρων σε εκείνη την περίοδο. Παρακάτω περιγράφουμε τη διαδικασία με ένα παράδειγμα.



ΕΙΚΟΝΑ 18 ΔΙΑΓΡΑΜΜΑ ΒΗΜΑΤΩΝ ΣΕΙΡΙΑΚΗΣ ΜΕΘΟΔΟΥ

Παράδειγμα Σειριακή Μέθοδος Παραγωγής Χρονοπρογραμμάτων

ΠΙΝΑΚΑΣ 5 ΠΑΡΑΔΕΙΓΜΑ ΕΡΓΟΥ

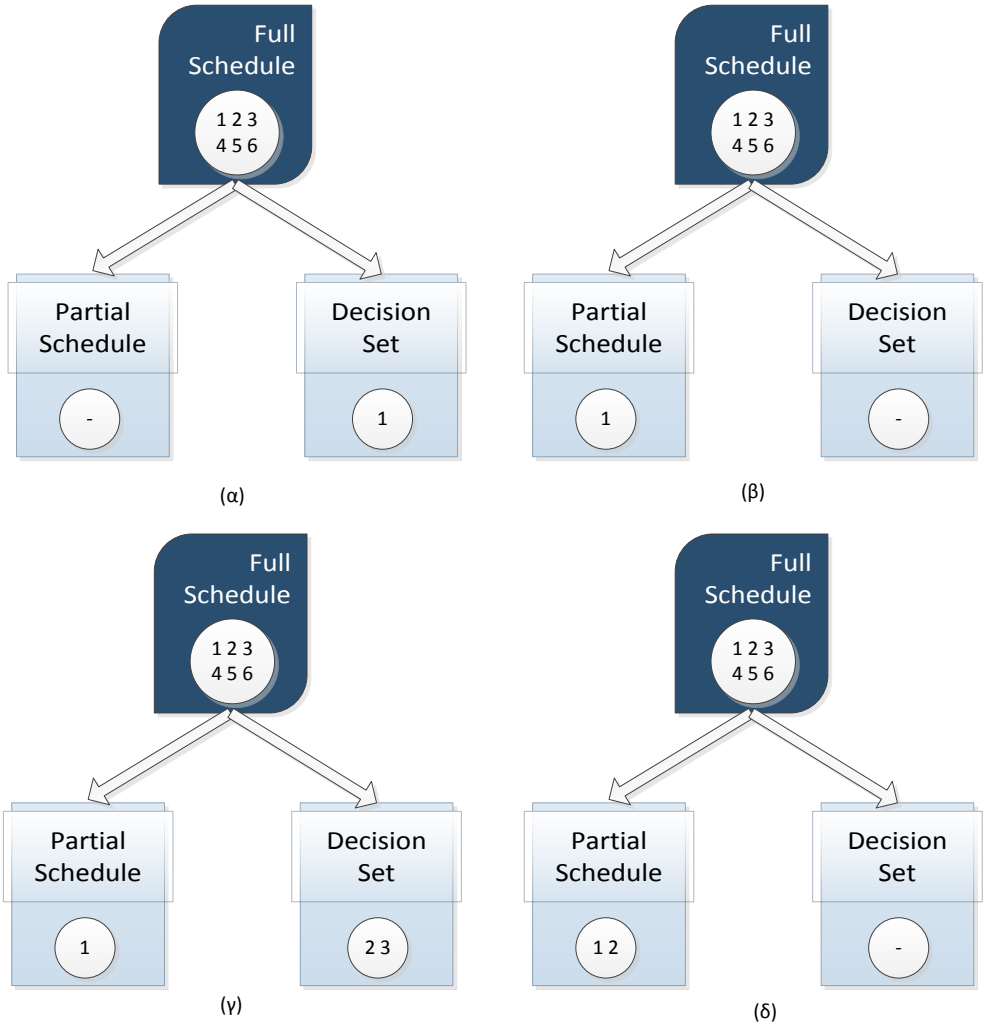
Δραστηριότητα	Διάρκεια Εκτέλεσης	Απαιτούμενοι Πόροι	Προαπαιτούμενες Δραστηριότητες
1	0	-	-
2	1	2	1
3	4	1	1
4	5	5	2,3
5	2	4	2,3

6	0	-	-

Για το παράδειγμα θα χρησιμοποιήσουμε τα δεδομένα από τον πίνακα 5. Θεωρούμε ως δεδομένο ότι το σύνολο διαθεσιμότητας του πόρου είναι ίσο με 6, όπως επίσης και μία λίστα προτεραιότητας η οποία είναι η εξής:

$$V(j) \in (2,4,1,6,5,3) \qquad j = 1, \dots, J.$$

Όπως προαναφέραμε οι δραστηριότητες χωρίζονται σε δύο ομάδες. Αυτές φαίνονται σε κάθε επιμέρους σχήμα (Partial Schedule, Decision Set).



ΕΙΚΟΝΑ 19 ΒΗΜΑΤΑ ΣΕΙΡΙΑΚΗΣ ΜΕΘΟΔΟΥ

Βήμα 1ο

Όπως έχει οριστεί το πρόβλημα του RCPSP, η πρώτη και η τελευταία δραστηριότητα είναι πλασματικές. Δεν έχουν διάρκεια ούτε απαίτηση σε πόρους. Η

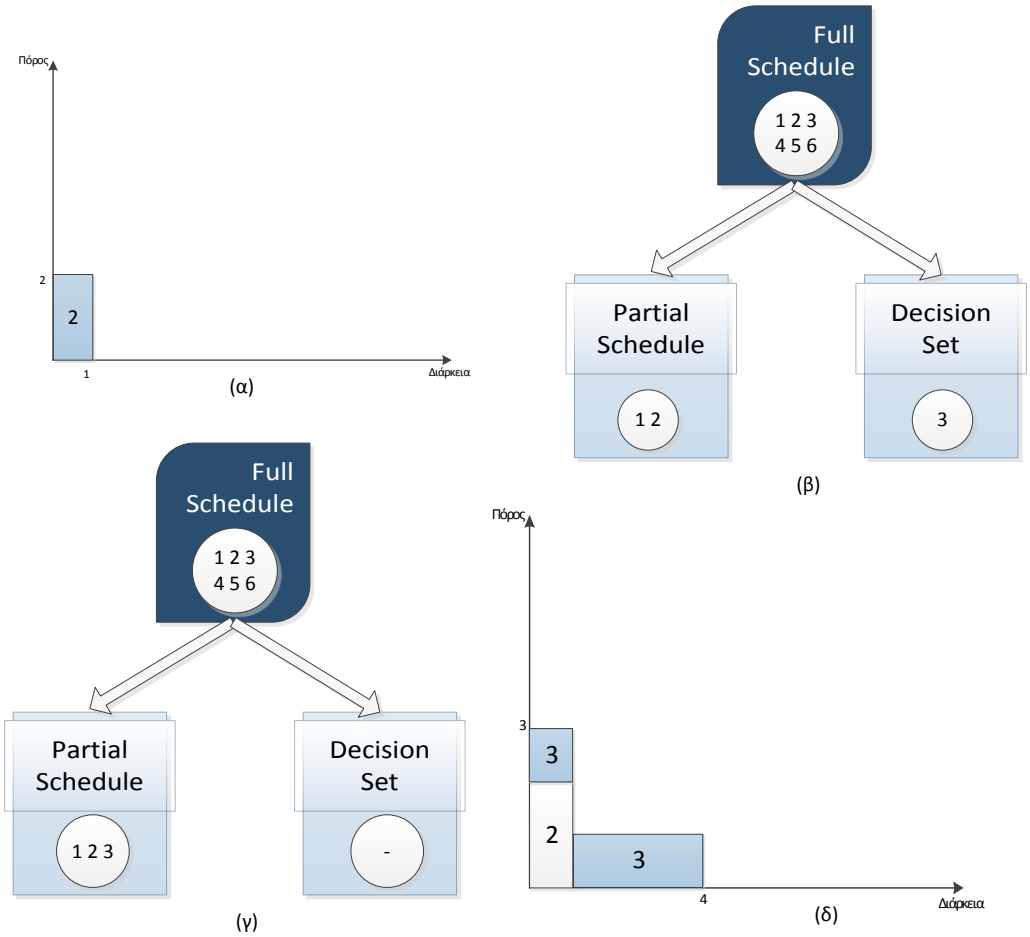
εμφάνισή τους σηματοδοτεί την έναρξη και την λήξη του έργου αντίστοιχα. Έτσι πρώτα προγραμματίζεται η δραστηριότητα 1 (εικόνα 19α, 19β).

Βήμα 2ο

Στο 2ο βήμα προγραμματίζεται η δραστηριότητα της οποίας οι προαπαιτούμενες έχουν προγραμματιστεί. Μέχρι τώρα μόνο η 1 έχει προγραμματιστεί οπότε υποψήφιες είναι η 2 και η 3. Αυτές περνάνε στο *Decision Set* (εικόνα 19γ).

Η επιλογή για το ποια δραστηριότητα εκ των δύο θα προγραμματιστεί γίνεται βάσει της λίστας προτεραιότητας. Η δραστηριότητα 2 έχει μεγαλύτερη τιμή προτεραιότητας $V(2) = 4$ έναντι της 3 που έχει $V(3) = 1$ οπότε αυτή είναι η επόμενη για προγραμματισμό (εικόνα 19δ).

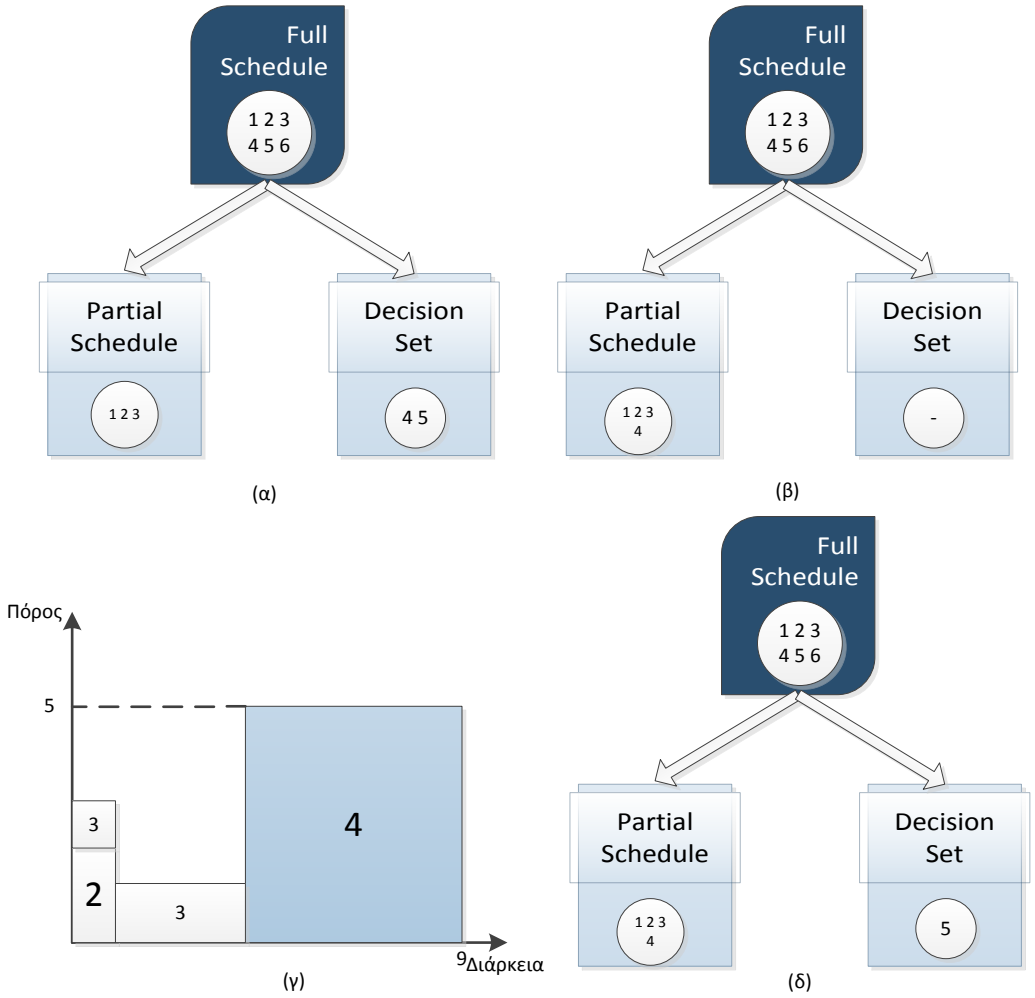
Η δραστηριότητα 2, λοιπόν, περνάει στο ημιτελές πρόγραμμα (PS_h). Πλέον εφόσον η δραστηριότητα 2 απαιτεί πόρους για την εκτέλεσή της μπορούμε να απεικονίσουμε την εκτέλεσή της σε ένα διαξονικό σύστημα συντεταγμένων με κάθετο άξονα τη χρησιμοποίηση των πόρων και οριζόντιο τη διάρκεια εκτέλεσης των δραστηριοτήτων (εικόνα 20α).



ΕΙΚΟΝΑ 20 ΒΗΜΑΤΑ ΣΕΙΡΙΑΚΗΣ ΜΕΘΟΔΟΥ

Βήμα 3ο

Εφόσον προγραμματίστηκε η 2, μόνον η 3 βρίσκεται πλέον στο *Decision Set* (E_h) (εικόνα 20β). Οπότε είναι αυτή που θα προγραμματιστεί επόμενη (εικόνα 20γ). Παρατηρούμε ότι η εκτέλεση της δραστηριότητας 3 δεν υπερβαίνει τη μέγιστη διαθεσιμότητα του πόρου, η οποία είναι ίση με 6 μονάδες (εικόνα 20δ).



ΕΙΚΟΝΑ 21 ΒΗΜΑΤΑ ΣΕΙΡΙΑΚΗΣ ΜΕΘΟΔΟΥ

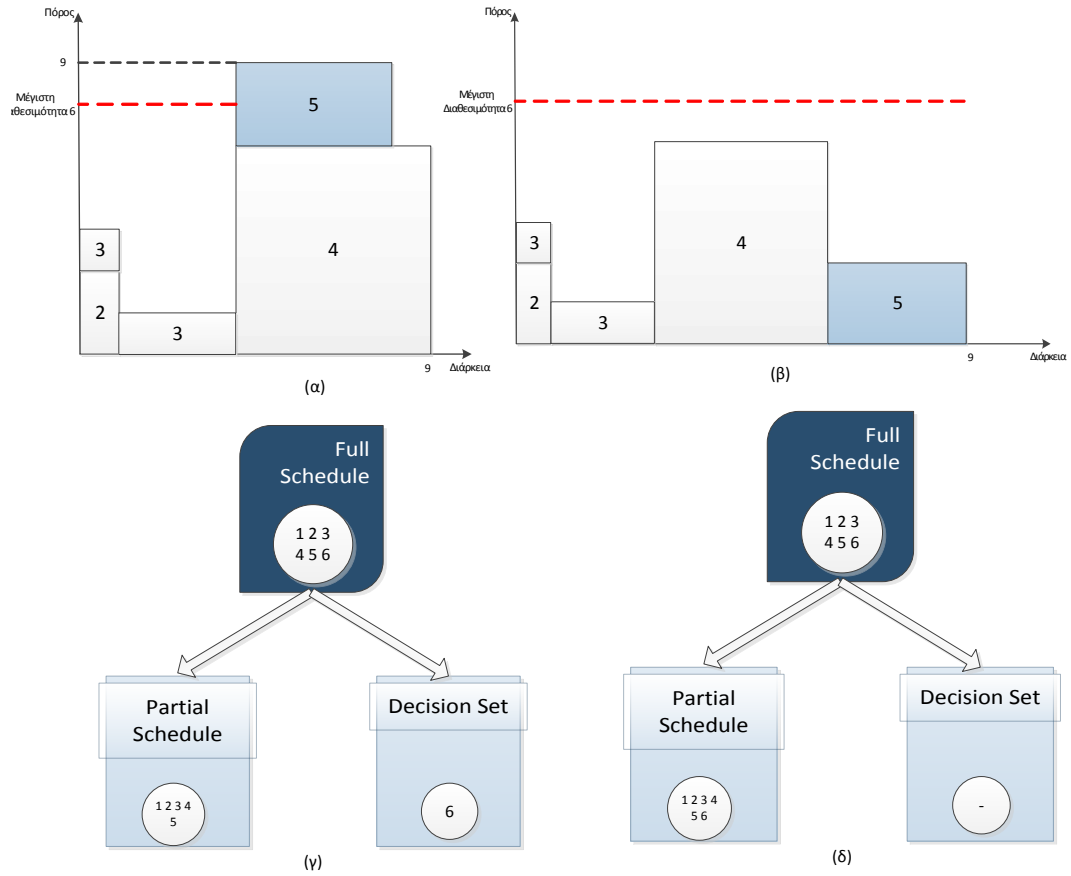
Βήμα 4ο

Οι δραστηριότητες που ανήκουν στο *Decision Set* (E_h) είναι η 4 και η 5, αφού οι προαπαιτούμενες δραστηριότητές τους έχουν προγραμματιστεί ($2,3 \in PS_h$) (εικόνα 21α). Προγραμματίζεται η 4, η οποία έχει μεγαλύτερη τιμή προτεραιότητας από την 5 ($V(4) = 6 > V(5) = 5$). Η δραστηριότητα 4 περνά από το Decision Set (E_h) στο PS_h (εικόνα 21β). Η δραστηριότητα 4 ξεκινά όταν έχει ολοκληρωθεί η αργότερη από τις προαπαιτούμενες δραστηριότητές της, δηλαδή η 3 (εικόνα 21γ).

Βήμα 5ο

Επόμενη για προγραμματισμό είναι η δραστηριότητα 5 (εικόνα 21δ). Η νωρίτερη έναρξή της ορίζεται ως ο αργότερος χρόνος ολοκλήρωσης των προαπαιτούμενων δραστηριοτήτων της, δηλαδή $FT_3 = 4$ μονάδες χρόνου. Όμως

όπως παρατηρούμε από το σχήμα με την έναρξη παραβαίνετε ο περιορισμός της διαθεσιμότητας του πόρου (εικόνα 22α). Για την έναρξη αυτή απαιτούνται συνολικά 9 μονάδες από τον πόρο, ενώ οι διαθέσιμοι κάθε περίοδο είναι 6. Οπότε η έναρξη της δραστηριότητας μετακινείται μπροστά στο χρόνο μέχρι να πληρείται ο περιορισμός.



ΕΙΚΟΝΑ 22 ΒΗΜΑΤΑ ΣΕΙΡΙΑΚΗΣ ΜΕΘΟΔΟΥ

Τελικά η δραστηριότητα 5 ξεκινάει, όταν τελειώσει η 4 (εικόνα 21β).

Βήμα 6ο

Στο τελευταίο βήμα, πάντα όπως έχει ορισθεί, προγραμματίζεται η πλασματική δραστηριότητα που σηματοδοτεί το τέλος του έργου (εικόνα 22γ,δ).

Στην εργασία, χρησιμοποιήθηκε η σειριακή μέθοδος για να αναπαράγει χρονοπρογράμματα χρησιμοποιώντας τα δεδομένα από το χρωμόσωμα. Στο MRC-DTCTP πρέπει να ληφθούν υπόψη όχι μόνο οι περιορισμοί διαθεσιμότητας των πόρων αλλά και οι χρονικοί ή οι περιορισμοί του κόστους του έργου. Επίσης, στο χρονοπρόγραμμα πρέπει να ληφθούν υπόψη οι επιλεγμένοι τρόποι εκτέλεσης και το αν θα εκτελεσθεί η δραστηριότητα σε απλή ή συμπιεσμένη διάρκεια. Πληροφορίες οι

οποίες λαμβάνονται επίσης από το χρωμόσωμα. Με βάση όλα αυτά δεν μπορεί να χρησιμοποιηθεί η κλασική σειριακή μέθοδος, και για αυτό μία αναθεωρημένη μέθοδος παρουσιάζεται προσαρμοσμένη στις απαιτήσεις του προβλήματος. Αυτή η μέθοδος αποκωδικοποιεί τις τρεις λίστες που παρουσιάστηκαν στο χρωμόσωμά μας. Η αναθεωρημένη σειριακή μέθοδος παρατίθεται παρακάτω υπό αλγοριθμική μορφή.

$h = 1, PS_h = \{1\}$

$While\ PS_h < J\ DO$

$Begin$

$E_h = \{j | j \notin PS_h, P_j \subseteq PS_h\}$

$j^* \leftarrow j | v(j) = max_{i \in E_h}\{v(i)\}$

$$d_{j^*} = \begin{cases} ed_{j^*m}, & x_{j^*m} = 1, \quad y_{j^*} = 1 \\ nd_{j^*m}, & x_{j^*m} = 1, \quad y_{j^*} = 0 \end{cases}$$

$ES_{j^*} = max\{f_i | i \in P_{j^*}\}$

$$s_{j^*} = min\Bigg\{ t | ES_{j^*} \leq t, r_{j*k} \leq vr_{kt}, vr_{kt} = a_k - \sum_{j \in A_t} r_{jk}, \tau = t, t+1, t+2, \dots t + d_{j*} - 1, k = 1,2, \dots, K \Bigg\}$$

$f_{j^*} = s_{j^*} + d_{j^*}$

$PS_{h+1} = PS_h \cup \{j^*\}$

$h = h + 1$

End

(Wuliang and Chengen, 2009)

6 Σχεδιασμός του Συστήματος

6.1 Αντικειμενική Συνάρτηση

Η αντικειμενική συνάρτηση η οποία καθορίζει τον στόχο βελτιστοποίησης, ονομάζεται συνάρτηση κόστους ή συνάρτηση καταλληλότητας. Η συνάρτηση κόστους δέχεται ως είσοδο ένα χρωμόσωμα και επιστρέφει έναν αριθμό που υποδηλώνει πόσο κατάλληλο είναι, δηλαδή αναθέτει μία τιμή σε κάθε χρωμόσωμα του πληθυσμού. Η τιμή αυτή αποτελεί κριτήριο για το πόσο ικανοποιητική είναι η λύση που αντιπροσωπεύει το κάθε χρωμόσωμα στη συγκεκριμένη φάση.

Ανάλογα με το πρόβλημα, η κατασκευή της συνάρτησης καταλληλότητας μπορεί να είναι από απλή, έως εξαιρετικά πολύπλοκη. Η ιδανική συνάρτηση καταλληλότητας θα πρέπει να είναι συνεχής και μονότονη. Ωστόσο αυτό σπάνια συμβαίνει, οπότε αυτό που επιζητείται είναι μια συνάρτηση καταλληλότητας που δεν έχει πολλά τοπικά μέγιστα ή ένα απομονωμένο ολικό μέγιστο. Ο γενικός κανόνας στην κατασκευή της συνάρτησης καταλληλότητας είναι ότι αυτή πρέπει να αντικατοπτρίζει ρεαλιστικά την αξία του χρωμοσώματος.

6.1.1 Ταξινόμηση του κόστους για καταλογισμό κόστους στα διάφορα αντικείμενα κόστους.

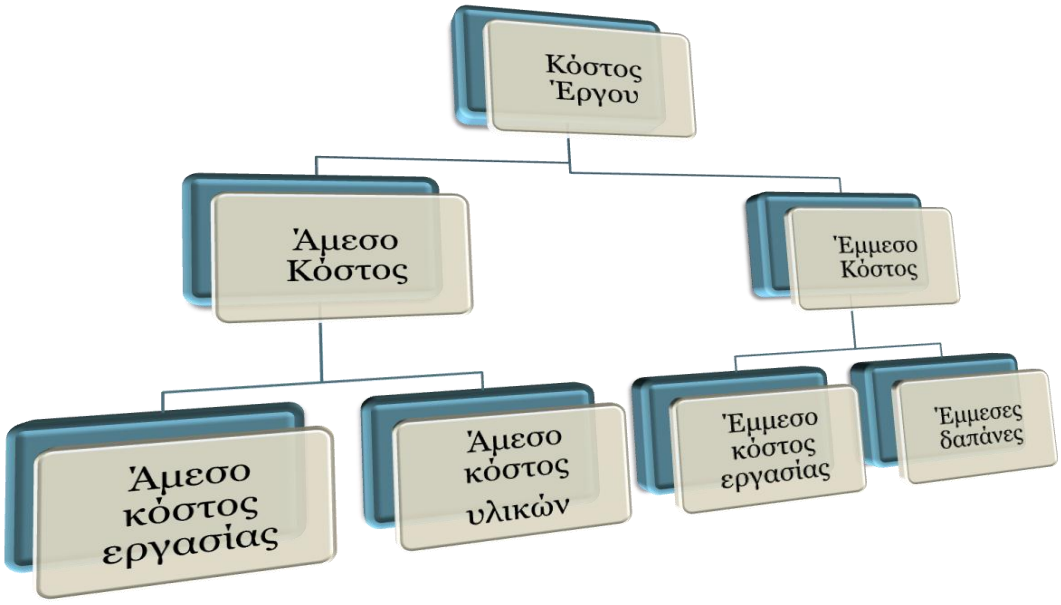
Το κόστος καταλογίζεται σε αντικείμενα για πολλούς λόγους (τιμολόγηση, μελέτες κερδοφορίας, έλεγχος δαπανών). Αντικείμενο κόστους ονομάζεται οτιδήποτε για το οποίο ζητούνται δεδομένα κόστους, όπως για παράδειγμα προϊόντα, γραμμές παραγωγή, πελάτες, θέσεις εργασίας. Σε σχέση με τον καταλογισμό στα αντικείμενα κόστους, το κόστος χαρακτηρίζεται είτε ως άμεσο είτε ως έμμεσο.

Άμεσο κόστος (direct cost) είναι το κόστος που μπορεί εύκολα και γρήγορα να εντοπιστεί στο συγκεκριμένο αντικείμενο κόστους που εξετάζουμε. Η έννοια του άμεσου κόστους εκτείνεται και πέρα από τα άμεσα υλικά και την άμεση εργασία. Για παράδειγμα, εάν μια εταιρία καταλογίζει το κόστος στα διάφορα περιφερειακά και εθνικά γραφεία πωλήσεων, τότε ο μισθός του διευθυντή πωλήσεων σε γραφείο εκτός της χώρας που εδράζεται η επιχείρηση θα είναι άμεσο κόστος για αυτό το γραφείο.

Έμμεσο κόστος (indirect cost) είναι το κόστος που δεν μπορεί να εντοπιστεί εύκολα και γρήγορα στο συγκεκριμένο αντικείμενο κόστους που εξετάζουμε. Για παράδειγμα, ένα εργοστάσιο παραγωγής σκυροδέματος παράγει πολλά είδη

σκυροδέματος. Ο μισθός του διευθυντή του εργοστασίου θα είναι έμμεσο κόστος για μια συγκεκριμένη ποικιλία σκυροδέματος. Ο λόγος είναι ότι ο μισθός του διευθυντή του εργοστασίου δεν είναι κόστος που δημιουργείται από ένα συγκεκριμένο τύπο σκυροδέματος αλλά συνέπεια της λειτουργίας όλου του εργοστασίου. Για να εντοπιστεί ένα στοιχείο κόστους σε ένα αντικείμενο κόστους όπως είναι ένα συγκεκριμένο προϊόν, πρέπει να δημιουργείται από το αντικείμενο κόστους.

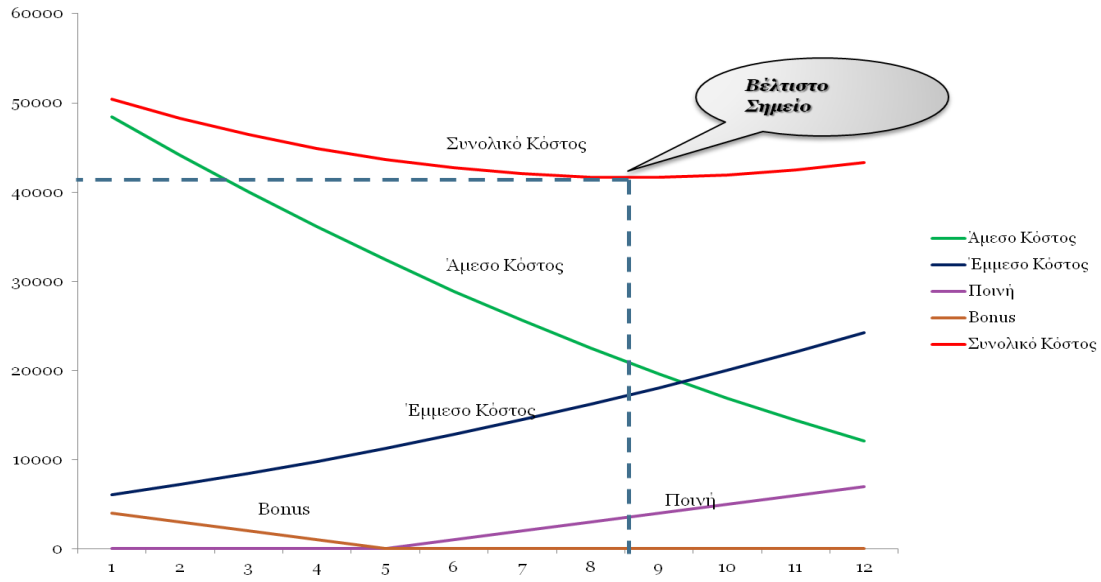
Ένα συγκεκριμένο στοιχείο κόστους μπορεί να είναι άμεσο ή έμμεσο, ανάλογα με το αντικείμενο κόστους. Ο μισθός του διευθυντή του εργοστασίου είναι έμμεσο κόστος για ένα συγκεκριμένο είδος σκυροδέματος αλλά είναι άμεσο κόστος για το τμήμα παραγωγής. Το τμήμα παραγωγής σε αυτήν την περίπτωση είναι το αντικείμενο κόστους.



ΕΙΚΟΝΑ 23 ΤΑΞΙΝΟΜΗΣΗ ΚΟΣΤΟΥΣ ΕΡΓΟΥ

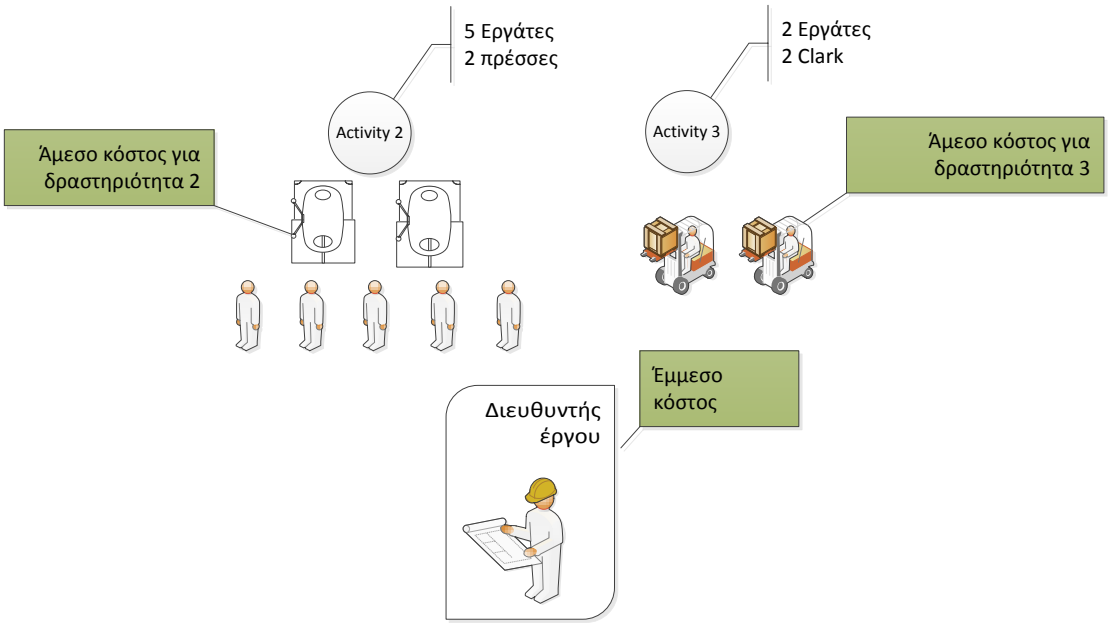
Στην περίπτωσή μας αντικείμενο κόστους είναι οι δραστηριότητες. Αυτές μπορούν όπως προαναφέραμε μα εκτελεσθούν γρηγορότερα με επιπλέον άμεσο κόστος. Έτσι, όπως συμβαίνει και στην πράξη, αυξάνοντας ή μειώνοντας πόρους όπως εργατικό δυναμικό, μηχανές, διαθέσιμο κεφάλαιο ή εγκαταστάσεις, η συνολική διάρκεια του έργου μειώνεται ή αυξάνεται αντίστοιχα. Τα επιπλέον κόστη αντισταθμίζονται από τη τα κέρδη που προέρχονται από τη μείωση του χρόνου διάρκειας του έργου. Από την άλλη εάν οι δραστηριότητες εκτελεσθούν σε αργότερο ρυθμό, μειώνονται μεν τα κόστη (άμεσα) των δραστηριοτήτων, αλλά αυξάνονται τα κόστη που προέρχονται από τη συνολική διάρκεια του έργου, δηλαδή τα έμμεσα. Επίσης μεγαλύτερη συνολική διάρκεια μπορεί να επιφέρει ποινές (εικόνα 24).

Έτσι δεδομένης μιας προθεσμίας για την ολοκλήρωση του έργου, σκοπός είναι να βρεθεί ένα πρόγραμμα το οποίο να ελαχιστοποιεί το κόστος του έργου, δηλαδή το άθροισμα των άμεσων και έμμεσων κοστών.



ΕΙΚΟΝΑ 24 ΣΧΕΣΗ ΑΜΕΣΟΥ-ΕΜΜΕΣΟΥ-ΣΥΝΟΛΙΚΟΥ ΚΟΣΤΟΥΣ

Έμμεσα κόστη μπορεί να συμπεριλαμβάνονται στην επιτήρηση κατά όλη τη διάρκεια του έργου (π.χ. διευθυντές έργου), ενοίκια για μηχανές ή εγκαταστάσεις, ενώ άμεσα είναι αυτά που μπορούν να εντοπισθούν κατευθείαν στις δραστηριότητες (εικόνα 25).



ΕΙΚΟΝΑ 25 ΠΑΡΑΔΕΙΓΜΑ ΑΜΕΣΩΝ-ΕΜΜΕΣΩΝ ΚΟΣΤΩΝ

6.1.2 Κατασκευή Συνάρτησης Καταλληλότητας (Fitness Function)

Για την κατασκευή της συνάρτησης καταλληλότητας ή αντικειμενικής συνάρτησης για το πρόβλημά πρέπει να διευκρινιστούν κάποιες μεταβλητές.

c_k: είναι το κόστος ανά μονάδα χρόνου του πόρου k, όταν η δραστηριότητα εκτελείτε σε συμπιεσμένη διάρκεια.

n_k: είναι το κόστος ανά μονάδα χρόνου του πόρου k, όταν η δραστηριότητα εκτελείτε σε κανονική διάρκεια.

u: είναι το κόστος ανά μονάδα χρόνου των συνολικών έμμεσων κοστών που υπάρχουν στο έργο.

Προφανώς ισχύει $c_k > n_k$. Αυτά τα κόστη στην παρούσα εργασία είναι ανάλογα της διάρκειας της δραστηριότητας. Ο συντελεστής αναλογίας τους ψ τέθηκε ίσος με $3{,}5$ αλλά δοθέντων συγκεκριμένων στοιχείων μπορεί να υπολογιστεί επακριβώς. Ο αριθμός αυτός επιλέχθηκε για να αυξήσει την διαφορά ανάμεσα στο κόστος της συμπιεσμένης έναντι της κανονικής διάρκειας. Έτσι ισχύει ότι:

$$c_k = \psi * n_k \qquad \forall k \in K$$

K_{jm} είναι οι τύποι πόρων όταν η δραστηριότητα j εκτελείται με τον τρόπο m.

$y_j = 1$: Η δραστηριότητα j εκτελείται σε συμπιεσμένη διάρκεια.

$$\boldsymbol{F_{c1}} = \sum_{j \in V} \sum_{m \in M_j} (x_{jm} \quad * \sum_{k \in K_{jm}} (ed_{im} * c_k))$$

$y_j = 0$: Η δραστηριότητα j εκτελείται σε κανονική διάρκεια.

$$\boldsymbol{F_{c2}} = \sum_{j \in V} \sum_{m \in M_j} (x_{jm} \quad * \sum_{k \in K_{jm}} (nd_{im} * n_k))$$

Άρα $\boldsymbol{F_c = F_{c1} + F_{c2} + u * FT_J}$

Όπως έχει προαναφερθεί το έμμεσο κόστος αφορά μόνο τη συνολική διάρκεια του έργου για αυτό και στο συνολικό κόστος προστίθεται ο όρος u επί το χρόνο ολοκλήρωσης της τελευταίας δραστηριότητας του έργου FT_J. Στην συνάρτηση αυτή προστίθεται ποινή (penalty) εάν το πρόγραμμα που αντιστοιχεί σε αυτό το άτομα υπερβεί τη μέγιστη δυνατή διάρκεια του έργου $Tmax$. Έτσι ικανοποιείται και ο περιορισμός (7) στην παράγραφο 4.4, όπου πρέπει η συνολική διάρκεια του έργου να είναι μικρότερη από μία καθορισμένη τιμή. Θέτουμε ως μη ανανεώσιμο πόρο μόνο το κόστος του έργου. Έτσι εάν ο συνολικός χρόνος εκτέλεσης του προγράμματος υπερβεί το $Tmax$, η συνάρτηση καταλληλότητας αυξάνεται όσο είναι το penalty, επομένως είναι πολύ δύσκολο να επιλεγεί από τον αλγόριθμο για να περάσει στην επόμενη γενιά μαζί με τις καλύτερες λύσεις. Με αυτόν τον τρόπο απορρίπτονται από τον αλγόριθμο τα non-feasible schedules εφόσον έχουμε πρόβλημα ελαχιστοποίησης (μείωση του συνολικού κόστους του έργου). Άρα η συνάρτηση καταλληλότητας έχει την παρακάτω μορφή:

$$\boldsymbol{F_c = F_{c1} + F_{c2} + u * FT_J + penalty}$$

Η συνάρτηση καταλληλότητας ανταποκρίνεται πλήρως στις απαιτήσεις του προβλήματος. Υπάρχει συσχέτιση του άμεσου και του έμμεσου κόστους όσο αφορά τη συνολική διάρκεια του έργου και είναι προφανής η εύρεση του κατάλληλου συνδυασμού των δύο έτσι ώστε να βρεθεί το κατάλληλο σημείο το οποίο ελαχιστοποιεί το κόστος με δεδομένη μια προθεσμία του έργου. Ο παραπάνω συνδυασμός αποτελεί το βέλτιστο σημείο με μεταβλητές το κόστος και το χρόνο όπως φαίνεται και στην εικόνα 24.

6.2 Γενετικός Αλγόριθμος

Το 1950 και το 1960 διάφοροι επιστήμονες ασχολήθηκαν με εξελικτικά συστήματα με την ιδέα ότι η εξέλιξη μπορεί να χρησιμοποιηθεί ως ένα εργαλείο για λύση μηχανολογικών προβλημάτων. Η ιδέα σε αυτά τα συστήματα στηρίζεται στην εξέλιξη του πληθυσμού του απαρτίζεται από πιθανές λύσεις για ένα δεδομένο πρόβλημα, χρησιμοποιώντας μηχανισμούς που εμπνεύστηκαν από τη φυσική γενετική παραλλαγή και τη φυσική επιλογή.

Οι Γενετικοί Αλγόριθμοι (Genetic Algorithms) είναι ένα παράδειγμα τέτοιου συστήματος και ανήκει σε μια κατηγορία συστημάτων επίλυσης προβλημάτων που είναι ευρύτερα γνωστή με τον όρο Εξελικτικοί Αλγόριθμοι (Evolutionary Algorithms).

Ο καθηγητής John Holland θεμελίωσε τους Γενετικούς Αλγορίθμους ως μεθόδους βελτιστοποίησης στις αρχές του 1970 με τους συνεργάτες του στο Πανεπιστήμιο του Michigan.

Ένας Γενετικός Αλγόριθμος για ένα συγκεκριμένο πρόβλημα πρέπει να έχει τα πέντε παρακάτω συστατικά, όπως συνοψίζει ο Michalewicz (Michalewicz, 1998) :

- μία γενετική αναπαράσταση για πιθανές λύσεις του προβλήματος,
- μία μέθοδο δημιουργίας ενός αρχικού πληθυσμού πιθανών λύσεων,
- μία αντικειμενική συνάρτηση, η οποία θα αναπαριστά το περιβάλλον και θα βαθμολογεί τις λύσεις ανάλογα με την καταλληλότητά τους,
- γενετικούς τελεστές που μεταβάλλουν την σύνθεση των παιδιών,
- τιμές για διάφορες παραμέτρους τους οποίους ο γενετικός αλγόριθμος χρησιμοποιεί (μέγεθος πληθυσμού, πιθανότητες εφαρμογής των γενετικών τελεστών, κλπ).

Οι πιθανές λύσεις του προβλήματος βρίσκονται στη γειτονιά μιας λύσης (neighborhood). Γειτονιά μιας λύσης χ μπορεί να οριστεί ως μία ομάδα λύσεων που μπορούν να βρεθούν χρησιμοποιώντας μια διαδικασία σ. Αυτή η διαδικασία δεν εφαρμόζεται στις λύσεις, δηλαδή στα προγράμματα, αλλά σε κατάλληλα διαμορφωμένες αναπαραστάσεις των προγραμμάτων. Αυτά τα συστήματα αναπαραστάσεων λοιπόν, πρέπει να περιέχουν κατάλληλες διαδικασίες κωδικοποίησης για να μετατρέπουν την αναπαράσταση σε ένα πρόγραμμα. Παρακάτω θα αναλυθούν τα πιο σημαντικά συστήματα αναπαραστάσεων που βρέθηκαν στην βιβλιογραφία.

Πριν εκτελεσθεί ο γενετικός αλγόριθμος, εφαρμόζουμε έναν μηχανισμό κατά τον οποίο μειώνουμε των χώρο που θα αναζητήσει λύσεις ο αλγόριθμος. Ο μηχανισμός αυτός ονομάζεται preprocessing. Μετά από αυτήν την διαδικασία ο αλγόριθμος παράγει τον αρχικό πληθυσμό (initial population), δηλαδή την πρώτη γενιά. Το σύνολο των ατόμων (individuals) που απαρτίζει τον πληθυσμό συμβολίζεται ως POP. Στην παρούσα εργασία θεωρούμε ότι ο POP είναι θετικός ζυγός αριθμός. Η επόμενη κίνηση είναι να υπολογίσει τη συνάρτηση καταλληλότητα (Fitness value) των ατόμων του αρχικού πληθυσμού. Εφόσον έχει υπολογιστεί, τα άτομα χωρίζονται σε ομάδες των δύο, δηλαδή ζευγαρώνουν. Σε κάθε ένα από τα ζεύγη των ατόμων (πατέρας-μητέρα), εφαρμόζουμε τη διαδικασία της διασταύρωσης για να παράξουμε δύο καινούργια άτομα, τα παιδιά (ιό και κόρη). Έπειτα εφαρμόζουμε την μετάλλαξη στα φαινότυπα των παιδιών. Αφού υπολογίσουμε και την fitness function κάθε παιδιού, προσθέτουμε τα παιδιά στον τρέχον πληθυσμό. Έτσι έχουμε πληθυσμό ίσο με 2*POP. Για να μειωθεί ο πληθυσμός σε αριθμό ίσο με POP πάλι, εφαρμόζουμε τη διαδικασία της επιλογής (selection). Έχουμε λοιπόν ένα πληθυσμό πλέον ίσο με POP, ο οποίος περνά στην επόμενη γενιά στην οποία και εφαρμόζουμε τη διαδικασία διασταύρωσης. Αυτή η διαδικασία επαναλαμβάνεται μέχρι ένα συγκεκριμένο αριθμό γενεών που συμβολίζεται ως GEN, ή εναλλακτικά έως ένα συγκεκριμένου υπολογιστικού χρόνου.

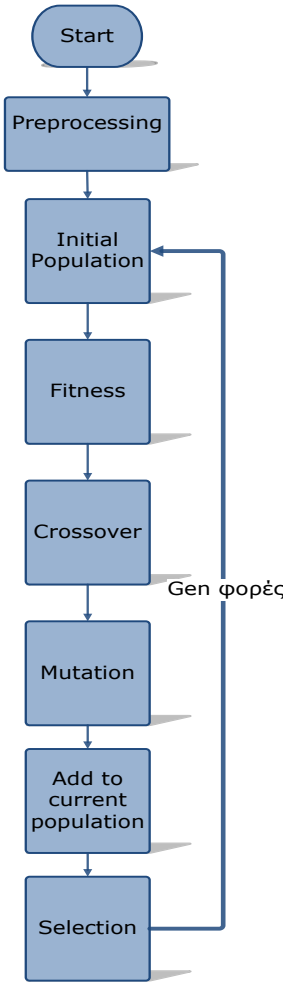
Συνοπτικά ο γενετικός αλγόριθμος μπορεί να παρουσιαστεί ως εξής:

Ο αριθμός POP συμβολίζει τον τρέχον πληθυσμό (ένα σύνολο από άτομα), και CHI είναι το σύνολο των απογόνων (παιδιών). G είναι ο τρέχον αριθμός που συμβολίζει την γενιά (G<GEN).

1. $Εφαρμόζεται\ η\ διαδικασία\ Προεπεξεργασίας\ (Preprocessing)$
2. $G = 1$
3. $Παράγεται\ ο\ Αρχικός\ Πληθυσμός\ (Initial\ Population)$
4. $Υπολογίζεται\ η\ Συνάρτηση\ Καταλληλότητας\ (Fitness\ Function)$
5. $\boldsymbol{ΓΙΑ}\ G < GEN\ \boldsymbol{ΚΑΙ}\ ο\ υπολογιστικός\ χρόνος\ δεν\ έχει\ ξεπεραστεί\ \boldsymbol{ΚΑΝΕ}$
6. $\boldsymbol{Αρχή}$
7. $G = G + 1$
8. $Παραγωγή\ CHI\ παιδιών\ μέσω\ της\ Διασταύρωσης$
9. $Μετάλλαξη\ στα\ CHI\ παιδιά$
10. $Υπολογισμός\ Συνάρτησης\ Καταλληλότητας\ στα\ CHI\ παιδιά$
11. $POP = POP \cup CHI$
12. $Επιλογή\ και\ μείωση\ του\ πληθυσμού\ σε\ POP$

13. Τέλος

Ο συνολικός αριθμός ατόμων που παράγεται είναι ίσος με $GEN * POP$ (εάν δεν υπάρχει περιορισμός για τον υπολογιστικό χρόνο) (Hartmann, 2001). Στην εικόνα 26 φαίνονται και διαγραμματικά τα βήματα του γενετικού αλγορίθμου.



Εικόνα **26** Βήματα Γενετικού Αλγορίθμου

6.2.1 Προ-επεξεργασία (Preprocessing)

Η διαδικασία αυτή είναι γνωστή από την έρευνα που έχει κάνει ο Sprecher (1997) για να μειώσει τον χρόνο υπολογισμού του αλγορίθμου Branch and Bound. Η διαδικασία έχει ως σκοπό να μειώσει τον χώρο στον οποίο αναζητά ο αλγόριθμος την λύση. Οι παρακάτω ορισμοί εξυπηρετούν στην κατανόηση αυτής της διαδικασίας.

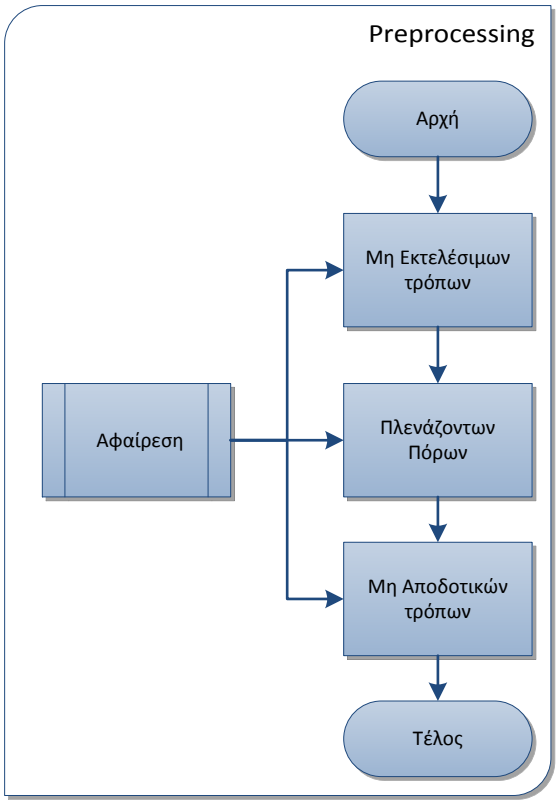
- **Μη εκτελέσιμος** (***non-executable***) ονομάζεται ο τρόπος ο οποίος παραβιάζει τους περιορισμούς διαθεσιμότητας των πόρων σε ένα πρόγραμμα.

- **Μη αποδοτικός** (***inefficient***) ονομάζεται όταν η διάρκειά του είναι μεγαλύτερη και συγχρόνως απαιτεί περισσότερους πόρους για την εκτέλεσή του από ότι οι άλλοι τρόποι εκτέλεσης για την ίδια δραστηριότητα.

- **Πλεονάζων (*redundant*)** ονομάζεται ένας μη ανανεώσιμος πόρος εάν η μέγιστη ζήτησή του, δηλαδή το άθροισμα των μέγιστων δυνατών συνδυασμών για όλες τις δραστηριότητες, δεν υπερβαίνει την μέγιστη διαθεσιμότητά του.

Στην αρχή του υπολογισμού κάθε προγράμματος, εφαρμόζεται η παραπάνω διαδικασία όπου οι μη εκτελέσιμοι και οι μη αποδοτικοί τρόποι εκτέλεσης, όπως και οι πλεονάζοντες πόροι διαγράφονται από τα δεδομένα του προβλήματος, χωρίς να επηρεάζουν την εύρεση της βέλτιστης λύσης.

Για να εφαρμοστεί σωστά η διαδικασία της προ-επεξεργασίας και να μην υπάρχουν απώλειες εφικτών ή βέλτιστων λύσεων πρέπει τα δεδομένα του προγράμματος να επεξεργαστούν με μία συγκεκριμένη σειρά. Η σειρά με την οποία διαγράφονται τα δεδομένα φαίνεται στην εικόνα 27.



ΕΙΚΟΝΑ 27 ΒΗΜΑΤΑ ΠΡΟΕΠΕΞΕΡΓΑΣΙΑΣ

Η λανθασμένη σειρά διαγραφής μπορεί να αποφέρει μείωση των εφικτών λύσεων. Για παράδειγμα, η αφαίρεση ένας πλεονάζων μη ανανεώσιμου πόρου, ως πρώτη ενέργεια, μπορεί να οδηγήσει σε μη αποδοτική συμπεριφορά ενός τρόπου εκτέλεσης. Ενώ η διαγραφή ενός τρόπου εκτέλεσης μπορεί να δημιουργήσει πλεονασμό για έναν μη ανανεώσιμο πόρο.

Το αποτέλεσμα αυτής της διαδικασίας είναι η μείωση των εφικτών αλλά και μη εφικτών λύσεων, δηλαδή μικρότερος χώρος για αναζήτηση λύσεων.

6.2.2 Κωδικοποίηση

Η κωδικοποίηση αφορά ένα σύνολο πιθανών λύσεων του προβλήματος. Η αναπαράσταση των λύσεων πρέπει να γίνει με ένα μαθηματικό, φορμαλιστικό τρόπο, ώστε να είναι δυνατή η επεξεργασία από τον υπολογιστή. Εξάλλου, κωδικοποίηση υπάρχει και στο φυσικό μοντέλο (χρωμοσώματα) και μάλιστα όλες οι αλλαγές που παρατηρούνται στους οργανισμούς γίνονται πάνω στα κωδικοποιημένα χαρακτηριστικά των χρωμοσωμάτων. Κύριος στόχος της κωδικοποίησης είναι να αναπαριστά με ικανοποιητικό τρόπο τα επιμέρους χαρακτηριστικά των λύσεων, ώστε να διευκολύνει τις επόμενες λειτουργίες του αλγορίθμου (κυρίως την επιλογή). Αποτέλεσμα της κωδικοποίησης πρέπει να είναι η ύπαρξη ομοιοτήτων ανάμεσα στα άτομα με σκοπό την κατάλληλη εκμετάλλευση τους, διότι οι ομοιότητες βοηθούν την κατεύθυνση του ψαξίματος.

Κάθε λύση του προβλήματος κωδικοποιείται και εμπεριέχεται στο χρωμόσωμα. Η διαδικασία της κωδικοποίησης, δηλαδή το πως η πληροφορία που περιέχει το χρωμόσωμα μεταφράζεται σε λύση του προβλήματος, αποτελεί βασικό πρόβλημα και τίθεται κάθε φορά που ξεκινάμε να λύσουμε ένα πρόβλημα με γενετικούς αλγορίθμους. Η κωδικοποίηση εξαρτάται κυρίως από τον τύπο του προβλήματος που έχουμε να αντιμετωπίσουμε. Υπάρχουν πολλοί τρόποι κωδικοποίησης όπως είναι:

1. Η δυαδική κωδικοποίηση (binary encoding)

Όλο το χρωμόσωμα είναι ένας ακέραιος αριθμός και τα γονίδια του είναι δυαδικά στοιχεία (0 η 1).

2. Η κωδικοποίηση βάση πραγματικών τιμών (real-number encoding)

Η κωδικοποίηση βάση πραγματικών τιμών είναι η καλύτερη για προβλήματα που περιέχουν συναρτήσεις βελτιστοποίησης. Η κωδικοποίηση αυτή έχει αποδειχθεί ότι λειτουργεί καλύτερα από ότι οι όλες οι άλλες στα προβλήματα που περιέχουν

περιορισμούς που πρέπει να βελτιστοποιηθούν. (McCormick Jr et al., 1972, Michalewicz, 1998, Gen and Cheng, 2000). Εφόσον η τοπολογική δομή του γενοτύπου με κωδικοποίηση βάση πραγματικών τιμών είναι πανομοιότυπη με αυτή του φαινοτύπου, είναι εύκολο να χρησιμοποιηθούν οι γενετικοί τελεστές των συμβατικών μεθόδων.

3. Η κωδικοποίηση μετάθεσης (permutation encoding)

Κάθε χρωμόσωμα σε αυτήν την κωδικοποίηση αποτελείται από μια σειρά αριθμών, η οποία αναπαριστά μια αλληλουχία σε μια διαδικασία. Αυτή η κωδικοποίηση είναι χρήσιμη μόνο σε προβλήματα που χρειάζεται να τοποθετήσουμε σε μια σειρά – αλληλουχία κάποιες εργασίες-δραστηριότητες. Παράδειγμα προβλήματος που χρησιμοποιείται αυτή η κωδικοποίηση είναι το πρόβλημα του περιπλανώμενου πωλητή. Για πολύπλοκα όμως προβλήματα στον πραγματικό κόσμο χρησιμοποιούνται κατάλληλες πιο πολύπλοκες δομές για να κωδικοποιηθεί η φύση του προβλήματος.

Συστήματα Αναπαράστασης (Representation Schemes)

Η κωδικοποίηση είναι ο τρόπος που μεταφράζεται η λύση που περιέχεται στο χρωμόσωμα. Σύστημα αναπαράστασης είναι ο τρόπος με τον οποίο χρησιμοποιούμε την κωδικοποίηση μέσα στο χρωμόσωμα**.**

- **Σύστημα Αναπαράστασης με τιμές προτεραιότητας – Αναπαράσταση τυχαίου κλειδιού (Priority (value) list representation – Random key representation)**

Σε αυτήν την αναπαράσταση κάθε χρωμόσωμα αναπαριστάται από μία ακολουθία από τιμές προτεραιότητας (*priority values*). Για παράδειγμα, το χρωμόσωμα ή άτομο (*chromosome – individual*):

$$I = (pv_1^I, \dots, pv_j^I).$$

Για κάθε τιμή των δραστηριοτήτων $j = 1, \dots, J$, έχουμε είτε $pv_j^I \in [0,1]$είτε $pv_j^I \in [1, J]$. Αυτή η αναπαράσταση χρησιμοποιήθηκε από τους Lee και Kim (LEE et al., 1996) βάσει της παράλληλης μεθόδου παραγωγής χρονοπρογράμματος. Για τον αρχικό πληθυσμό αυτές οι τιμές επιλέγονται τυχαία. Στην παρούσα εργασία χρησιμοποιήθηκε η σειριακή μέθοδος παραγωγής χρονοπρογράμματος, η οποία σύμφωνα με τον Kolisch (Kolisch, 1996) ανταποκρίνεται καλύτερα σε προβλήματα με

πολλές δραστηριότητες, βάση αυτής της αναπαράστασης. Αυτό συμβαίνει γιατί η παράλληλη μέθοδος μπορεί να αποκλείσει την βέλτιστη λύση ψάχνοντας σε λάθος χώρο του προβλήματος, ενώ με την σειριακή μέθοδο πάντα θα βρίσκεται η βέλτιστη λύση.

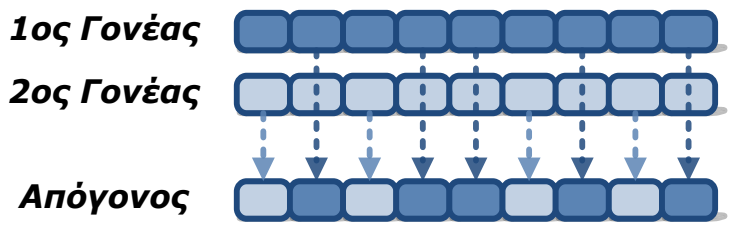
6.2.3 Διασταύρωση (Crossover)

Η διασταύρωση είναι η κύρια λειτουργία αναπαραγωγής. Διασταυρώνει τα χρωμοσώματα των δύο γονέων προκειμένου να παράγει δύο νέα άτομα, τα οποία ονομάζονται απόγονοι (γιος και κόρη, offsprings) και έχουν ουσιαστικά κληρονομήσει χαρακτηριστικά και από τους δύο γονείς.

Αφού έχουμε αποφασίσει ποιο είδος κωδικοποίησης θα χρησιμοποιήσουμε, μπορούμε να προχωρήσουμε στη διαδικασία διασταύρωσης. Η διασταύρωση εφαρμόζεται σε επιλεγμένα γονίδια από τα χρωμοσώματα-γονείς και δημιουργεί νέο απόγονο. Ο απλούστερος τρόπος για να γίνει η διασταύρωση είναι να επιλεγεί τυχαία πάνω στα χρωμοσώματα ένα σημείο διασταύρωσης και να αντιγραφούν όλα τα γονίδια πριν από το συγκεκριμένο σημείο από τον πρώτο γονέα και όλα τα γονίδια μετά το σημείο διασταύρωσης από τον δεύτερο γονέα στον απόγονο. Υπάρχουν και άλλοι τρόποι να εφαρμόσουμε διασταύρωση όπως το να επιλέξουμε περισσότερα του ενός σημεία διασταύρωσης. Η διασταύρωση μπορεί γενικώς να γίνει αρκετά πολύπλοκη διαδικασία και αυτό εξαρτάται κυρίως από την κωδικοποίηση των χρωμοσωμάτων. Ο σωστός τρόπος διασταύρωσης αν εφαρμοστεί στο σωστό πρόβλημα μπορεί να βελτιώσει αισθητά την απόδοση του γενετικού αλγορίθμου.

Διασταύρωση βάσει της θέσης (Position-based crossover)

Η διασταύρωση βάση της θέσης αναπτύχθηκε από τον Syswerda (Davis, 1991). Το χρωμόσωμα του ενός απόγονου γεμίζει με γονίδια από τον ένα γονέα τυχαία και με έναν έλεγχο από τα αριστερά στα δεξιά γεμίζει τις άδειες θέσεις από τον άλλο γονέα.



ΕΙΚΟΝΑ 28 POSITION-BASED CROSSOVER

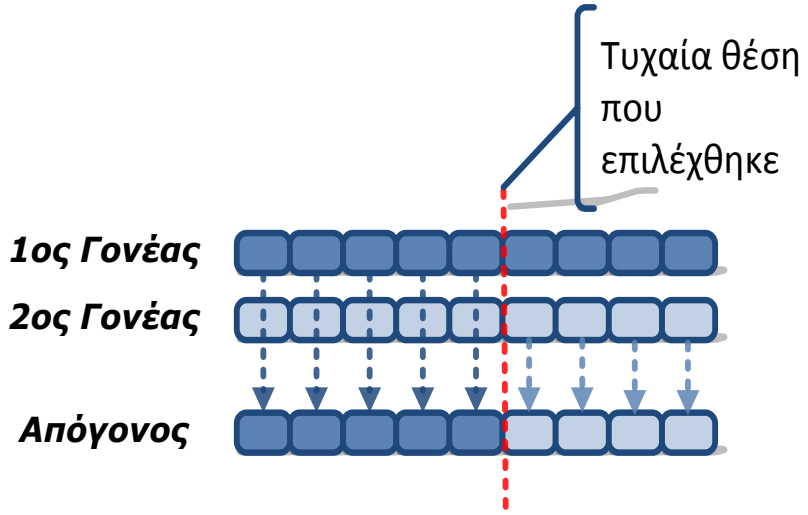
- **Διασταύρωση ενός σημείου (One Point Crossover)**

Η διασταύρωση ενός σημείου είναι η πλέον απλούστερη και η πιο συνηθισμένη μορφή διασταύρωσης. Χρησιμοποιείται κυρίως σε ακέραια χρωμοσώματα, αλλά είναι τόσο γενική και απλή στην υλοποίηση που μπορεί να χρησιμοποιηθεί και σε χρωμοσώματα διαφορετικής μορφής. Η διαδικασία που ακολουθείται είναι η επόμενη:

• Επιλέγεται με τυχαίο τρόπο μια θέση στα χρωμοσώματα που θα διασταυρωθούν

• Ανταλλάσσεται το δεξιό τμήμα του πρώτου χρωμοσώματος με το αριστερό τμήμα του δεύτερου και αντίστροφα, πριν και μετά το σημείο που επιλέχτηκε

Πρέπει να επισημανθεί ωστόσο, πως δεν είναι υποχρεωτικό τα δύο χρωμοσώματα να είναι ίσα σε μέγεθος, προκειμένου να εφαρμοστεί η διαδικασία της διασταύρωσης ενός σημείου. Παρά την απλή της μορφή η διασταύρωση αυτή έχει τα ίδια αποτελέσματα με πιο περίπλοκες διασταυρώσεις.

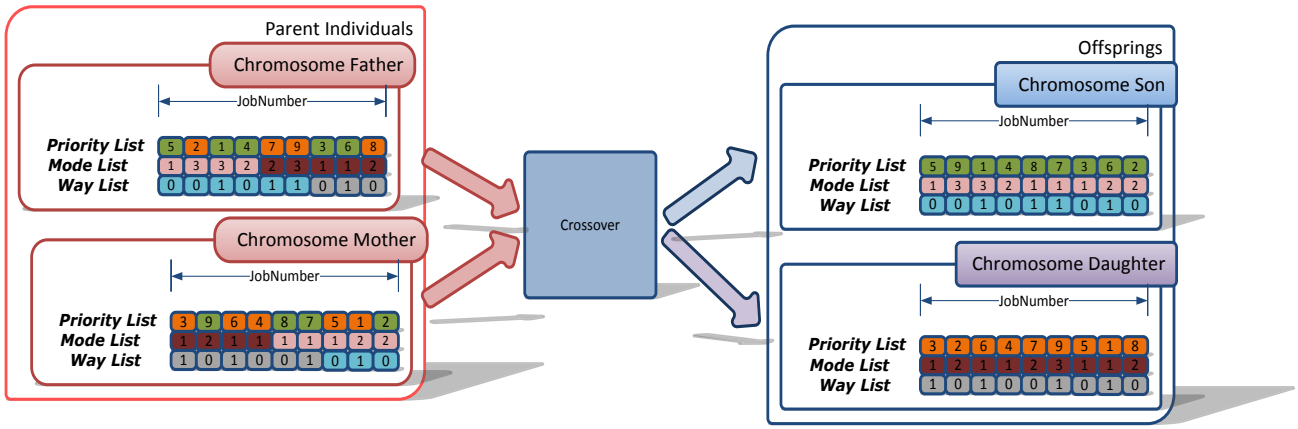


ΕΙΚΟΝΑ 29 ONE POINT CROSSOVER

Στον κώδικά μας χρησιμοποιήσαμε για την πρώτη λίστα, των τιμών προτεραιότητας, την διασταύρωση βάση της θέσης των γονιδίων. Έτσι για κάθε γονίδιο παράγεται τυχαία ο αριθμός 0 ή ο αριθμός 1. Εάν παραχθεί 0 τότε επιλέγεται το γονίδιο από τη λίστα του πατέρα να γεμίσει το γονίδιο του γιου. Αντίστοιχα εάν παραχθεί ο αριθμός 1 επιλέγεται το γονίδιο της μητέρας. Τα γονίδια για τη λίστα της κόρης είναι τα εναπομείναντα από τον πατέρα και της μητέρας μετά από το γέμισμα της λίστας του γιου.

Για τη δεύτερη και την τρίτη λίστα χρησιμοποιήσαμε τη διασταύρωση ενός σημείου (One Point Crossover). Ένας τυχαίος αριθμός παράγεται

(Random<JobNumber) και βάση αυτού γεμίζεται η λίστα του γιου. Δηλαδή, μέχρι αυτόν τον αριθμό τα γονίδια παίρνονται από τον πατέρα και τα υπόλοιπα από τη μητέρα. Με αντίστροφη σειρά γεμίζει η λίστα για την κόρη. Η διαδικασία για τη διασταύρωση και των τριών λιστών φαίνεται στην εικόνα 30.



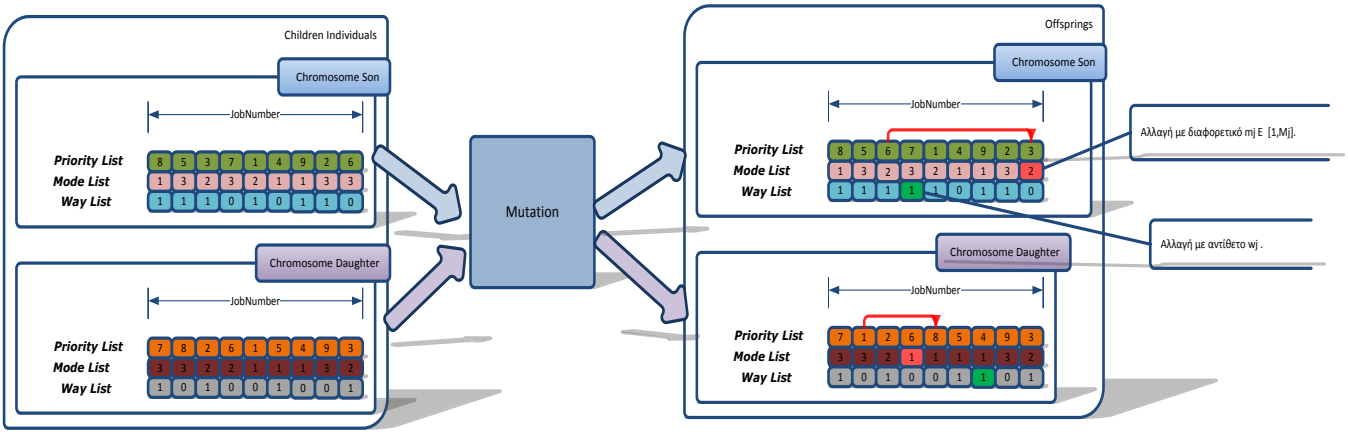
ΕΙΚΟΝΑ 30 ΔΙΑΣΤΑΥΡΩΣΗ

6.2.4 Μετάλλαξη (Mutation)

Μετάλλαξη είναι η διαδικασία κατά την οποία τα γονίδια ενός ατόμου μεταλλάσσονται, προκειμένου να προκύψει ένα νέο άτομο που αντικαθιστά το υπάρχον. Κάθε νέος απόγονος μεταβάλλει τα γονίδιά του με μία μικρή πιθανότητα. Αν για παράδειγμα έχουμε δυαδική κωδικοποίηση αυτό μπορεί να γίνει μεταλλάσσοντας τα γονίδια που περιέχουν το 0 σε 1 ή αντίστροφα. Η μετάλλαξη αποτελεί βασική λειτουργία των γενετικών αλγορίθμων αφού στόχος της είναι να επαναφέρει χαρακτηριστικά του πληθυσμού τα οποία χάθηκαν κατά τη προηγούμενη λειτουργία, δηλαδή την διασταύρωση.

Για την πρώτη λίστα, επιλέγονται δύο διαφορετικές δραστηριότητες τυχαία και εναλλάσσουν μεταξύ τους τις τιμές προτεραιότητάς τους. Η καινούργια λίστα που προκύπτει για το γιο, για παράδειγμα, ανταποκρίνεται στις απαιτήσεις προτεραιότητας, εφόσον δεν αλλάζει ο τρόπος με τον οποίο είναι συνδεδεμένες οι δραστηριότητες. Δηλαδή και μετά από αυτήν την αλλαγή τα προγράμματα που προκύπτουν είναι εφικτά (υπακούουν στους περιορισμούς προτεραιότητας). Για την δεύτερη λίστα και την τρίτη λίστα παρόμοια με την πρώτα επιλέγεται ένα γονίδιο για να αλλάξει. Στην δεύτερη επιλέγεται διαφορετικός τρόπος εκτέλεσης από τις δυνατές επιλογές. Προσοχή χρειάζεται στο ότι μετά την προ-επεξεργασία κάποιοι τρόποι εκτέλεσης έχουν διαγραφεί, οπότε μπορεί να μην υπάρχει εναλλακτικός συνδυασμός

και για αυτό το λόγο το γονίδιο παραμένει ίδιο. Στην τρίτη λίστα ένα γονίδιο με τιμή 0 (normal duration) παίρνει την τιμή 1 (crashing duration) και αντίστροφα.



ΕΙΚΟΝΑ 31 ΜΕΤΑΛΛΑΞΗ

6.2.5 Πιθανότητα μετάλλαξης

Πιθανότητα μετάλλαξης: πόσο συχνά κομμάτια (γονίδια) χρωμοσωμάτων μεταλλάσσονται. Αν δεν υπάρξει μετάλλαξη, οι απόγονοι παράγονται ακριβώς μετά τη διασταύρωση (ή κατευθείαν αντιγράφονται, αν δεν υπάρξει ούτε διασταύρωση) χωρίς καμία αλλαγή. Αν η πιθανότητα μετάλλαξης είναι 100%, όλος ο πληθυσμός, δηλαδή όλα τα άτομα, μεταλλάσσονται ενώ, αν είναι 0%, κανένα άτομο δεν μεταλλάσσεται και όλα περνάνε στην επόμενη γενιά όπως ήταν. Η μετάλλαξη δε θα πρέπει να συμβαίνει πολύ συχνά, διότι τότε ο γενετικός αλγόριθμος θα μετατραπεί σε τυχαία αναζήτηση, χάνοντας τα χαρακτηριστικά και τις ιδιότητές του.

6.2.6 Μέγεθος πληθυσμού (Population Size)

Πόσα χρωμοσώματα αποτελούν τον πληθυσμό (σε μια γενιά). Αν υπάρχουν πολύ λίγα χρωμοσώματα, ο γενετικός αλγόριθμος έχει λίγες δυνατότητες να εφαρμόσει διασταύρωση και μόνο ένα μικρό κομμάτι του πεδίου αναζήτησης εξερευνείται. Από την άλλη πλευρά, αν υπάρχουν πάρα πολλά χρωμοσώματα, ο γενετικός αλγόριθμος λειτουργεί λιγότερο αποδοτικά. Έρευνες δείχνουν ότι μετά από κάποιο όριο (το οποίο εξαρτάται κυρίως από την κωδικοποίηση και το είδος του προβλήματος) δεν είναι σκόπιμο να χρησιμοποιήσουμε πολύ μεγάλους πληθυσμούς γιατί ο αλγόριθμος λύνει το πρόβλημα σχετικά αργά.

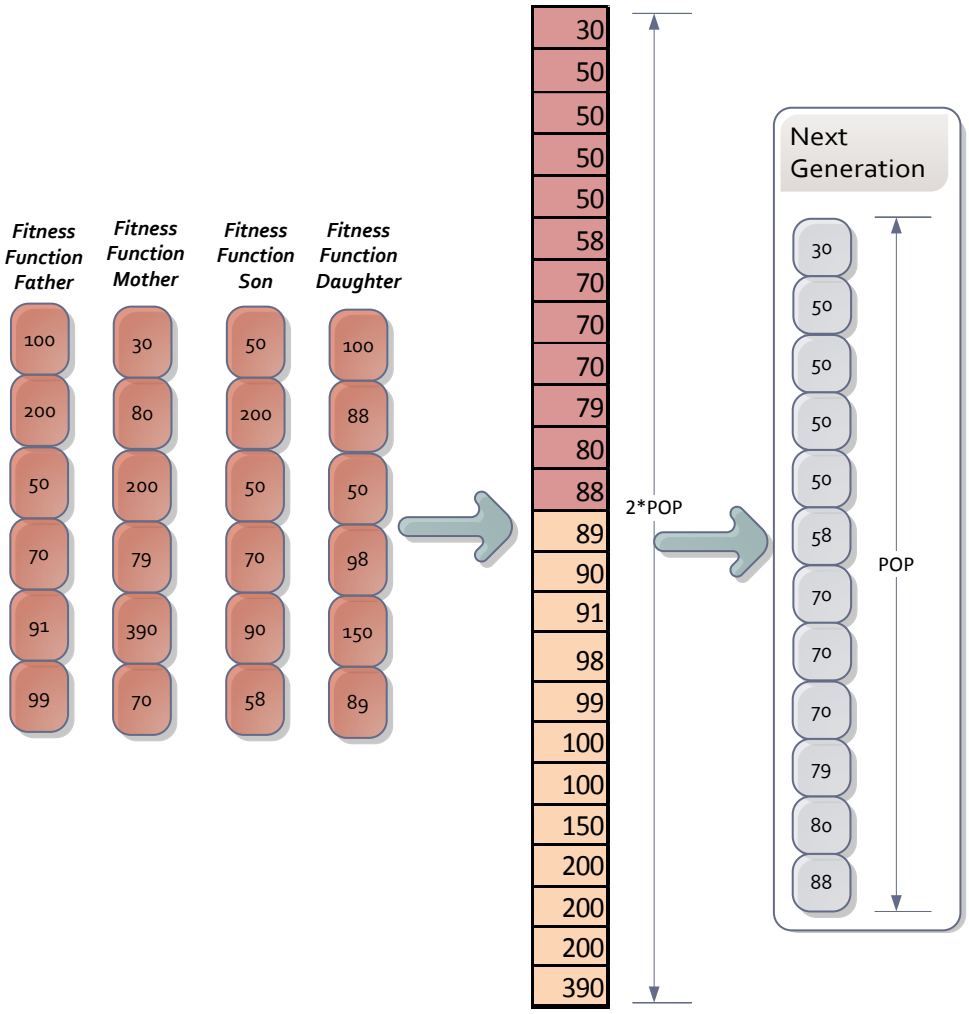
6.2.7 Επιλογή (Selection)

Επιλογή είναι η λειτουργία κατά την οποία τα άτομα που δημιουργήθηκαν από τη διασταύρωση και τα άτομα του αρχικού πληθυσμού επιλέγονται για να περάσουν

στην επόμενη γενιά. Αυτό γίνεται βάση της προσαρμογής τους, δηλαδή της τιμής της αντικειμενικής συνάρτησης του καθενός. Τα άτομα (Individuals) από κάθε ζεύγος διασταυρώνονται προκειμένου να δημιουργηθούν άτομα της επόμενης γενιάς. Η επιλογή στηρίζεται σε μεγάλο βαθμό στην τυχαιότητα, δεδομένου ότι τα ζεύγη των γονέων πρέπει να είναι από τα καλά του πληθυσμού. Η κατάλληλη επιλογή αποτελεί σημαντική λειτουργία των ΓΑ, η οποία διασφαλίζει την κυριαρχία του ισχυρότερου ατόμου. Διάφορες μέθοδοι έχουν προταθεί για την επιλογή. Οι βασικότερες (Rokou and Kirytopoulos, 2012) είναι:

- Tournament-Based Επιλογή
- Πιθανότητα θανάτου (Probability of death)
- Επιλογή Ταξινόμησης (Rank Selection)
- Επιλογή Ρουλέτας (Roulette wheel selection method)

Η σειρά που παρουσιάζονται είναι από την καλύτερη μέθοδο στη χειρότερη όσο αφορά την μέση απόκλιση από την βέλτιστη λύση (Rokou and Kirytopoulos, 2012). Στην εργασία μας χρησιμοποιήσαμε τη μέθοδο ranking ή elitism όπου ταξινομούνται όλες οι λύσεις του προβλήματος, δηλαδή οι τιμές των συναρτήσεων καταλληλότητας για κάθε άτομο (individual) και επιλέγονται οι καλύτερες. Δεδομένου του ότι έχουμε πρόβλημα ελαχιστοποίησης του κόστους, επιλέγονται τα άτομα των οποίων οι τιμές στην συνάρτηση καταλληλότητας είναι οι μικρότερες. Για παράδειγμα στην εικόνα 32 φαίνεται η διαδικασία επιλογής, όπου ο αρχικός πληθυσμός είναι ίσος με 12. Μετά τη διασταύρωση ο τρέχων πληθυσμός γίνεται $2 * 12 = 24$. Οι τιμές της συνάρτησης καταλληλότητας για τον αρχικό πληθυσμό (Fitness Function Father-Mother) και οι τιμές της έπειτα από τη διασταύρωση και τη μετάλλαξη για τους απογόνους (Fitness Function Son-Daughter) φαίνονται στην εικόνα 32. Όλες οι τιμές μεταφέρονται σε ένα καινούργιο πίνακα με μέγεθος $2 * POP$ και ταξινομούνται. Έπειτα γίνεται η επιλογή των καλύτερων τιμών και τα άτομα με τις καλύτερες λύσεις περνάνε στην επόμενη γενιά. Ο τρέχων πληθυσμός γίνεται πάλι ίσος με POP.



ΕΙΚΟΝΑ 32 ΔΙΑΔΙΚΑΣΙΑ ΕΠΙΛΟΓΗΣ

7 Ανάλυση του Συστήματος

Ο τρόπος με τον οποίο ο κώδικας διαβάζει τα αρχεία φαίνεται στο παρακάτω πίνακα. Για παράδειγμα ένα αρχείο της PSPlib, το j102_2 χρησιμοποιήθηκε και βρίσκεται στο παράρτημα. Τα δεδομένα από το αρχείο j102_2 τροποποιούνται σύμφωνα με τον παρακάτω πίνακα. Το πρόγραμμά διαβάζει σειρά-σειρά το τροποποιημένο αρχείο και τοποθετεί τα δεδομένα σε πίνακες με τρόπο που αναφέρεται παρακάτω.

ΠΙΝΑΚΑΣ 6 ΔΕΔΟΜΕΝΑ ΠΡΟΒΛΗΜΑΤΟΣ J102_2

JobNumber	**ResourcesType**										
12	4										
ResAvailabilityK1	ResAvailabilityK2	ResAvailabilityK3	ResAvailabilityK4								
9	4	29	40								
JobId	NumberOfModes	ModeNumber	Duration	ResNeedK1	ResNeedK2	ResNeededK3	ResNeedK4	NumrOfSuccessors	SucId1	SucId2	SucId3
1	1	1	0	0	0	0	0	3	2	3	4
2	3	1	3	6	0	9	0	2	5	6	
		2	9	5	0	0	8				
		3	10	0	6	0	6				
3	3	1	1	0	4	0	8	2	10	11	
		2	1	7	0	0	8				
		3	5	0	4	0	5				
4	3	1	3	10	0	0	7	1	9		
		2	5	7	0	2	0				
		3	8	6	0	0	7				
5	3	1	4	0	9	8	0	2	7	8	
		2	6	2	0	0	7				
		3	10	0	5	0	5				
6	3	1	2	2	0	8	0	2	10	11	
		2	4	0	8	5	0				
		3	6	2	0	0	1				
7	3	1	3	5	0	10	0	2	9	10	
		2	6	0	7	10	0				
		3	8	5	0	0	10				
8	3	1	4	6	0	0	1	1	9		
		2	10	3	0	10	0				
		3	10	4	0	0	1				
9	3	1	2	2	0	6	0	1	12		
		2	7	1	0	0	8				
		3	10	1	0	0	7				
10	3	1	1	4	0	4	0	1	12		
		2	1	0	2	0	8				

		3	9	4	0	0	5			
11	3	1	6	0	2	0	10	1	12	
		2	9	0	1	0	9			
		3	10	0	1	0	7			
12	1	1	0	0	0	0	0	0	0	0

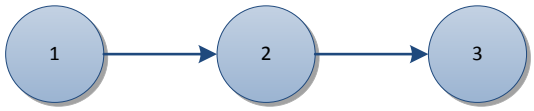
Το πρόγραμμα, λοιπόν, ξεκινά να διαβάζει το αρχείο. Ο πρώτος αριθμός αντιπροσωπεύει τον αριθμό των δραστηριοτήτων και βάσει αυτού δημιουργείτε ο πίνακας JobNumber με μέγεθος όσο αυτός ο αριθμός. Ο επόμενος αριθμός που διαβάζει (ResourcesType) αντιστοιχεί στον αριθμό των τύπων πόρων που χρειάζεται το έργο. Κάθε ResourcesType έχει διαφορετική διαθεσιμότητα ίση με ResourceAvailabilityK, για κάθε ResourcesType. Δημιουργείτε ο πίνακας ResourceAvailability με μέγεθος όσο ο αριθμός ResourcesType. Πρέπει να αναφερθεί ότι στο πρόβλημά του DTCTP, μόνο ένας μη ανανεώσιμος πόρος υπάρχει και αυτός θεωρείτε ότι είναι το κόστος. Οπότε οι μη ανανεώσιμοι πόροι που υπάρχουν στα προβλήματα της PSPlib δεν λαμβάνονται υπόψην. Έπειτα δημιουργείται ο πίνακας NumberOfModes που περιέχει τον αριθμό τρόπων εκτέλεσης (mode) για κάθε δραστηριότητα και ο πίνακας NumberOfSuccesors με τον αριθμό των επιτυχόντων δραστηριοτήτων για κάθε δραστηριότητα. Οι δισδιάστατοι πίνακες δημιουργούνται στη συνέχεια είναι:

4. SuccessorsId και περιέχει για κάθε δραστηριότητα τους αριθμούς (JobId) με τις δραστηριότητες που αποτελούν succesors τους.
5. ModeNumber, οι τρόποι εκτέλεσης που αντιστοιχούν σε κάθε δραστηριότητα
6. Duration, την διάρκεια εκτέλεσης για κάθε δραστηριότητα με κάθε τρόπο εκτέλεσης.

Και ο τρισδιάστατος πίνακας:

7. ResourcesNeed, την ποσότητα των πόρων που χρειάζεται κάθε τρόπος εκτέλεσης για κάθε δραστηριότητα και για κάθε τύπο πόρου (ResourcesType).

Στη συνέχεια του προγράμματος δημιουργούνται οι πίνακες NumberPredecessor και PredecessorId, οι οποίοι γεμίζουν με δεδομένα από τους πίνακες NumberOfSuccesors και SuccessorsId. Αυτό γίνεται για κάθε δραστηριότητα όπως στο παρακάτω παράδειγμα.



ΕΙΚΟΝΑ 33

Η δραστηριότητα 1 έχει successor την 2. Η δραστηριότητα 2 έχει successor την 3. Ενώ η δραστηριότητα 2 έχει predecessor την 1 και η δραστηριότητα 3 έχει predecessor την 2 (εικόνα 33).

Στη συνέχεια δημιουργείται ο πίνακας CrashingDuration. Για τη δημιουργία του χρησιμοποιείται ένας συντελεστής ξ ο οποίος πολλαπλασιάζεται με κάθε στοιχείο από τον πίνακα Duration. Προφανώς $\xi < 1$. Στην παρούσα εργασία θεωρήσαμε $\xi = 0{,}6$. Για να είναι όλες οι διάρκειες ακέραιες στρογγυλοποιήσαμε, τις συμπιεσμένες διάρκειες προς τα πάνω όπου αυτό χρειάστηκε. Το πρόγραμμα ζητάει από το χρήστη μια σειρά δεδομένων που έχουν σχέση με τη λειτουργία του γενετικού αλγορίθμου. Τα δεδομένα που χρειάζεται είναι:

8. Οι γενιές (Generation),
9. Ο αρχικός πληθυσμός (Initial population)
10. Τον μέγιστο χρόνο ολοκλήρωσης του έργου Tmax
11. Την πιθανότητα μετάλλαξης Pmutation
12. Τον τρόπο που γεμίζει ο πίνακας με τις τιμές προτεραιότητας (Priotity Values)
13. Τον τρόπο που γεμίζει ο πίνακας SelectedModes
14. Και τον τρόπο που γεμίζει ο πίνακας SelectedWays.

Οι πληροφορίες 8, 9 και 11, αφορούν τις διαδικασίες του γενετικού αλγορίθμου. Δεδομένης μιας πιθανότητας μετάλλαξης ο αλγόριθμος παράγει σε κάθε γενιά (generation) για κάθε άτομο (Individual), από τους απογόνους (γιο και κόρη) έναν τυχαίο αριθμό $\in [0{,}1]$. Εάν ο αριθμός αυτός είναι μικρότερος της πιθανότητας μετάλλαξης τότε σε αυτό άτομο εφαρμόζεται μετάλλαξη, εάν όχι αυτό το άτομο περνά ανέπαφο στην διαδικασία της επιλογής.

Οι τρεις τελευταίες πληροφορίες είναι απαραίτητες για την δημιουργία του χρωμοσώματος (Individual). Ο αριθμός των χρωμοσωμάτων για κάθε γενιά, όπως

έχει προαναφερθεί, είναι ίσος με τον αριθμό Initial Population. Και για τις τρεις, ο χρήστης μπορεί να επιλέξει να γεμίζουν οι αντίστοιχοι πίνακες τυχαία ή να δώσει αυτός τιμές για κάθε γονίδιο του χρωμοσώματος. Για παράδειγμα, για την πρώτη λίστα του χρωμοσώματος ο χρήστης μπορεί να διαλέξει τυχαίο γέμισμα ή να δώσει τιμές χειροκίνητα για κάθε δραστηριότητα. Εάν διαλέξει την δεύτερη επιλογή πρέπει να εισαχθούν τιμές διαφορετικές για κάθε δραστηριότητα. Με αυτόν τον τρόπο, μια δραστηριότητα που θεωρεί ο χρήστης σημαντική προγραμματίζεται πρώτη το συντομότερο δυνατό. Για την δεύτερη, εάν επιλεχθεί χειροκίνητη επιλογή, ο χρήστης πρέπει να εισάγει για κάθε δραστηριότητα τον τρόπο από τον πίνακα ModeNumber που θέλει να εκτελεσθεί κάθε δραστηριότητα. Στη τρίτη, η επιλογή είναι ανάμεσα σε 0, δηλαδή κανονική διάρκεια, ή 1, συμπιεσμένη.

Για παράδειγμα και για σύγκριση των αποτελεσμάτων χρησιμοποιήθηκε το παράδειγμα n311_10 της PSPlib. Η διαδικασία δείχνει πως γίνεται η σειριακή μέθοδος χρονοπρογραμματισμού και πως δέχεται ο αλγόριθμος τα δεδομένα από το χρωμόσωμα. Τα δεδομένα που χρησιμοποιήθηκαν και τα αποτελέσματα βρίσκονται στο Παράρτημα Γ. Το παράδειγμα έχει τα εξής χαρακτηριστικά:

- Δεν γίνεται προεπεξεργασία έτσι ώστε να είναι εμφανείς όλοι οι τρόποι εκτέλεσης που δίνονται από την Psplib.
- Ο πληθυσμός ισούται με 6.
- Επιλέχθηκαν χειροκίνητα τα γονίδια που αποτελούν και τις 6 λίστες έτσι ώστε να μπορούμε να τα επαληθεύσουμε με τα αποτελέσματα του κώδικα.
- Κάθε φύλλο excel αντιστοιχεί σε ένα πρόγραμμα, δηλαδή σε μία λύση του προβλήματος.

Για την αποτελεσματικότητα του αλγορίθμου στο πρόβλημα του MRCPSP, συγκρίθηκαν τα αποτελέσματα με τις βέλτιστες λύσεις της PSPLIB. Ο πίνακας 7 έχει τις παραμέτρους που χρησιμοποιήθηκαν για την εύρεση των αποτελεσμάτων.

ΠΙΝΑΚΑΣ 7 ΠΑΡΑΜΕΤΡΟΙ ΓΕΝΕΤΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ

POP	GEN	Pmut	REPETITIONS
30	40	0,05	100

Στον πίνακα 8 φαίνονται ένα μέρος των αποτελεσμάτων του αλγορίθμου μας και τα μεγέθη με τα οποία συγκρίθηκαν. Αναλυτικά στην πρώτη στήλη φαίνεται το όνομα του αρχείου. Οι δύο επόμενες στήλες περιέχουν την ελάχιστη και τη μέγιστη διάρκεια από τις επαναλήψεις που έγιναν. Η τέταρτη στήλη περιέχει το μέσο όρο των αποτελεσμάτων, ενώ στην πέμπτη στήλη βρίσκεται η βέλτιστη διάρκεια όπως αυτή δίνεται από τη PSPLIB. Η έκτη στήλη περιέχει τη μέση σχετική μέση απόκλιση από τη βέλτιστη λύση (standar relative deviation from optimum). Η έβδομη στήλη περιέχει τη συχνότητα με την οποία ο αλγόριθμος βρίσκει τη βέλτιστη λύση. Οι τρεις επόμενες στήλες περιέχουν πληροφορίες σχετικά με τη δυσκολία του κάθε προγράμματος. Τα μεγέθη των δύο πρώτων στηλών περιγράφουν τη σπανιότητα της διαθεσιμότητας των πόρων, ως τη διαθεσιμότητα του κάθε τύπου πόρου όταν οι δραστηριότητες εκτελούνται στους νωρίτερους χρόνους έναρξης μείον την ελάχιστη απαιτούμενη ποσότητα, προς τη μέγιστη απαιτούμενη ποσότητα μείον την ελάχιστη απαιτούμενη ποσότητα. Στην τελευταία στήλη το μέγεθος resource factor (RF) σχετίζεται με τον αριθμό των πόρων που απαιτούνται για κάθε δραστηριότητα και υπολογίζεται ως ο μέσος όρος των διαφορετικών τύπων πόρων όπου κάθε δραστηριότητα, εκτός των πλασματικών, έχει μη μηδενική τιμή.

$$RF = \frac{1}{nk}\sum_{i=1}^{n}\sum_{k=1}^{k}\left\{\begin{matrix}1, εάν\ η\ ποσότητα\ που\ απαιτείται\ είναι\ μη\ μηδενική \\ 0, αλλιώς\end{matrix}\right\}$$

ΠΙΝΑΚΑΣ 8 ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΛΓΟΡΙΘΜΟΥ MRCPSP

FileName	Min Dur	Max Dur	Average	Optimal	Average. Relative. Dev.	Frequency Of Optimum	Jobs Number	Res Strength 1	Res Strength 2	Resource Factor
j102_2	20	23	20,78	20	4%	46%	12	0,2	0,5	0,75
j102_4	18	24	18,82	18	5%	56%	12	0,2	0,5	0,75
j102_5	17	17	17	16	6%	0%	12	0,2	0,5	0,75
j102_6	16	24	17,3	16	8%	38%	12	0,2	0,5	0,75

Τα αποτελέσματα από τα υπόλοιπα προβλήματα βρίσκονται στο Παράρτημα Β.

8 Πρακτική Εφαρμογή Αλγορίθμου

8.1 Εισαγωγή

Το πρόγραμμα που δημιουργήθηκε για την έρευνα και λύση του προβλήματος MRC-DTCTP, χρησιμοποιήθηκε, έπειτα από αλλαγές για την προσαρμογή σε προβλήματα πραγματικού περιβάλλοντος, για την επίλυση ενός προβλήματος κτηματογράφησης.

Ο χρονοπρογραμματισμός έργων χρησιμοποιώντας γενετικούς αλγορίθμους σε προγραμματιστικό περιβάλλον βελτιώνει το αποτέλεσμα, δηλαδή δημιουργείται αποδοτικότερο πρόγραμμα έργου, βελτιώνει σημαντικά τον χρόνο και μειώνει τον κόπο, όπου γίνεται ο επαναπροσδιορισμός των παραμέτρων και των διαφόρων μεταβλητών του.

Το συγκεκριμένο έργο αποτελείται από 243 δραστηριότητες. Η συγκεκριμένη παράγραφος της εργασίας μας ασχολείται με τη δημιουργία χρονοπρογράμματος για αυτό το υπαρκτό έργο. Το αρχικό πρόβλημα μπορεί να θεωρηθεί ως πρόβλημα RCPSP, εφόσον αποτελείται από δραστηριότητες που πρέπει να προγραμματιστούν, με δεδομένη σειρά προτεραιότητας και με περιορισμένους πόρους. Στόχος είναι η ελαχιστοποίηση της συνολικής διάρκειας του έργου. Πιο συγκεκριμένα συνολικά υπάρχουν στο πρόβλημα 9 τύποι πόρων με διαφορετικές διαθεσιμότητες για τον καθένα. Όλοι οι πόροι είναι ανανεώσιμοι.

ΠΙΝΑΚΑΣ 9 ΤΥΠΟΙ ΠΟΡΩΝ

Τύποι Πόρων (ResourcesTypes)	Διαθεσιμότητα (Availability)
Έμπειροι Πολιτικοί Μηχανικοί	4
Έμπειροι Τοπογράφοι Μηχανικοί	9
Έμπειροι Δικηγόροι	10
Έμπειροι Προγραμματιστές	2
GIS	1
Τοπογράφοι	21
Δικηγόροι	22
Προγραμματιστές	3
Λοιποί	18

Το κόστος κάθε τύπου πόρου φαίνεται στον πίνακα 10. Το κόστος αυτό θεωρείται άμεσο κόστος καθώς από τη φύση του έργου κάθε πόρος σχετίζεται με κάποια δραστηριότητα.

ΠΙΝΑΚΑΣ 10 ΚΟΣΤΟΣ ΤΥΠΩΝ ΠΟΡΩΝ

Τύποι Πόρων (ResourcesTypes)	Κόστος Ανά Ημέρα(8 ώρες) €/ημέρα
Έμπειροι Πολιτικοί Μηχανικοί	70
Έμπειροι Τοπογράφοι Μηχανικοί	65
Έμπειροι Δικηγόροι	70
Έμπειροι Προγραμματιστές	60
GIS	60
Τοπογράφοι	50
Δικηγόροι	60
Προγραμματιστές	50
Λοιποί	40

Για τον υπολογισμό του συνολικού κόστους του έργου δεν λήφθηκαν υπόψη έμμεσα κόστη καθώς δεν δινόταν κάποια πληροφορία για αυτά από το αρχικό πρόβλημα. Οπότε η συνάρτηση κόστους είναι η εξής:

$$\boldsymbol{F_{c1}} = \sum_{j \in V} \sum_{m \in M_j} (x_{jm} \quad * \sum_{k \in K_{jm}} (ed_{im} * c_k)) \qquad (1)$$

$$\boldsymbol{F_{c2}} = \sum_{j \in V} \sum_{m \in M_j} (x_{jm} \quad * \sum_{k \in K_{jm}} (nd_{im} * n_k)) \qquad (2)$$

$$\boldsymbol{F_c = F_{c1} + F_{c2} + penalty} \qquad \boldsymbol{(3)}$$

Αντίστοιχα με το αρχικό πρόβλημα, η συνάρτηση (1) είναι το κόστος μιας δραστηριότητας η οποία εκτελείται σε συμπιεσμένη διάρκεια. Η συνάρτηση (2) είναι το κόστος μιας δραστηριότητας που εκτελείται σε κανονική διάρκεια. Η συνάρτηση (3) είναι το συνολικό κόστος, το οποίο αποτελείται, όπως αναφέραμε μόνο από το άμεσο κόστος των δραστηριοτήτων και προστίθεται σε αυτήν μια ποινή εάν υπερβεί το πρόγραμμα μια συγκεκριμένη προθεσμία (Tmax)

Διαφορές του πραγματικού προβλήματος με το MRC-DTCTP της εργασίας μας.

Στο αρχικό πρόβλημα, κάθε δραστηριότητα εκτελείται με έναν συγκεκριμένο τρόπο. Για την εφαρμογή του κώδικα πρέπει να παραμετροποιηθούν τα δεδομένα έτσι ώστε να δημιουργούνται πολλαπλοί τρόποι εκτέλεσης. Για να ερευνηθεί η σχέση χρόνου-κόστους πρέπει στο αρχικό πρόβλημα να εισαχθούν παράμετροι όπως η συμπίεση δραστηριοτήτων. Το αρχικό πρόβλημα θέτει την δυσκολία ένα σκαλί ψηλότερα όσο αφορά τον τρόπο με τον οποίο συσχετίζονται οι αλληλουχίες των δραστηριοτήτων (εικόνα 35) καθώς υπάρχουν σχέσεις αλληλουχίας τύπου Start-Start, Start-Finish, Finish-Finish. Στο κλασσικό πρόβλημα RCPSP, όλες οι σχέσεις είναι τέλους-αρχής (Finish to Start – FS). Όμως, το αρχικό έργο μπορεί να αναπαρασταθεί με ένα δικτυωτό γράφημα όπου οι δραστηριότητες απεικονίζονται με κόμβους και οι σχέσεις αλληλουχίας τους με βέλη. Οι σχέσεις αλληλουχίας περιγράφονται στην παράγραφο 2.2.2. Για την μετατροπή αυτών των σχέσεων έτσι ώστε να προσαρμόζονται στις ανάγκες του συστήματός, δηλαδή σε τύπου Τέλους-Αρχής χωρίς καθυστέρηση, δεν χρησιμοποιήθηκε κάποιος μαθηματικός κανόνας αλλά δημιουργήθηκαν επιπλέον μεταβλητές στον κώδικά και λύθηκε το πρόβλημα προγραμματιστικά. Πιθανή επίλυση τέτοιου προβλήματος με κάποιον μηχανισμό μετατροπής Δικτυωτού Γραφήματος MPM σε CPM, θα απαιτούσε δημιουργία επιπλέον βοηθητικών δραστηριοτήτων, το οποίο συνεπάγεται επιπλέον υπολογιστικός χρόνος και μεγαλύτερη δυσκολία επίλυσης του προβλήματος από την ήδη υπάρχουσα. Σημαντικό στοιχείο είναι το ότι το πρόβλημά αποτελείται ήδη από αρκετές δραστηριότητες και επιπλέον αύξηση του αριθμού τους θα μας πήγαινε ένα βήμα πίσω παρά εμπρός.

	Task Name
0	**Έργο Κτηματογράφησης**
1	**1ο ΣΤΑΔΙΟ: Δημιουργία Προκαταρκτικού Υποβάθρου – Έναρξη λειτουργίας των Γραφείων Κτηματογράφησης και πιστοποίηση από την Κτηματολόγιο Α.Ε. της ορθής λειτουργίας του.**
2	Έναρξη σύμβασης / Υπογραφή σύμβασης
3	**ΦΑΣΗ 1.1 Προκαταρκτικές εργασίες / Εκκίνηση του έργου**
4	**Εναρκτήριες εργασίες**
5	Έναρξη 1ου Σταδίου Σύμβασης
6	Ανάλυση έργου/ Προετοιμασία για την εναρκτήρια συνάντηση
7	Εναρκτήρια συνάντηση σύμβασης
8	Παραλαβή στοιχείων από ΚΤΗΜΑΤΟΛΟΓΙΟ Α.Ε
9	**Προετοιμασία , εξοπλισμός, έλεγχος συστημάτων πληροφορικής Μονάδας Διαχείρισης Δεδομένων**
10	Προμήθεια πρόσθετου υλικοτεχνικού εξοπλισμού, Software

ΕΙΚΟΝΑ 34 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ ΒΑΣΙΚΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

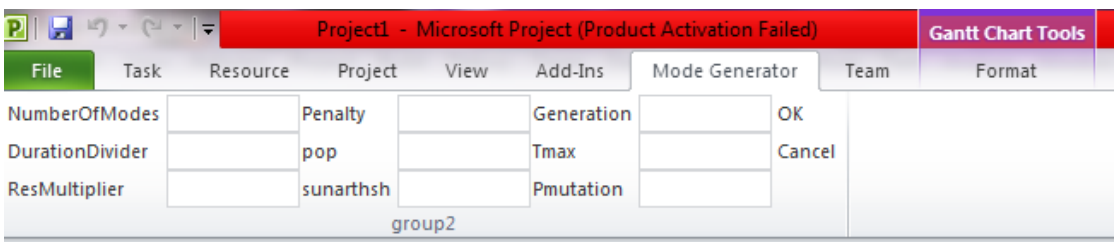
172	Σύνταξη και εκτύπωση Κτηματολογικών Διαγραμμάτων, Πινάκων, αλφαβητικών ευρετήριων και Αποσπασμάτων της Ανάρτησης. Αποστολή Αποσπασμάτων	22 days	Wed 8/10/14	Fri 7/11/14	154FS+23 days
173	Σύνταξη Τεχνικής έκθεσης «σύνταξης των κτηματολογικών διαγραμμάτων και πινάκων της Ανάρτησης» (Τ.Π. 5.1.4)	8 days	Wed 29/10/14	Fri 7/11/14	172SS+14 days
174	Υποβολή Κτηματολογικών Διαγραμμάτων, Πινάκων (και αλφαβητικά ευρετήρια) και ψηφιακών αρχείων σχετικά με αποστολή αποσπασμάτων της Ανάρτησης	0 days	Fri 7/11/14	Fri 7/11/14	172
175	Υποβολή Τεχνικής έκθεσης σύνταξης των κτηματολογικών διαγραμμάτων και πινάκων της Ανάρτησης	0 days	Fri 7/11/14	Fri 7/11/14	173
176	**Διενέργεια Ανάρτησης και διαδικασία παραλαβής ενστάσεων και διαβίβαση των φακέλων στις επιτροπές**	**148 days**	**Fri 7/11/14**	**Tue 9/6/15**	
177	**Έναρξη Ανάρτησης**	**0 days**	**Fri 7/11/14**	**Fri 7/11/14**	**174;154FS+44 days;171;175**
178	Παραλαβή ενστάσεων και αιτήσεων διόρθωσης της ανάρτησης κατοίκων εσωτερικού και εξωτερικού	45 days	Fri 7/11/14	Tue 13/1/15	177FS-1 day

ΕΙΚΟΝΑ 35 ΣΧΕΣΕΙΣ ΑΛΛΗΛΟΥΧΙΑΣ ΒΑΣΙΚΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

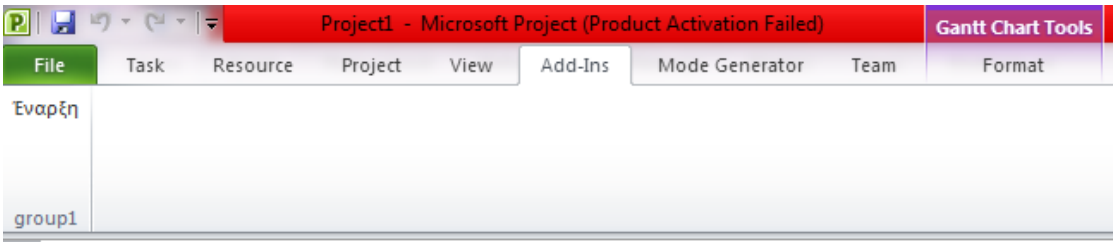
Στο αρχικό πρόβλημα, στο σύνολο των 243 δραστηριοτήτων υπάρχουν αρκετές ενοποιημένες δραστηριότητες (summary tasks) (εικόνα 34). Όταν μία ομάδα δραστηριοτήτων έχει κοινό σημείο έναρξης και κοινό σημείο λήξης, είναι δυνατόν να αντικατασταθεί ολόκληρη η ομάδα από μια ενιαία δραστηριότητα, η οποία ονομάζεται ενοποιημένη δραστηριότητα. Για την προσαρμογή των δεδομένων στο κώδικα έπρεπε να αφαιρεθούν αυτές τις δραστηριότητες, έτσι ώστε να μείνουν μόνο οι δραστηριότητες που έχουν όλες τις πληροφορίες που χρειάζονται όπως διάρκεια, προαπαιτούμενες δραστηριότητες, πόρους κλπ.

8.2 Διεπαφή μέσω Microsoft Office

Για την εκτέλεση του αλγορίθμου, τα δεδομένα του έργου εισάγονται μέσω του Microsoft Project. Για την εκτέλεση του προβλήματος MRC-DTCTP, επιπλέον παράμετροι έπρεπε να εισαχθούν. Οι παράμετροι αυτοί εισάγονται μέσω του Microsoft Project πάλι. Σχεδιάστηκε πρόγραμμα κατά το οποίο ο χρήστης εισάγει αυτές τις παραμέτρους, και ο τρόπος εισαγωγής γίνεται μέσω της μπάρας εργασιών του MS Project. Ο κώδικας σχεδιάστηκε σε γλώσσα προγραμματισμού C#, σε προγραμματιστικό περιβάλλον Visual Studio 2010. Η διεπαφή χρήστη-προγράμματος φαίνεται στις εικόνες 36, 37.



ΕΙΚΟΝΑ 36 ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟ MS PROJECT



ΕΙΚΟΝΑ 37 ΕΙΣΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟ MS PROJECT

Για την εκτέλεση του προγράμματος ο χρήστης εισάγει τα δεδομένα από την καρτέλα “Mode Generator”, πατάει το κουμπί “OK” και έπειτα πατά την επιλογή “˙˙Έναρξη” από την καρτέλα “Add-Ins”.

8.3 Επιλογή Κριτηρίων – Μεταβλητών

Για την προσαρμογή του αρχικού προβλήματος στο δικό μας, το MRC-DTCTP, όπως προαναφέραμε έπρεπε να εισαχθούν επιπλέον παράμετροι. Οι παράμετροι αυτοί εισάγονται από το χρήστη μέσω της διεπαφής που δημιουργήθηκε και φαίνεται στην εικόνα 34. Οι παράμετροι αυτοί είναι οι:

1. NumberOfModes
2. DurationDivider
3. ResMultiplier
4. Penalty
5. Pop
6. Generation
7. Sunarthsh
8. Tmax
9. Pmutation

Οι παράμετροι 4-9 είναι οι ίδιοι που εισάγονται και στο αρχικό πρόβλημα, με τη μόνη διαφορά της $7^{ης}$ παραμέτρου. Ο χρήστης για αυτή τη πληροφορία επιλέγει είτε 1 είτε 2. Αυτή η επιλογή αποτελεί παράμετρο του συστήματος ως προς τον τρόπο λύσης του προβλήματος. Στην επίλυση του πρακτικού προβλήματος ο στόχος ήταν η ελαχιστοποίηση του χρόνου. Η επιλογή «2» αντικατοπτρίζει αυτόν τον στόχο. Η επιλογή «1» αποτελεί παράμετρο για ελαχιστοποίηση του κόστους.

Οι παράμετροι 1-3 είναι καινούργιοι και χρησιμοποιούνται για να μετατρέψουν το αρχικό πρόβλημα προγραμματισμού του έργου που μπορεί να θεωρηθεί RCPSP σε MRC-DTCTP. Συγκεκριμένα η παράμετρος 1 είναι ο αριθμός των τρόπων

εκτέλεσης (modes) που θέλει ο χρήστης να δημιουργήσει για κάθε δραστηριότητα. Στο αρχικό πρόβλημα μόνο ένας τρόπος υπήρχε. Επειδή κάθε τρόπος εκτέλεσης σημαίνει διαφορετική αναγκαιότητα σε πόρους και διαφορετική διάρκεια, οι περιορισμοί 2 και 3 χρησιμοποιούνται για την προσαρμογή αυτή. Δηλαδή, για κάθε τρόπο εκτέλεσης που επιλέγεται η αρχική διάρκεια της δραστηριότητας διαιρείται με τον αριθμό DurationDivider και οι πόροι που της αναλογούν πολλαπλασιάζονται με τον αριθμό ResMultiplier. Οπότε μια δραστηριότητα η οποία είχε αρχικά έναν τρόπο εκτέλεσης με τους παρακάτω πόρους.

ΠΙΝΑΚΑΣ 11

Δραστηριότητα	Διάρκεια	Ποσότητα Απαιτούμενων Πόρων	
		Πληροφορική-GIS	
1	3	1	1

Εάν ο χρήστης επιλέξει τις παρακάτω παραμέτρους, θα έχει τις παρακάτω δυνατές επιλογές εκτέλεσης με τα δεδομένα που φαίνονται στον πίνακα 13.

ΠΙΝΑΚΑΣ 12

NumberOfModes	**3**
DurationDivider	2
ResMultiplier	2

ΠΙΝΑΚΑΣ 13

Δραστηριότητα	Τρόπος Εκτέλεσης	Διάρκεια	Ποσότητα Απαιτούμενων Πόρων	
			Πληροφορική-GIS	
1	1	12	1	1
	2	6	2	2
	3	3	4	4

Ελαττώνοντας τη διάρκεια μιας δραστηριότητας, αυξάνονται τα κόστη της. Επειδή δεν θεωρούμε ως μεταβλητή του προβλήματός μας την ποιότητα κάθε πόρου, θεωρούμε ότι τα κόστη αυξάνονται αναλογικά με την τοποθέτηση περισσότερων πόρων σε μια δραστηριότητα. Δηλαδή, στη δραστηριότητα στον παρακάτω πίνακα εκτελείτε σε 15 ημέρες με το κόστος της να ανέρχεται στα

1.830,48€ (πίνακας 14) αλλά μπορεί να εκτελεσθεί και σε 5 ημέρες με κόστος 5.491,44 € (πίνακας 15), τοποθετώντας περισσότερους πόρους σε αυτήν.

Πίνακας 14 Επιλογή 1

Task Name	Duration	Cost
Εγκατάσταση συστημάτων πυρανίχνευσης και ασφάλειας	15 days	1.830,48 €

Πίνακας 15 Επιλογή 2

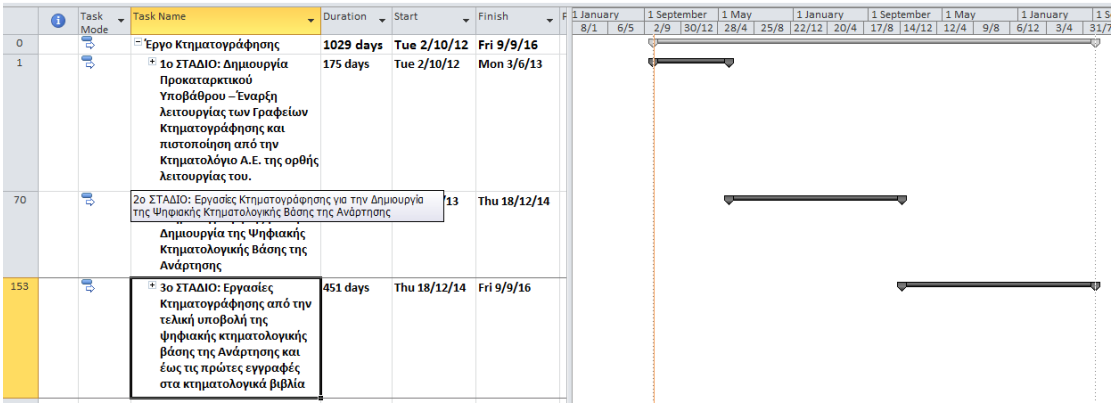
Task Name	Duration	Cost
Εγκατάσταση συστημάτων πυρανίχνευσης και ασφάλειας	5 days	5.491,44 €.

Επίσης, εάν η δραστηριότητα αυτή επιλεγεί να εκτελεσθεί σε συμπιεσμένη διάρκεια τότε, το κόστος της αυξάνεται πολλαπλασιάζοντας το με ένα συντελεστή ψ=2. Εάν ο Τοπογράφος Μηχανικός, για παράδειγμα, πληρωνόταν με 7,5€/ώρα για κανονική διάρκεια της δραστηριότητας, για συμπιεσμένη θα πληρωθεί με 15€/ώρα (υπερωρία). Καθώς αυξάνεται χρόνος που απαιτείται να εκτελεσθεί κάθε δραστηριότητα με τους επιπλέον τρόπους εκτέλεσης, είναι αναγκαίο να αυξηθούν οι διαθεσιμότητες των πόρων.

8.4 Ανάλυση Αποτελεσμάτων

Τα ακόλουθα αποτελέσματα υπολογίστηκαν από υπολογιστή με επεξεργαστή Intel® Core(TM) i5-2410M CPU @ 2.30GHz 2.30GHz και 2,67 GB RAM. Ο γενετικός αλγόριθμος κατασκευάστηκε σε γλώσσα C# στο προγραμματιστικό περιβάλλον Visual Studio 2010.

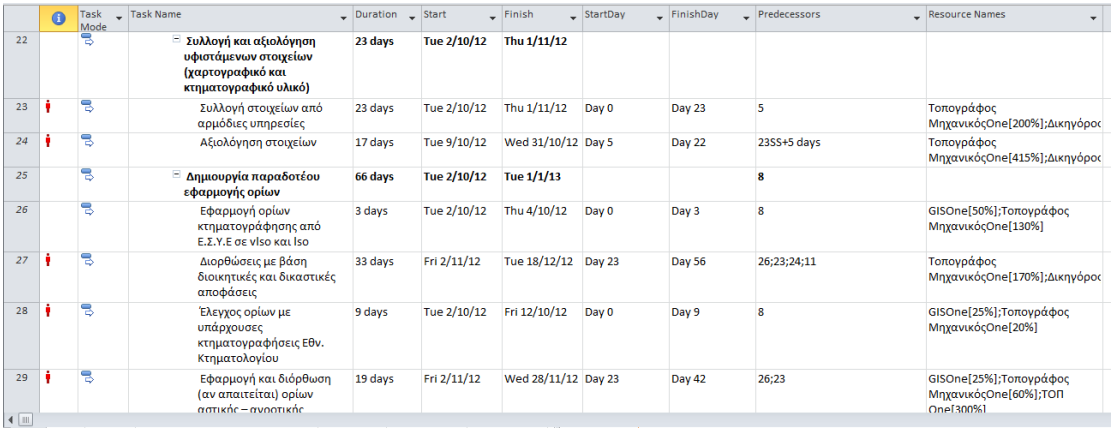
Αρχικά θα αναλυθούν τα αποτελέσματα του προβλήματος με στόχο την ελαχιστοποίηση της συνολικής διάρκειας, με έναν τρόπο εκτέλεσης και τα δεδομένα που δίνονται από το βασικό πρόγραμμα. Η συνολική διάρκεια του έργου, προγραμματίζοντας τις δραστηριότητες στην νωρίτερη έναρξή τους χωρίς να λάβουμε υπόψη τους πόρους είναι ίση με **1029** ημέρες σύμφωνα με το MS Project (εικόνα 38). Είναι η αντίστοιχη λύση που θα έδινε η μέθοδος του κρίσιμου δρόμου (Critical Path Method), η οποία δεν λαμβάνει υπόψη της τους πόρους.



	Task Mode	Task Name	Duration	Start	Finish
0		Έργο Κτηματογράφησης	1029 days	Tue 2/10/12	Fri 9/9/16
1		1ο ΣΤΑΔΙΟ: Δημιουργία Προκαταρκτικού Υποβάθρου – Έναρξη λειτουργίας των Γραφείων Κτηματογράφησης και πιστοποίηση από την Κτηματολόγιο Α.Ε. της ορθής λειτουργίας του.	175 days	Tue 2/10/12	Mon 3/6/13
70		2ο ΣΤΑΔΙΟ: Εργασίες Κτηματογράφησης για την Δημιουργία της Ψηφιακής Κτηματολογικής Βάσης της Ανάρτησης	13	Thu 18/12/14	
		Δημιουργία της Ψηφιακής Κτηματολογικής Βάσης της Ανάρτησης			
153		3ο ΣΤΑΔΙΟ: Εργασίες Κτηματογράφησης από την τελική υποβολή της ψηφιακής κτηματολογικής βάσης της Ανάρτησης και έως τις πρώτες εγγραφές στα κτηματολογικά βιβλία	451 days	Thu 18/12/14	Fri 9/9/16

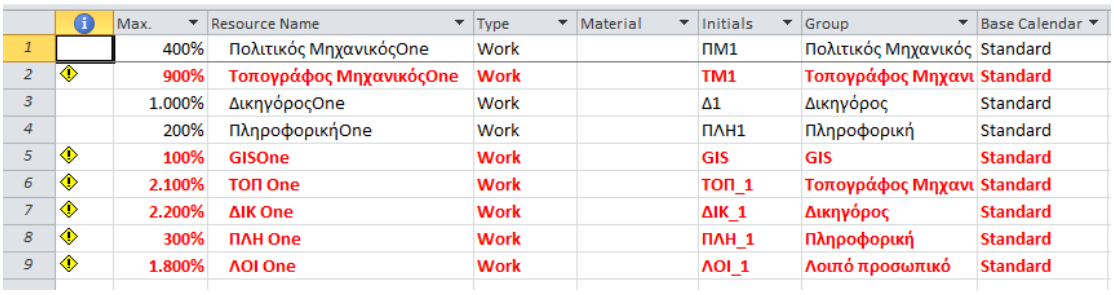
ΕΙΚΟΝΑ 38 ΔΙΑΡΚΕΙΑ ΕΡΓΟΥ ΣΤΗ ΝΩΡΙΤΕΡΗ ΕΝΑΡΞΗ ΤΩΝ ΔΡΑΣΤΗΡΙΟΤΗΤΩΝ

Στην εικόνα 39 και η εικόνα 40 φαίνεται το έργο με τους πόρους και τις απαιτούμενες ποσότητες για την εκτέλεση των δραστηριοτήτων. Όπως φαίνεται από την εικόνα δημιουργούνται προβλήματα διαθεσιμότητας των πόρων όταν οι δραστηριότητες εκτελούνται στην νωρίτερη έναρξή τους.



	Task Mode	Task Name	Duration	Start	Finish	StartDay	FinishDay	Predecessors	Resource Names
22		Συλλογή και αξιολόγηση υφιστάμενων στοιχείων (χαρτογραφικό και κτηματογραφικό υλικό)	23 days	Tue 2/10/12	Thu 1/11/12				
23		Συλλογή στοιχείων από αρμόδιες υπηρεσίες	23 days	Tue 2/10/12	Thu 1/11/12	Day 0	Day 23	5	Τοπογράφος ΜηχανικόςOne[200%];Δικηγόρος
24		Αξιολόγηση στοιχείων	17 days	Tue 9/10/12	Wed 31/10/12	Day 5	Day 22	23SS+5 days	Τοπογράφος ΜηχανικόςOne[415%];Δικηγόρος
25		Δημιουργία παραδοτέου εφαρμογής ορίων	66 days	Tue 2/10/12	Tue 1/1/13			8	
26		Εφαρμογή ορίων κτηματογράφησης από Ε.Σ.Υ.Ε σε vIso και Iso	3 days	Tue 2/10/12	Thu 4/10/12	Day 0	Day 3	8	GISOne[50%];Τοπογράφος ΜηχανικόςOne[130%]
27		Διορθώσεις με βάση διοικητικές και δικαστικές αποφάσεις	33 days	Fri 2/11/12	Tue 18/12/12	Day 23	Day 56	26;23;24;11	Τοπογράφος ΜηχανικόςOne[170%];Δικηγόρος
28		Έλεγχος ορίων με υπάρχουσες κτηματογραφήσεις Εθν. Κτηματολογίου	9 days	Tue 2/10/12	Fri 12/10/12	Day 0	Day 9	8	GISOne[25%];Τοπογράφος ΜηχανικόςOne[20%]
29		Εφαρμογή και διόρθωση (αν απαιτείται) ορίων αστικής – αγροτικής	19 days	Fri 2/11/12	Wed 28/11/12	Day 23	Day 42	26;23	GISOne[25%];Τοπογράφος ΜηχανικόςOne[60%];ΤΟΠ One[300%]

ΕΙΚΟΝΑ 39

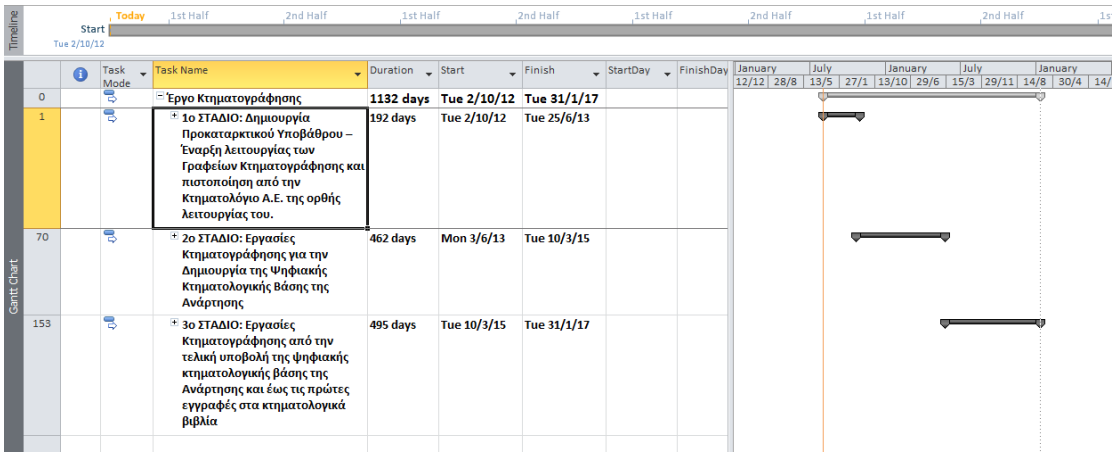


	Max.	Resource Name	Type	Material	Initials	Group	Base Calendar
1	400%	ΠολιτικόςΜηχανικόςOne	Work		ΠΜ1	Πολιτικός Μηχανικός	Standard
2	900%	ΤοπογράφοςΜηχανικόςOne	Work		ΤΜ1	Τοπογράφος Μηχανι	Standard
3	1.000%	ΔικηγόροςOne	Work		Δ1	Δικηγόρος	Standard
4	200%	ΠληροφορικήOne	Work		ΠΛΗ1	Πληροφορική	Standard
5	100%	GISOne	Work		GIS	GIS	Standard
6	2.100%	ΤΟΠ One	Work		ΤΟΠ_1	Τοπογράφος Μηχανι	Standard
7	2.200%	ΔΙΚ One	Work		ΔΙΚ_1	Δικηγόρος	Standard
8	300%	ΠΛΗ One	Work		ΠΛΗ_1	Πληροφορική	Standard
9	1.800%	ΛΟΙ One	Work		ΛΟΙ_1	Λοιπό προσωπικό	Standard

ΕΙΚΟΝΑ 40

Για την εξομάλυνση των πόρων και την εύρεση της συνολικής διάρκειας του έργου ώστε να μην υπάρχουν διενέξεις πόρων, χρησιμοποιώντας την επιλογή Leveling από το MS Project, η συνολική διάρκεια του έργου βρέθηκε ίση με **1148**.

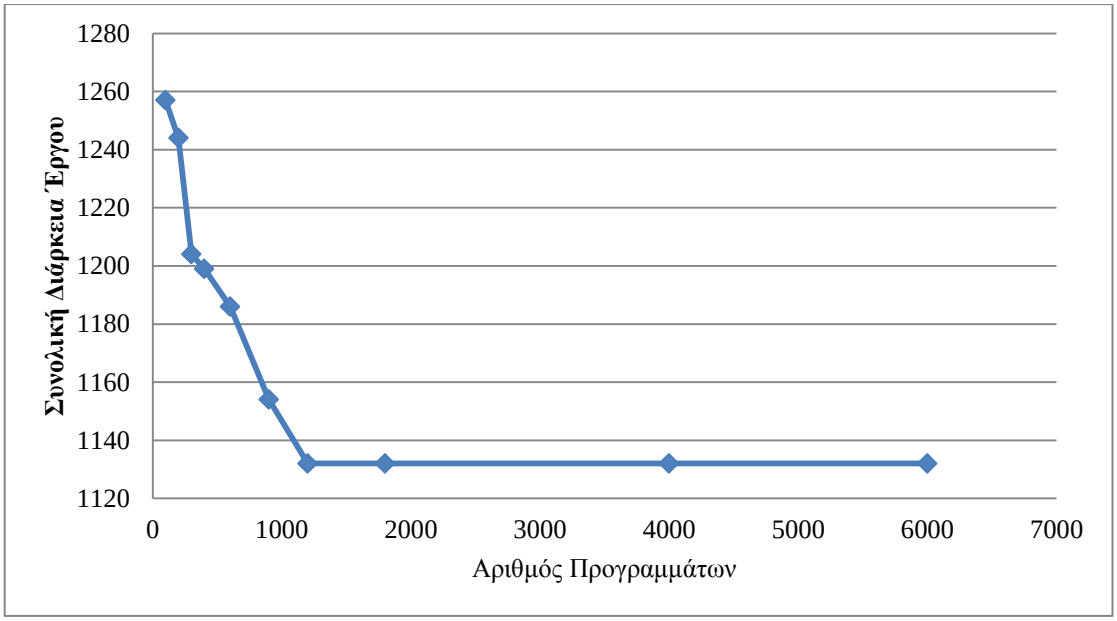
Για την εύρεση των αποτελεσμάτων με τον αλγόριθμο του δημιουργήθηκε ο αλγόριθμος εκτελέσθηκε με διαφορετικό αριθμό προγραμμάτων έτσι ώστε να αποκτηθούν τα αποτελέσματα που φαίνονται στον πίνακα 16. Η βέλτιστη λύση βρίσκεται όταν ο αριθμός των προγραμμάτων είναι ίσος με 1200 (POP=40, GEN=30), σε λιγότερο από 2 λεπτά (98 sec) και είναι ίση με 1132 ημέρες. Από εκεί και πέρα η λύση σταθεροποιείται σε αυτήν την τιμή. Ο χρόνος τρεξίματος καθώς αυξάνεται ο αριθμός των προγραμμάτων για κάθε επανάληψη, αυξάνεται εκθετικά όπως φαίνεται στο διάγραμμα στην εικόνα 43.



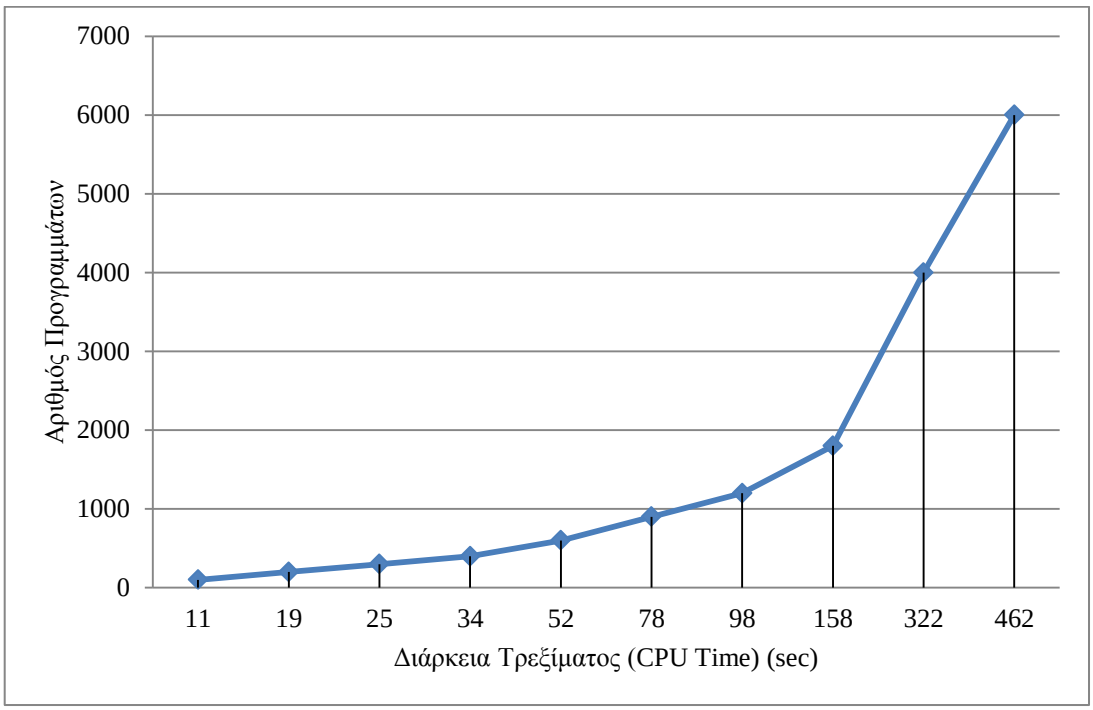
ΕΙΚΟΝΑ 41 ΑΠΟΤΕΛΕΣΜΑ ΔΙΑΡΚΕΙΑΣ ΕΡΓΟΥ ΜΕ ΓΕΝΕΤΙΚΟ ΑΛΓΟΡΙΘΜΟ

ΠΙΝΑΚΑΣ 16 ΑΠΟΤΕΛΕΣΜΑΤΑ ΓΕΝΕΤΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ ΓΙΑ ΤΗΝ ΠΡΑΚΤΙΚΗ ΕΦΑΡΜΟΓΗ

Schedules	POP	GEN	Συνολική Διάρκεια Έργου	CPU Time (sec)
100	10	10	1257	11
200	20	10	1244	19
300	30	10	1204	25
400	20	20	1199	34
600	30	20	1186	52
900	30	30	1154	78
1200	40	30	1132	98
1800	60	30	1132	158
4000	80	50	1132	322
6000	100	60	1132	462

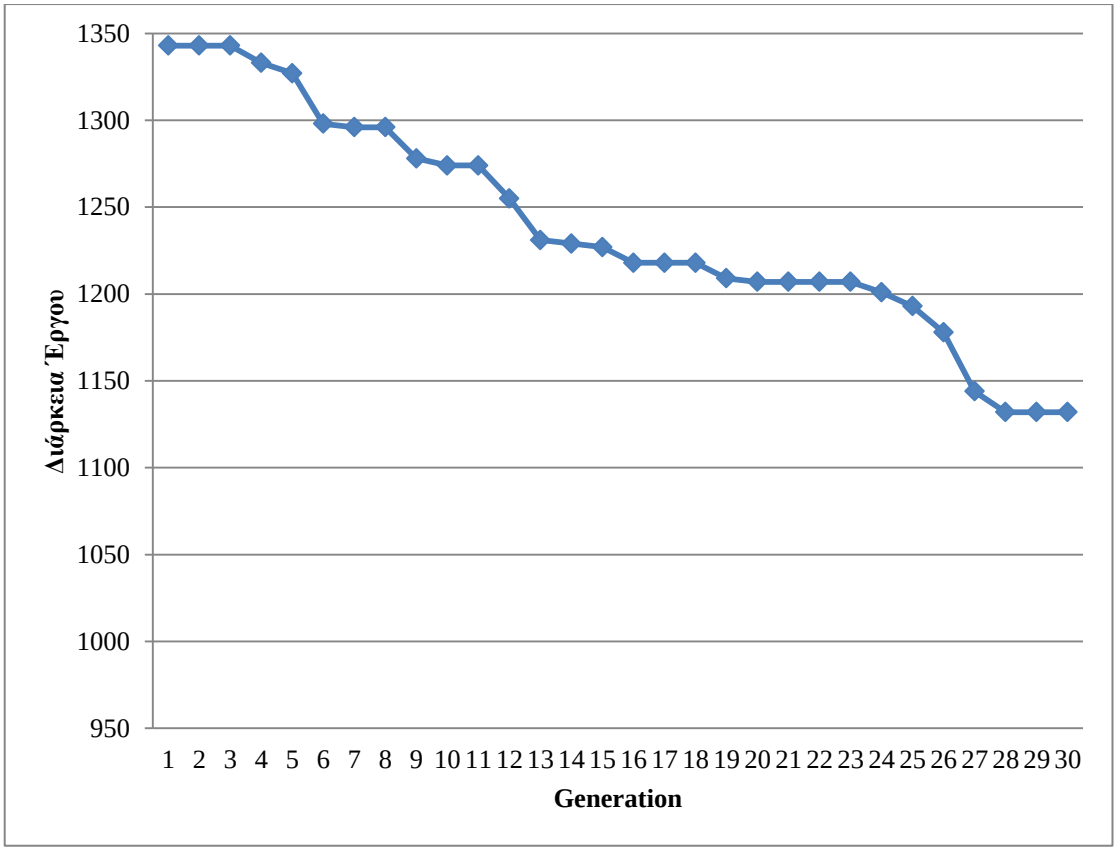


ΕΙΚΟΝΑ 42 ΔΙΑΓΡΑΜΜΑ ΣΥΝΟΛΙΚΗΣ ΔΙΑΡΚΕΙΑΣ ΕΡΓΟΥ – ΑΡΙΘΜΟΥ ΠΡΟΓΡΑΜΜΑΤΩΝ



ΕΙΚΟΝΑ 43 ΔΙΑΓΡΑΜΜΑ ΔΙΑΡΚΕΙΑΣ ΤΡΕΞΙΜΑΤΟΣ ΑΛΓΟΡΙΘΜΟΥ-ΑΡΙΘΜΟ ΠΡΟΓΡΑΜΜΑΤΩΝ

Από το διάγραμμα της εικόνας 43 παρατηρείτε ότι καθώς ο αριθμός των Generation αυξάνεται η συνολική διάρκεια του έργου σταθεροποιείται σε κάποιες τιμές, οι οποίες αποτελούν τοπικά βέλτιστα για το πρόβλημά μας. Η καλή λειτουργία του γενετικού αλγορίθμου φαίνεται από το διάγραμμα στην εικόνα καθώς ξεπερνάει τελικά τις λύσεις αυτές και φθάνει στη βέλτιστη (1132). Αυτό σημαίνει ότι η μετάλλαξη και η πιθανότητα μετάλλαξης που επιλέχθηκαν για την επίλυση του προβλήματος έχουν ικανοποιητικά αποτελέσματα.



ΕΙΚΟΝΑ 43 ΔΙΑΓΡΑΜΜΑ ΔΙΑΡΚΕΙΑΣ ΕΡΓΟΥ-ΑΡΙΘΜΟΥ GENERATION

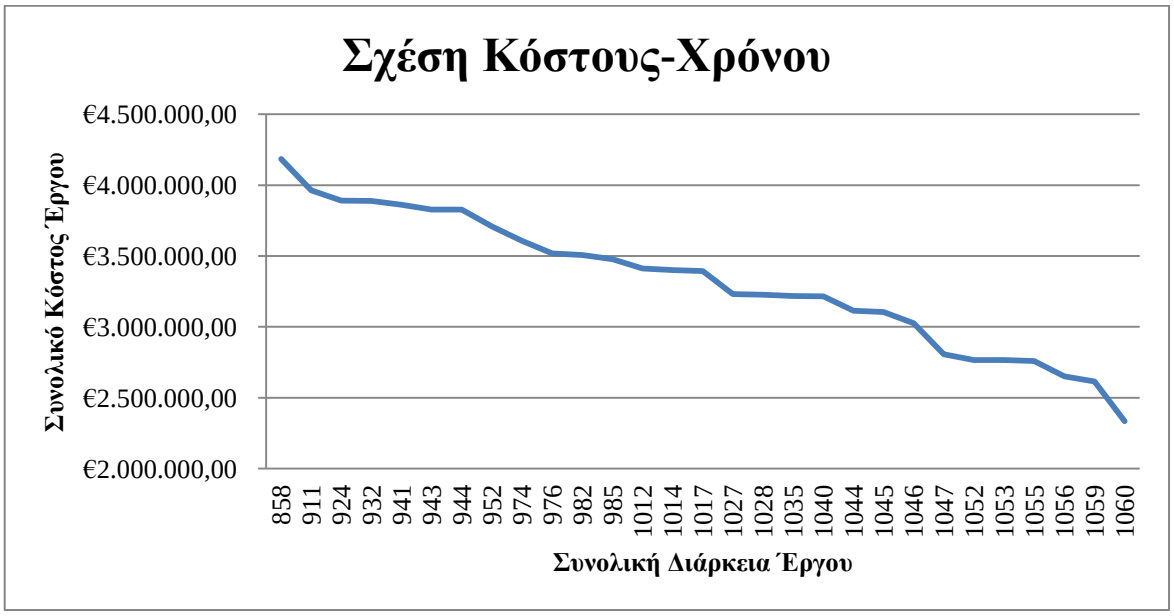
Για την εξαγωγή αποτελεσμάτων προσαρμόζοντας το πρακτικό πρόβλημα σε πρόβλημα τύπου ***MRC-DTCTP***, θεωρήσαμε τα δεδομένα που φαίνονται στον Πίνακα 12. Οπότε για κάθε δραστηριότητα από τις 243 του προβλήματος, υπάρχουν 3 πιθανοί τρόποι εκτέλεσης και η επιλογή για κανονική ή συμπιεσμένη διάρκεια. Αυτές οι επιλογές αυξάνουν εκθετικά την πολυπλοκότητα του αλγορίθμου με αποτέλεσμα ο χρόνος εύρεσης ενός προγράμματος από τον αλγόριθμο να αυξάνεται εκθετικά.

Το κόστος του έργου από τη βασική λύση του ανέρχεται στα 1.968.975,48€ και η διάρκεια του έργου στις 1148 ημέρες. Οι συνδυασμοί χρόνου-κόστους του έργου με πολλαπλούς τρόπους εκτέλεσης και συμπιεσμένη διάρκεια δίνονται στον πίνακα 16. Η ποινή για καθυστέρηση του έργου παραπάνω από 1148 ημέρες τέθηκε ίση με 500.000€.

ΠΙΝΑΚΑΣ 16 ΣΧΕΣΗ ΧΡΟΝΟΥ-ΚΟΣΤΟΥΣ ΕΡΓΟΥ

Διάρκεια	Κόστος Έργου
858	4.183.941,00 €
911	3.962.488,00 €
924	3.891.187,00 €
932	3.889.052,00 €
941	3.861.696,00 €

943	3.827.938,00 €
944	3.826.911,00 €
952	3.708.023,00 €
974	3.605.166,00 €
976	3.519.194,00 €
982	3.505.724,00 €
985	3.478.047,00 €
1012	3.412.248,00 €
1014	3.401.737,00 €
1017	3.393.466,00 €
1027	3.230.787,00 €
1028	3.226.440,00 €
1035	3.217.510,00 €
1040	3.215.738,00 €
1044	3.113.626,00 €
1045	3.105.150,00 €
1046	3.025.356,00 €
1047	2.805.783,00 €
1052	2.765.228,00 €
1053	2.765.228,00 €
1055	2.759.012,00 €
1056	2.650.519,00 €
1059	2.614.351,00 €
1060	2.334.622,00 €



ΕΙΚΟΝΑ 44 ΣΧΕΣΗ ΧΡΟΝΟΥ-ΚΟΣΤΟΥΣ ΕΡΓΟΥ

Τα αποτελέσματα αυτά και η χρήση του αλγορίθμου με τις παραλλαγές μέσω της εφαρμογής των δεδομένων από το MS Project, μπορούν να παρέχουν χρήσιμες

πληροφορίες στον υπεύθυνο του έργου για τη σύγκριση και αξιολόγηση διαφορετικών προγραμμάτων.

8.5 Περιορισμοί

Για τη λειτουργία του αλγορίθμου, μέσω της διεπαφής που δημιουργήθηκε, είναι απαραίτητοι κάποιοι περιορισμοί.

Οι δραστηριότητες από τις οποίες αποτελείται το έργο πρέπει να έχουν διαφορετικό όνομα (Task Name) η κάθε μία.

Ο αριθμός που εισάγει ο χρήστης για τον αρχικό πληθυσμό πρέπει να είναι ζυγός.

Ο χρήστης μπορεί να εισάγει και δεκαδικές τιμές για τα τις επιλογές ResMultiplier και DurationDivider, έτσι ώστε να μπορεί να μειώσει ή να αυξήσει τη διάρκεια του έργου, αυξάνοντας ή μειώνοντας αντίστοιχα τους πόρους για κάθε δραστηριότητα.

Για την περαιτέρω μείωση της διάρκειας του έργου και το σχεδιασμό της καμπύλης αντιστάθμισης κόστους-χρόνου απαιτείται η αύξηση της διαθεσιμότητας των πόρων.

9 Συμπεράσματα – Κατευθύνσεις περαιτέρω έρευνας

Οι γενετικοί αλγόριθμοι είναι ένα εξαιρετικό εργαλείο για την επίλυση του προβλήματος χρονοπρογραμματισμού. Έχουν απλή δομή και είναι κατάλληλοι για την επίλυση προβλημάτων με μείγμα συνεχής και διακριτές μεταβλητές. Ωστόσο, η εφαρμογή τους δεν είναι καθόλου ασήμαντη. Παρόλο που οι βασικές του ιδέες είναι απλές, υπάρχει πολύ δρόμος στο μέλλον για την εφαρμογή των γενετικών αλγορίθμων σε πραγματικά προβλήματα με μεγάλους χώρους αναζήτησης λύσεων.

Η λειτουργία του αλγορίθμου σε προβλήματα της PSPLIB δεν ήταν δυνατή για τα προβλήματα DTCTP και MRC-DTCTP καθώς δεν υπάρχουν συγκρίσιμα αποτελέσματα άλλων αλγορίθμων αφού το πρόβλημα που μελετήσαμε είναι καινούργιο στον ερευνητικό τομέα και δεν έχουν δημιουργηθεί πολλοί αλγόριθμοι για την μελέτη του.

Για την αποτελεσματικότητα του αλγορίθμου στο πρόβλημα του MRCPSP βρέθηκαν αποτελέσματα και συγκρίθηκαν με τα βέλτιστα που δίνονται από τη βιβλιογραφία. Αυτά είναι αρκετά ικανοποιητικά ως προς τη συχνότητα εύρεσης της βέλτιστης λύσης και της σχετικής μέσης απόκλισης από τη αυτή.

Όσον αφορά την πρακτική εφαρμογή, βρέθηκε βέλτιστη λύση του προβλήματος, δηλαδή ελαχιστοποίηση της συνολικής διάρκειας του έργου, αρκετά μικρότερη από την δοσμένη, ενώ ο χρόνος εύρεσης αυτής ήταν λιγότερο από 6 λεπτά. Επίσης μελετήθηκε και η επίδραση της μείωσης της διάρκειας του έργου στο κόστος

Ιδιαίτερο ενδιαφέρον παρουσιάζει η μελέτη διαφορετικών παραμέτρων του γενετικού αλγορίθμου, όπως η διασταύρωση, η μετάλλαξη, η επιλογή και ο συγκερασμός διαφορετικών συνδυασμών τους για την μελέτη των αποτελεσμάτων.

10 ΒΙΒΛΙΟΓΡΑΦΙΑ

AHN, T. & ERENGUC, S. S. 1998. The resource constrained project scheduling problem with multiple crashable modes: A heuristic procedure. European Journal of Operational Research, 107, 250-259.

BAAR, #160, T., BRUCKER, P., KNUST & S. Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem.

BLAZEWICZ, J. 2007. Handbook on scheduling : from theory to applications ; with 28 tables, Berlin; Heidelberg; New York, Springer.

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

DAVIS, E. W. 1973. Project Scheduling under Resource Constraints—Historical Review and Categorization of Procedures. A I I E Transactions, 5, 297-313.

DAVIS, L. 1991. Handbook of genetic algorithms, Van Nostrand Reinhold.

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

DEMEULEMEESTER, E. L. & HERROELEN, W. S. 2002. Project Scheduling: A Research Handbook, Springer.

DEMEULEMEESTER, E. L., HERROELEN, W. S. & ELMAGHRABY, S. E. 1996. Optimal procedures for the discrete time/cost trade-off problem in project networks. European Journal of Operational Research, 88, 50-68.

DORIGO, M. & GAMBARDELLA, L. 1997. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1, 53-66.

ELMAGHRABY, S. E. & KAMBUROWSKI, J. 1992. The analysis of activity networks under generalized precedence relations (GPRs). Manage. Sci., 38, 1245-1263.

ERENGUC, S. S., AHN, T. & CONWAY, D. G. 2001. The resource constrained project scheduling problem with multiple crashable modes: An exact solution method. Naval Research Logistics (NRL), 48, 107-127.

GEN, M. & CHENG, R. 2000. Genetic Algorithms and Engineering Optimization, Wiley.

GLOVER, F. & TAILLARD, E. 1993. A user's guide to tabu search. Annals of Operations Research, 41, 1-28.

HARTMANN, S. 2001. Project Scheduling with Multiple Modes: A Genetic Algorithm. Annals of Operations Research, 102, 111-135.

HARTMANN, S. & BRISKORN, D. 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research, 207, 1-14.

HARTMANN, S. & KOLISCH, R. 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research, 127, 394-407.

HINDELANG, T. J. & MUTH, J. F. 1973. A Dynamic Programming Algorithm for Decision CPM Networks, Indiana University.

JAMES E. KELLEY, J. & WALKER, M. R. 1959. Critical-path planning and scheduling. Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference. Boston, Massachusetts: ACM.

KIRKPATRICK, S., GELATT, C. D. & VECCHI, M. P. 1983. Optimization by Simulated Annealing. Science, Number 4598, 13 May 1983, 220, 4598, 671-680.

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

KOLISCH, R. & DREXL, A. 1997. Local for multi-mode resource-constrained project. IIE Transactions (Institute of Industrial Engineers), 29, 987-999.

KOLISCH, R. & HARTMANN, S. 1999. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis.

LAND, A. H. & DOIG, A. G. 1960. An Automatic Method of Solving Discrete Programming Problems. Econometrica, 28, 497-520.

LEE, #160, J.-K., KIM & Y.-D. 1996. Search heuristics for resource constrained project scheduling, Basingstoke, ROYAUME-UNI, Palgrave Macmillan.

MCCORMICK JR, W. T., SCHWEITZER, P. J. & WHITE, T. W. 1972. PROBLEM DECOMPOSITION AND DATA REORGANIZATION BY A CLUSTERING TECHNIQUE. Operations Research, 20, 993-1009.

MERKLE, D. & MIDDENDORF, M. 2000. An Ant Algorithm with a New Pheromone Evaluation Rule for Total Tardiness Problems. Real-World Applications of Evolutionary Computing, EvoWorkshops 2000: EvoIASP, EvoSCONDI, EvoTel, EvoSTIM, EvoROB, and EvoFlight. Springer-Verlag.

MERKLE, D., MIDDENDORF, M. & SCHMECK, H. 2002. Ant colony optimization for resource-constrained project scheduling. Evolutionary Computation, IEEE Transactions on, 6, 333-346.

MICHALEWICZ, Z. 1998. Genetic Algorithms + Data Structures = Evolution Programs, Springer.

PANAGIOTAKOPOULOS, D. 1977. COST-TIME MODEL FOR LARGE CPM PROJECT NETWORKS. ASCE J Constr Div, 103, 201-211.

PETEGHEM, V. V. & VANHOUCKE, M. 2010. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. European Journal of Operational Research, 201, 409-418.

PROJECT MANAGEMENT INSTITUTE, S. 2008. A Guide to the Project Management Body of Knowledge (PMBOK Guide) [Online]. Exton: Project Management Institute. Available: http://encompass.library.cornell.edu/cgi-bin/checkIP.cgi?access=gateway_standard%26url=http://techbus.safaribooksonline.com/9781933890517 [Accessed].

SPRECHER, A., HARTMANN, S. & DREXL, A. 1997. An exact algorithm for project scheduling with multiple modes. OR Spectrum, 19, 195-203.

ST\, T., \#252, TZLE & HOOS, H. H. 2000. <italic>MAX-MIN</italic> Ant system. Future Gener. Comput. Syst., 16, 889-914.

VANHOUCKE, M. 2005. New computational results for the discrete time/cost trade-off problem with time-switch constraints. European Journal of Operational Research, 165, 359-374.

WULIANG, P. & CHENGEN, W. 2009. A multi-mode resource-constrained discrete time–cost tradeoff problem and its genetic algorithm based solution. International Journal of Project Management, 27, 600-609.

ΠΑΡΑΡΤΗΜΑ Α

****

file with basedata : mm2_.bas

initial value random generator: 1424959589

****

projects : 1

jobs (incl. supersource/sink): 12

horizon : 86

RESOURCES

- renewable : 2 R
- nonrenewable : 2 N
- doubly constrained : 0 D

****

PROJECT INFORMATION:

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

PRECEDENCE RELATIONS:

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

****

REQUESTS/DURATIONS:

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

****

RESOURCEAVAILABILITIES:

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

****

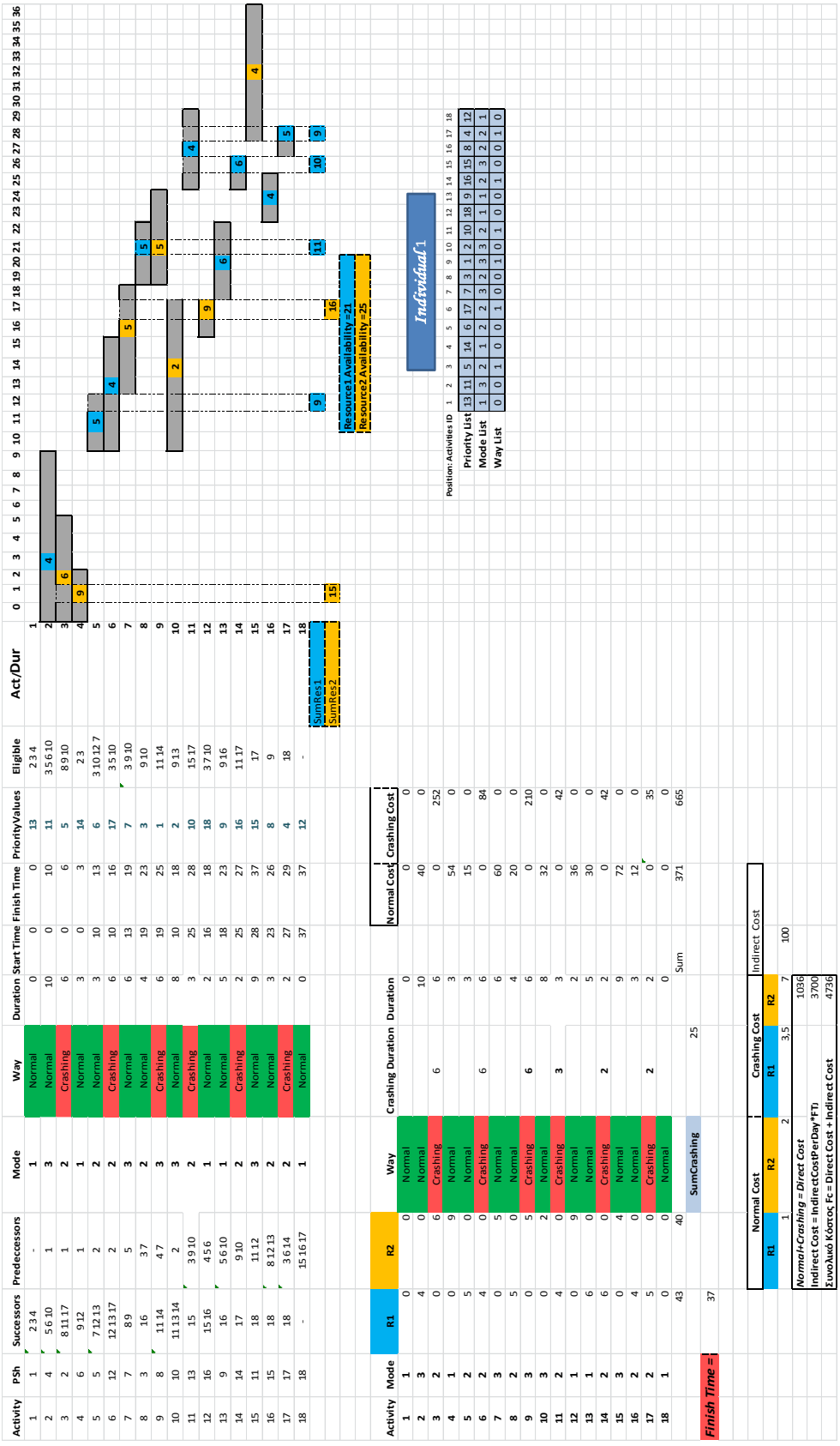
ΠΑΡΑΡΤΗΜΑ Β

FileName	Min Dur	Max Dur	Average	Optimal	Average.Relative.Dev.	Frequency Of Optimum	Jobs Number	Res Strength 1	Res Strength 2	Resource Factor
j102_2	20	23	20,78	20	4%	46%	12	0,2	0,5	0,75
j102_4	18	24	18,82	18	5%	56%	12	0,2	0,5	0,75
j102_5	17	17	17	16	6%	0%	12	0,2	0,5	0,75
j102_6	16	24	17,3	16	8%	38%	12	0,2	0,5	0,75
j102_7	25	32	25,9	25	4%	76%	12	0,2	0,5	0,75
j102_9	17	20	18,16	17	7%	38%	12	0,2	0,5	0,75
j102_10	33	33	33	33	0%	100%	12	0,2	0,5	0,75
j103_2	10	17	14,1	13	8%	38%	12	0,2	0,5	0,75
j103_3	19	23	19,68	19	4%	52%	12	0,2	0,5	0,75
j103_4	23	24	23,1	23	0%	90%	12	0,2	0,5	0,75
j103_5	21	23	21,34	21	2%	74%	12	0,2	0,5	0,75
j104_6	21	24	21,5	21	2%	64%	12	0,2	0,5	0,75
j105_10	25	30	25,76	25	3%	54%	12	0,2	1	0,75
j106_8	23	24	23	23	0,00	96%	12	0,2	1	0,75
j107_4	19	23	21	19	11%	14%	12	0,2	1	0,75
j107_5	24	28	25,07	24	4%	48%	12	0,2	1	0,75
j108_7	19	26	22,04	20	10%	13%	12	0,2	1	0,75
j108_10	23	26	24,02	23	4%	4%	12	0,2	1	0,75
j1010_1	17	23	17,93	17	5%	0,51	12	0,5	0,5	0,5
j1010_2	24	26	24,3	24	1%	0,71	12	0,5	0,5	0,5
j1011_6	15	18	15,55	15	4%	52%	12	0,5	0,5	0,5
j1011_7	11	17	13,22	11	20%	15%	12	0,5	0,5	0,5
j1012_2	15	19	16,12	15	7%	42%	12	0,5	0,5	0,5
j1012_4	17	20	17,12	17	1%	93%	12	0,5	0,5	0,5
j1013_1	24	29	24,71	24	3%	45%	12	0,5	1	0,75
j1013_2	21	22	21,63	21	3%	37%	12	0,5	1	0,75
j1014_9	24	29	24,7	24	3%	49%	12	0,5	1	0,75
j1014_10	15	20	17,7	15	18%	3%	12	0,5	1	0,75
j1015_4	15	21	16,04	15	7%	32%	12	0,5	1	0,75
j1015_5	11	15	12,63	11	15%	27%	12	0,5	1	0,75
j1016_3	12	15	12,72	12	6%	64%	12	0,5	1	0,75
j1016_6	20	25	21,48	21	2%	9%	12	0,5	1	0,75
j1018_3	17	18	17,29	17	2%	71%	12	0,7	0,5	0,5
j1018_10	16	18	16,4	16	2%	66%	12	0,7	0,5	0,5
j1019_5	13	18	14,02	13	8%	55%	12	0,7	0,5	0,5
j1019_6	17	21	17,48	17	3%	69%	12	0,7	0,5	0,5
j1020_9	9	11	9,27	9	3%	74%	12	0,7	0,5	0,5

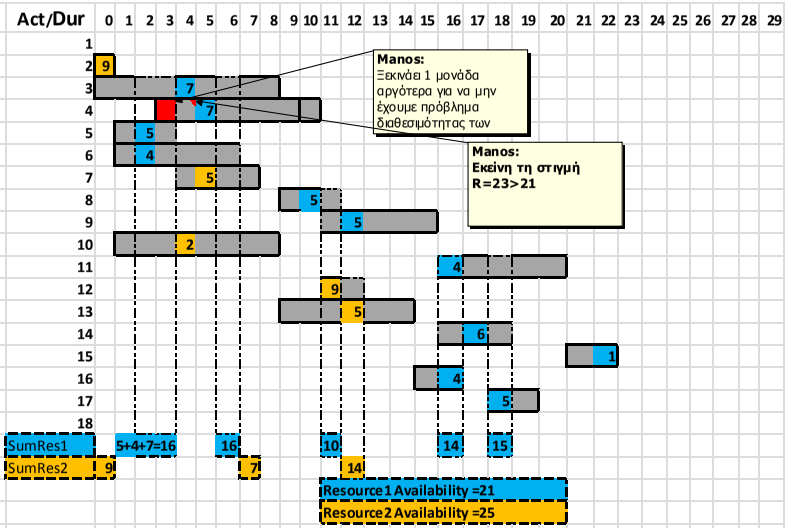
j1020_10	10	14	10,96	10	10%	25%	12	0,7	0,5	0,5
j1021_2	23	26	23,39	23	2%	0,64	12	0,7	1	0,75
j1021_9	25	26	25,13	25	1%	87%	12	0,7	1	0,75
j1022_3	23	27	24,93	23	8%	5%	12	0,7	1	0,75
j1022_9	19	24	20,43	19	8%	37%	12	0,7	1	0,75
j1023_4	21	27	22,7	21	8%	23%	12	0,7	1	0,75
j1023_5	14	20	15,6	14	11%	1%	12	0,7	1	0,75
j1024_2	14	17	14,16	14	1%	86%	12	0,7	1	0,75
j1027_10	14	17	14,23	14	2%	87%	12	1	0,5	0,5
j1028_7	15	17	15,24	15	2%	77%	12	1	0,5	0,5
j1028_8	16	18	16,18	16	1%	83%	12	1	0,5	0,5
j1029_2	19	23	19,42	19	2%	82%	12	1	1	0,75
j1029_3	24	28	25,15	24	5%	60%	12	1	1	0,75
j1030_1	16	20	17,48	16	9%	11%	12	1	1	0,75
j1030_2	17	19	18,25	17	7%	25%	12	1	1	0,75
j1032_6	17	18	17,08	17	0%	92%	12	1	1	0,75
j1032_7	16	19	16,25	16	2%	78%	12	1	1	0,75
j1034_5	22	27	22,9	22	4%	45%	12	0,2	0,5	0,75
j1034_7	26	31	26,77	26	3%	65%	12	0,2	0,5	0,75
j1035_4	23	28	24,51	23	7%	10%	12	0,2	0,5	0,75
j1035_5	23	25	24,34	24	1%	76%	12	0,2	0,5	0,75
j1036_3	17	20	17,65	17	4%	88%	12	0,2	0,5	0,75
j1036_4	22	28	23,65	22	7%	69%	12	0,2	0,5	0,75
j1040_1	19	23	21	19	11%	56%	12	0,2	1	0,75

ΠΑΡΑΡΤΗΜΑ Γ

Παράδειγμα n311_10



Activity	PSh	Predeccessors	Successors	Mode	Way	Duration	Start Time	Finish Time	PriorityValues	Eligible
1	1	-	2 3 4	1	Normal	0	0	0	10	2 3 4
2	3	1	5 6 10	1	Crashing	1	0	1	4	4 5 6 10
3	2	1	8 11 17	3	Normal	9	0	9	18	2 4
4	10	1	9 12	3	Normal	7	3	10	2	9 12
`	5	2	7 12 13	2	Normal	3	1	4	11	4 6 7
6	7	2	12 13 17	2	Crashing	6	1	7	7	4 13
7	8	5	8 9	3	Crashing	4	4	8	8	4 6 8
8	6	3 7	16	2	Crashing	3	9	12	12	4 6
9	13	4 7	11 14	2	Crashing	5	10	15	4	11 12 14
10	4	2	11 13 14	3	Normal	8	1	9	17	4 5 6
11	9	3 9 10	15	3	Crashing	5	15	20	9	12 14
12	11	4 5 6	15 16	1	Normal	2	10	12	1	15 16
13	14	5 6 10	16	3	Crashing	6	9	15	16	4
14	17	9 10	17	2	Normal	3	15	18	6	12 17
15	12	11 12	18	1	Crashing	2	20	22	13	18
16	16	8 12 13	18	2	Crashing	2	15	17	15	15
17	15	3 6 14	18	2	Crashing	2	18	20	5	12
18	18	15 16 17	-	1	Normal	0	22	22	14	-



Activity	Mode	R1	R2	Way	Crashing Duration	Duration	Normal Cost	Crashing Cost
1	1	0	0	Normal		0		
2	1	0	9	Crashing	1	1	0	63
3	3	7	0	Normal		9	63	0
4	3	7	0	Normal		7	49	0
5	2	5	0	Normal		3	15	0
6	2	4	0	Crashing	6	6	0	84
7	3	0	5	Crashing	4	4	**0**	140
8	2	5	0	Crashing	3	3	**0**	52,5
9	2	5	0	Crashing	5	5	0	87,5
10	3	0	2	Normal		8	**32**	0
11	3	4	0	Crashing	**5**	5	**0**	70
12	1	0	9	Normal		2	36	0
13	3	0	5	Crashing	**6**	6	**0**	210
14	2	6	0	Normal		3	18	0
15	1	1	0	Crashing	**2**	2	0	7
16	2	4	0	Crashing	**2**	2	0	28
17	2	5	0	Crashing	**2**	2	0	35
18	1	0	0	Normal		0	0	0
		53	30			sum	213	777
				SumCrashin	36			
Finish Time=		22						

Individual 2

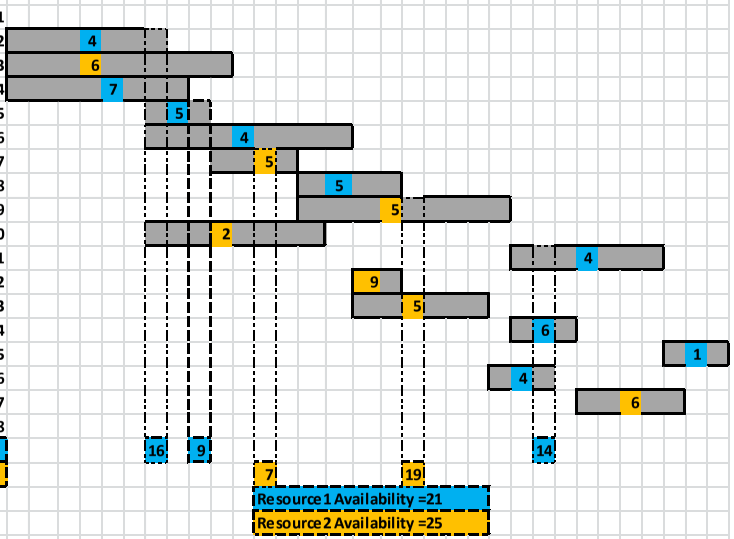
Position: Activities ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Priority List	10	4	18	2	11	7	8	12	4	17	9	1	16	6	13	15	5	14
Mode List	1	1	3	3	2	2	3	2	2	3	3	1	3	2	1	2	2	1
Way List	0	1	0	0	10	1	1	1	0	1	1	0	1	0	1	1	1	0

Normal Cost		Crashing Cost		Indirect Cost
R1	R2	R1	R2	
1	2	3,5	7	100

Normal+Crashing = Direct Cost	990
Indirect Cost = IndirectCostPerDay*FTJ	2200
Συνολικό Κόστος Fc = Direct Cost + Indirect	3190

Activity	PSh	Predeccessors	Successors	Mode	Way	Duration	Start Time	Finish Time	PriorityValues	Eligible
1	1	-	2 3 4	1	Normal	0	0	0	10	2 3 4
2	2	1	5 6 10	3	Crashing	6	0	6	18	4 5 6 10
3	6	1	8 11 17	2	Normal	9	0	9	4	2 4
4	4	1	9 12	3	Normal	7	0	7	8	9 12
5	10	2	7 12 13	2	Normal	3	6	9	1	4 6 7
6	3	2	12 13 17	2	Normal	9	6	15	9	4 13
7	5	5	8 9	3	Crashing	4	9	13	11	4 6 8
8	12	3 7	16	2	Normal	4	13	17	3	4 6
9	7	4 7	11 14	3	Normal	9	13	22	12	11 12 14
10	9	2	11 13 14	3	Normal	8	6	14	7	4 5 6
11	14	3 9 10	15	3	Normal	7	22	29	6	12 14
12	11	4 5 6	15 16	1	Normal	2	15	17	17	15 16
13	15	5 6 10	16	3	Crashing	6	15	21	2	4
14	17	9 10	17	2	Normal	3	22	25	15	12 17
15	8	11 12	18	1	Normal	3	29	32	13	18
16	13	8 12 13	18	2	Normal	3	21	24	14	15
17	16	3 6 14	18	3	Crashing	5	25	30	5	12
18	18	15 16 17	-	1	Normal	0	32	32	16	-

Act/Dur
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
SumRes1
SumRes2



Activity	Mode	R1	R2	Way	ashing Duratic	Duration	Normal Cost	Crashing Cost
1	1	0	0	Normal		0		
2	3	4	0	Crashing	6	6	0	84
3	2	0	6	Normal		9	108	0
4	3	7	0	Normal		7	49	0
5	2	5	0	Normal		3	15	0
6	2	4	0	Normal		9	36	#REF!
7	3	0	5	Crashing	4	4	**0**	140
8	2	5	0	Normal		4	**20**	#REF!
9	3	0	5	Normal		9	90	#REF!
10	3	0	2	Normal		8	**32**	0
11	3	4	0	Normal		7	**28**	#REF!
12	1	0	9	Normal		2	36	0
13	3	0	5	Crashing	**6**	6	**0**	210
14	2	6	0	Normal		3	18	0
15	1	1	0	Normal		3	3	#REF!
16	2	4	0	Normal		3	12	#REF!
17	3	0	6	Crashing	**5**	5	0	210
18	1	0	0	Normal		0	0	0
		40	38				447	#REF!
				SumCrashing	21			
Finish Time=		32						

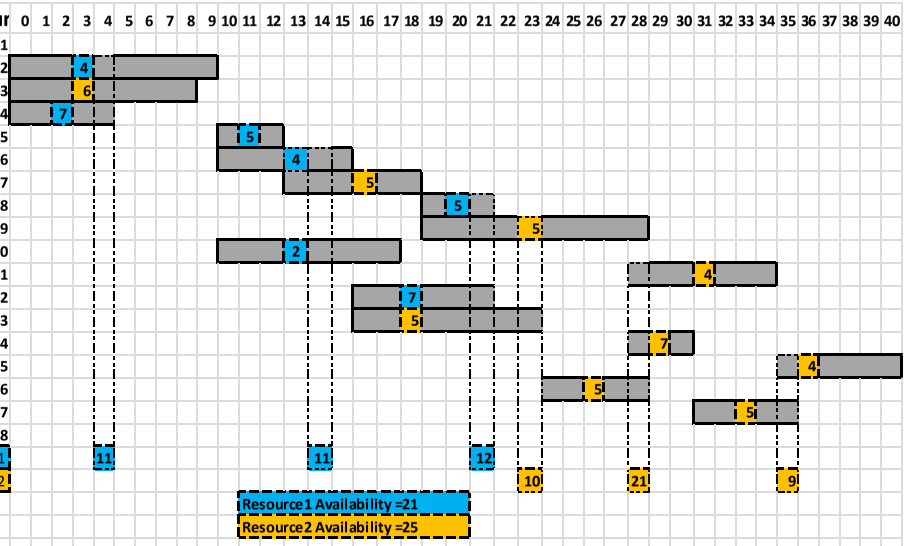
Normal Cost		Crashing Cost		Indirect Cost
R1	R2	R1	R2	
1	2	3,5	7	100

Normal+Crashing = Direct Cost	#REF!
Indirect Cost = IndirectCostPerDay*F	3200
Συνολικό Κόστος Fc = Direct Cost + In	#REF!

Individual 3

Position: Activities ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Priority List	10	18	4	8	1	9	11	3	12	7	6	17	2	15	13	14	5	16
Mode List	1	3	2	3	2	2	3	2	3	3	3	1	3	2	1	2	3	1
Way List	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	

Activity	PSh	Predeccessors	Successors	Mode	Way	Duration	Start Time	Finish Time	PriorityValues	Eligible	Act/Dur
1	1	-	2 3 4	1	Crashing	0	0	0	1	2 3 4	1
2	2	1	5 6 10	3	Normal	10	0	10	18	4 5 6 10	2
3	10	1	8 11 17	2	Normal	9	0	9	4	2 4	3
4	5	1	9 12	3	Crashing	5	0	5	11	9 12	4
5	7	2	7 12 13	2	Normal	3	10	13	14	4 6 7	5
6	4	2	12 13 17	2	Crashing	6	10	16	5	4 13	6
7	9	5	8 9	3	Normal	6	13	19	16	4 6 8	7
8	14	3 7	16	2	Crashing	3	19	22	6	4 6	8
9	6	4 7	11 14	3	Normal	10	19	29	15	11 12 14	9
10	12	2	11 13 14	3	Normal	8	10	18	17	4 5 6	10
11	3	3 9 10	15	3	Normal	7	28	35	12	12 14	11
12	11	4 5 6	15 16	3	Crashing	6	16	22	9	15 16	12
13	15	5 6 10	16	3	Crashing	6	18	24	2	4	13
14	8	9 10	17	3	Crashing	3	28	31	7	12 17	14
15	17	11 12	18	3	Crashing	6	35	41	13	18	15
16	13	8 12 13	18	3	Crashing	5	24	29	8	15	16
17	16	3 6 14	18	3	Crashing	5	31	36	3	12	17
18	18	15 16 17	-	1	Normal	0	41	41	10	-	18
											SumRes1
											SumRes2



Activity	Mode	R1	R2	Way	Crashing Duration	Duration	Normal Cost	Crashing Cost
1	1	0	0	Crashing	0	0		
2	3	4	0	Normal	0	10	40	0
3	2	0	6	Normal	0	9	108	0
4	3	7	0	Crashing	5	5	0	122,5
5	2	5	0	Normal	0	3	15	0
6	2	4	0	Crashing	6	6	0	84
7	3	0	5	Normal	0	6	**60**	0
8	2	5	0	Crashing	3	3	**0**	52,5
9	3	0	5	Normal	0	10	100	0
10	3	0	2	Normal	0	8	32	0
11	3	4	0	Normal	0	7	**28**	0
12	3	7	0	Crashing	**6**	6	0	147
13	3	0	5	Crashing	**6**	6	**0**	210
14	3	0	7	Crashing	**3**	3	0	147
15	3	0	4	Crashing	**6**	6	0	168
16	3	0	5	Crashing	**5**	5	0	175
17	3	0	6	Crashing	**5**	5	0	210
18	1	0	0	Normal	**0**	0	0	0
		36	45				383	1316
				SumCrashin	45			
Finish Time=		41						

Individual 4

Position: Activities ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Priority List	1	3	2	3	2	2	3	2	3	3	3	3	3	3	3	3	3	1
Mode List	1	18	4	11	14	5	16	6	15	17	12	9	2	7	13	8	3	10
Way List	1	0	0	1	0	1	0	1	0	0	0	1	1	1	1	1	1	0

Normal Cost		Crashing Cost		Indirect Cost
R1	R2	R1	R2	
1	2	3,5	7	100

Normal+Crashing = Direct Cost	1699
Indirect Cost = IndirectCostPerDay*FTJ	4100
Συνολικό Κόστος Fc = Direct Cost + Indirect	5799

Activity	PSh	Predeccessors	Successors	Mode	Way	Duration	Start Time	Finish Time	PriorityValues	Eligible
1	1	-	2 3 4	1	Normal	0	0	0	3	2 3 4
2	2	1	5 6 10	1	Normal	1	0	1	16	4 5 6 10
3	5	1	8 11 17	2	Normal	9	0	9	6	2 4
4	6	1	9 12	3	Crashing	5	0	5	9	9 12
5	10	2	7 12 13	2	Normal	3	1	4	18	4 6 7
6	4	2	12 13 17	2	Normal	9	1	10	15	4 13
7	12	5	8 9	3	Normal	6	4	10	4	4 6 8
8	13	3 7	16	3	Crashing	3	10	13	8	4 6
9	3	4 7	11 14	3	Normal	9	10	19	1	11 12 14
10	7	2	11 13 14	3	Crashing	5	1	6	10	4 5 6
11	8	3 9 10	15	3	Crashing	5	19	24	12	12 14
12	16	4 5 6	15 16	1	Crashing	2	10	12	14	15 16
13	9	5 6 10	16	1	Normal	5	10	15	7	4
14	14	9 10	17	2	Crashing	2	19	21	17	12 17
15	11	11 12	18	3	Normal	9	24	33	13	18
16	15	8 12 13	18	3	Crashing	5	15	20	2	15
17	17	3 6 14	18	2	Normal	3	21	24	5	12
18	18	15 16 17	-	1	Normal	0	33	33	11	-

Act/Dur	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1																																	
2	5																																
3				6																													
4			7																														
5			5																														
6					4																												
7						5																											
8								7																									
9														5																			
10			2																														
11																					4												
12										9																							
13												6																					
14																					6												
15																													4				
16																		5															
17																						5											
18																																	
SumRes1			16																														
SumRes2					13							21								9													

Resource1 Availability =21
Resource2 Availability =25

Activity	Mode	R1	R2	Way	Crashing Duration	Duration	Normal Cost	Crashing Cost
1	1	0	0	Normal	0	0		
2	1	0	9	Normal		1	18	0
3	2	0	6	Normal		9	108	
4	3	7	0	Crashing	5	5		122,5
5	2	5	0	Normal		3	15	
6	2	4	0	Normal		9	36	0
7	3	0	5	Normal		6	**60**	
8	3	0	7	Crashing	3	3		147
9	3	0	5	Normal		9	90	
10	3	0	2	Crashing	5	5	0	70
11	3	4	0	Crashing	**5**	5	**0**	70
12	1	0	9	Crashing	**2**	2		126
13	1	6	0	Normal		5	**30**	
14	2	6	0	Crashing	**2**	2		42
15	3	0	4	Normal		9	72	
16	3	0	5	Crashing	**5**	5		175
17	2	5	0	Normal		3	15	
18	1	0	0	Normal		0		
		37	52				444	752,5
				SumCrashing	27			
Finish Time=		33						

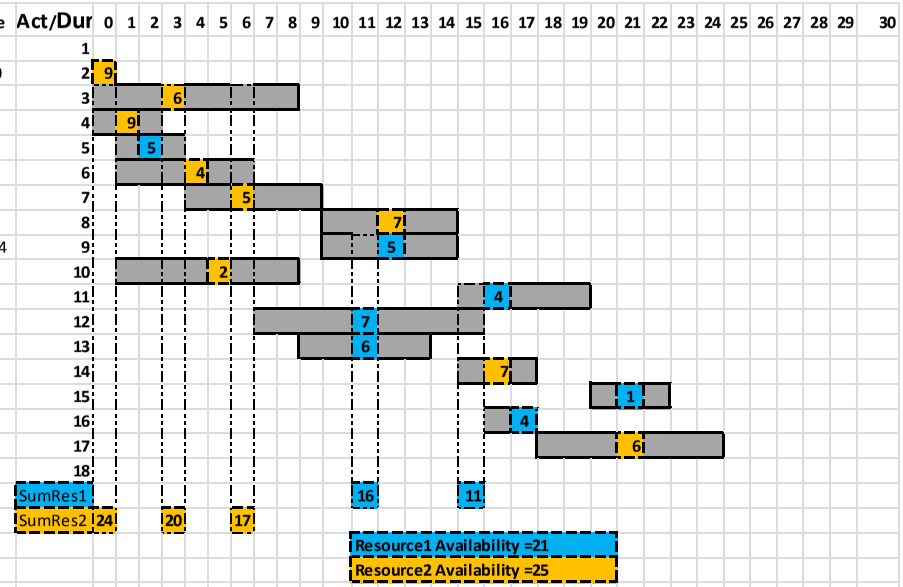
Individual 5

Position: Activities ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Priority List	3	16	6	9	18	15	4	8	1	10	12	14	7	17	13	2	5	11
Mode List	1	1	2	3	2	2	3	3	3	3	3	1	1	2	3	3	2	1
Way List	0	0	0	1	0	0	0	1	0	1	1	1	0	1	0	1	0	0

Normal Cost		Crashing Cost		Indirect Cost
R1	R2	R1	R2	
1	2	3,5	7	100

Normal+Crashing = Direct Cost	1196,5
Indirect Cost = IndirectCostPerDay*FTJ	3300
Συνολικό Κόστος Fc = Direct Cost + Indirect (	4496,5

Activity	PSh	Predeccessors	Successors	Mode	Way	Duration	Start Time	Finish Time	PriorityValues	Eligible
1	1	-	2 3 4	1	Normal	0	0	0	6	2 3 4
2	3	1	5 6 10	1	Normal	1	0	1	1	4 5 6 10
3	4	1	8 11 17	2	Normal	9	0	9	12	2 4
4	2	1	9 12	1	Normal	3	0	3	3	9 12
5	6	2	7 12 13	2	Normal	3	1	4	4	4 6 7
6	10	2	12 13 17	3	Crashing	6	1	7	10	4 13
7	5	5	8 9	3	Normal	6	4	10	15	4 6 8
8	13	3 7	16	3	Normal	5	10	15	14	4 6
9	7	4 7	11 14	2	Crashing	5	10	15	8	11 12 14
10	8	2	11 13 14	3	Normal	8	1	9	9	4 5 6
11	12	3 9 10	15	3	Crashing	5	15	20	7	12 14
12	9	4 5 6	15 16	3	Normal	9	7	16	13	15 16
13	14	5 6 10	16	1	Normal	5	9	14	18	4
14	17	9 10	17	3	Crashing	3	15	18	11	12 17
15	11	11 12	18	1	Normal	3	20	23	2	18
16	16	8 12 13	18	2	Crashing	2	16	18	5	15
17	15	3 6 14	18	3	Normal	7	18	25	17	12
18	18	15 16 17	-	1	Normal	0	25	25	16	-



Activity	Mode	R1	R2	Way	Crashing Duration	Duration	Normal Cost	Crashing Cost
1	1	0	0	Normal	0	0	**0**	**0**
2	1	0	9	Normal	0	1	18	0
3	2	0	6	Normal	0	9	108	0
4	1	0	9	Normal	0	3	54	0
5	2	5	0	Normal	0	3	15	0
6	3	0	4	Crashing	6	6	0	168
7	3	0	5	Normal	0	6	**60**	0
8	3	0	7	Normal	0	5	**70**	0
9	2	5	0	Crashing	5	5	87,5	87,5
10	3	0	2	Normal	0	8	32	0
11	3	4	0	Crashing	5	5	**0**	70
12	3	7	0	Normal	0	9	63	0
13	1	6	0	Normal	0	5	30	0
14	3	0	7	Crashing	3	3	0	147
15	1	1	0	Normal	0	3	3	0
16	2	4	0	Crashing	2	2	0	28
17	3	0	6	Normal	**0**	7	84	0
18	1	0	0	Normal	**0**	0	0	0
		32	55				624,5	500,5
				SumCrashin	21			

Finish Time= 25

Individual 6

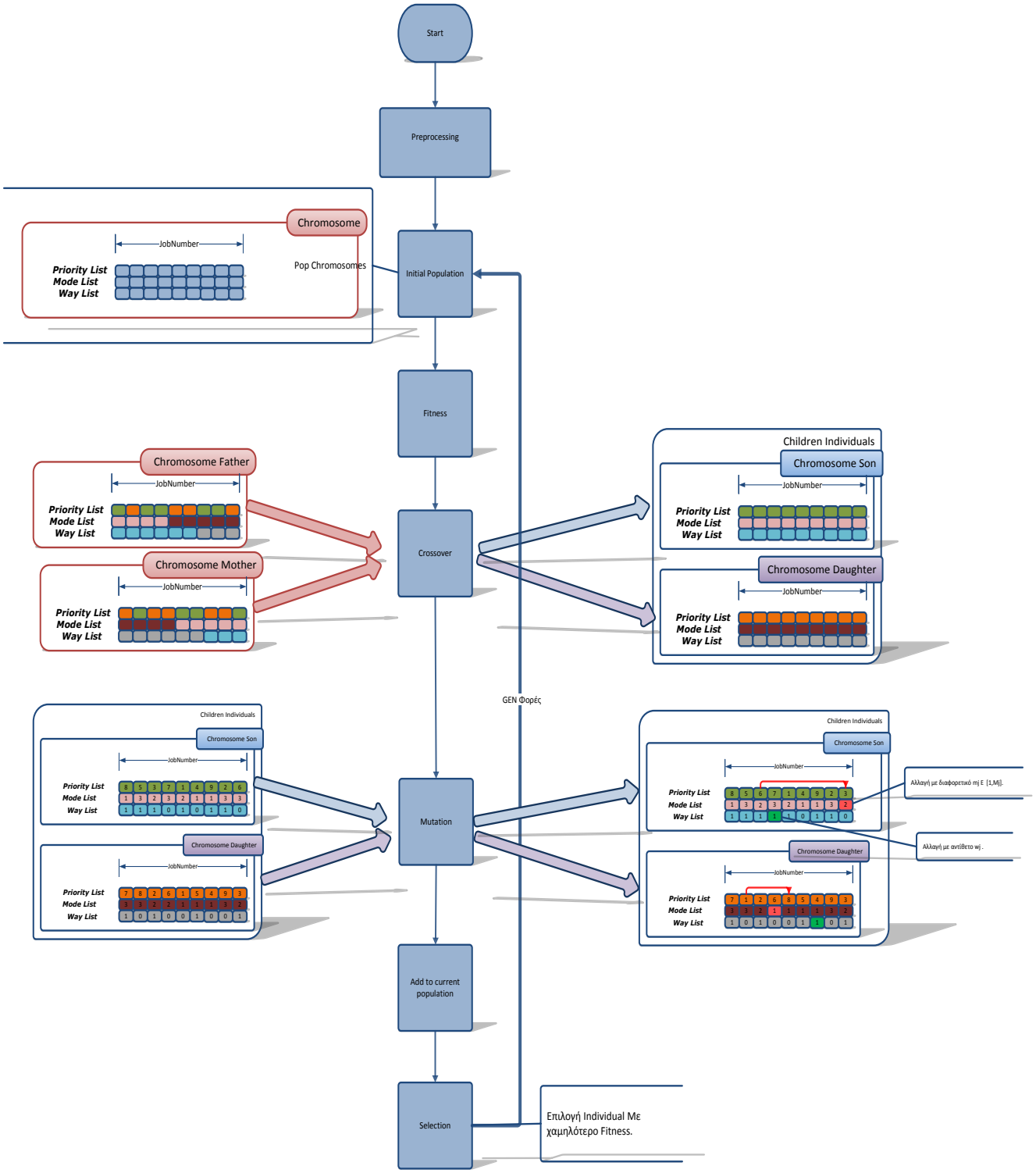
Position: Activities ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Priority List	6	1	12	3	4	10	15	14	8	9	7	13	18	11	2	5	17	16
Mode List	1	1	2	1	2	3	3	3	2	3	3	3	1	3	1	2	3	1
Way List	0	0	0	0	0	1	0	0	1	0	1	0	0	1	0	1	0	0

Normal Cost		Crashing Cost		Indirect Cost
R1	R2	R1	R2	
1	2	3,5	7	100

Normal+Crashing = Direct Cost	1125
Indirect Cost = IndirectCostPerDay*FTJ	2500
Συνολικό Κόστος Fc = Direct Cost + Indir	3625

ΠΑΡΑΡΤΗΜΑ Δ

Γενετικός Αλγόριθμος με όλα τα στοιχεία του



1

ΠΑΡΑΡΤΗΜΑ Ε

Μέρη από τον κώδικα.

#region Multi Mode Serial Generation Scheme

```
  #region Μηδενισμοί πινάκων
for (int i = 0; i < JobNumber; i++)
  {
      PSh[i] = 0;
      StartTime[i] = 0;
      EarlyStart[i] = 0;
      FinishTime[i] = 0;
      FinalDuration[i] = 0;
      At[i, 0] = 0;
      At[i, 1] = 0;
      for (int j = 0; j < ResourceType; j++)
      {
          ResourcesUsed[i, j] = 0;
          SumResources[j] = 0;
          Prosthesh[i, j] = 0;
      }
  }
  #endregion

  for (int h = 0; h < JobNumber; h++) // to megalo loop tha kathe stadio
  {
      int count2 = 0;
      int count4 = 0;

      for (int i = 0; i < JobNumber; i++)
      {
          Eh[i] = 0;
          Eh1[i] = 0;
      }
      // Ο πίνακας Eh1 μας βοηθάει σε κάθε loop του h να βρούμε ποιες
δραστηριότητες δεν έχουνε προγραμματιστεί.
      // Ο πίνακας Eh βρίσκει τις δραστηριότητες που ανήκουν στον Eh1 και
έχουνε προγραμματιστεί οι Predecessors τους . Δηλαδή οι
      // Predecessors τους ανήκουν στον πίνακα PSh.

      for (int i = 0; i < JobNumber; i++) // Σαρώνει κάθε φορά τον πίνακα
JobId.
      {
          bool program = false; // Αρχικοποιεί τη μεταβλητή program ως false.

          for (int j = 0; j < JobNumber; j++) //  Σαρώνει κάθε φορά τον πίνακα
PSh
          {
              if (JobId[i] == PSh[j]) // Για κάθε μια δραστηριότητα ελέγχει όλο
τον πίνακα PSh κάθε φορά αν έχει προγραμματιστεί.
              {
                  program = true; // Εάν έχει προγραμματιστεί η δραστηριότητα
JobId[i], η μεταβλητή program γίνεται True.
              }
          }
          if (program == false) // Εάν είναι False σημαίναι πως δεν έχει
προγραμματιστεί.
          {
```

```
                Eh1[count2] = JobId[i];  // Προστίθετε στον πίνακα Eh1[] η μη
προγραμματισμένη δραστηριότητα.
                count2 = count2 + 1; // Αυξάνουμε κάθε φορά τον δείκτη του πίνακα
Eh1 κατά 1.
            }
        }
        for (int i = 0; i < count2; i++)// Σαρώνει τον πίνακα NumberPredecessor
με τη βοήθεια του πίνακα Eh1.
        {
            int count3 = 0;
            for (int j = 0; j < NumberPredecessor[Eh1[i] - 1]; j++)  // Σαρώνει
μέχρι τον αριθμό των Predecessor κάθε δραστηριότητας.
            {
                for (int z = 0; z < h; z++)
                {
                    if (IdPredessesor[Eh1[i] - 1, j] == PSh[z])  //  Βρίσκει εάν
οι Predecessor κάθε δραστηριότητας είναι προγραμματισμένες.
                    {
                        count3 = count3 + 1;
                    }
                }
            }
            if (count3 == NumberPredecessor[Eh1[i] - 1]) // Εάν όλες οι
Predecessor είναι προγραμμάτισμένες, τοποθετεί την δραστηριότητα στον πίνακα
Eh.
            {
                Eh[count4] = Eh1[i];
                count4 = count4 + 1;
            }
        }
        int highPriority = priority(Eh, JobNumber, h, p, PositionValuesSon); //
Καλεί τη συνάρτηση Priority για να βρει την επόμενη δραστηριότητα για
προγραμματισμό.

        /* Στην επόμενη ενέργεια διαλέγει μεταξύ Crashing και Normal Duration  */
        /* Στην επόμενη ενέργεια διαλέγει mode για την δραστηριοτητα που
επιλέχτηκε να προγραμματιστεί από τη πάνω συνάρτηση  */

        PSh[h] = highPriority; //Βάζει στον πίνακα partial scheduled τη
δραστηριότητα που επέστρεψε η συνάρτηση με το Highest Priority.
        // Δέχεται ως δεδομένα την διάρκεια και τον mode που εκτελείτε η
δραστηριότητα.

        int xa = Way(CrashingDuration, Duration, highPriority, NumberOfModes,
SelectedWaysSon, p); // xa=1 crashing , xa=0 normal dialegei diarkeia
        int ya = Mode(highPriority, NumberOfModes, p, SelectedModesSon,
ModeNumber); // dialegei mode

        // Ανάλογα με τα xa, ya που πήρε από τις πάνω συναρτήσεις φτιάχνει τον
πίνακα με τις διάρκειες που θα έχουν οι δραστηριότητες.

        #region Φτιάχνει τον πίνακα FinalDuration
        if (xa == 1)
        {
            FinalDuration[h] = CrashingDuration[highPriority - 1, ya - 1]; //
prosoxh to ya. susxetish me pinaka diarkeias
        }
        else
        {
            FinalDuration[h] = Duration[highPriority - 1, ya - 1];
        }
        #endregion
```

```
      if (h == 0)
      {
          EarlyStart[h] = 0;
          StartTime[h] = 0;
          FinishTime[h] = FinalDuration[h];
          At[h, 0] = FinishTime[h] - FinalDuration[h] + 1;
          At[h, 1] = FinishTime[h];

          for (int i = 0; i < ResourceType; i++)  // h dummy den exei
resourcesneed ara oute xrhsimopoiei resources.
          {
              ResourcesUsed[h, i] = 0;
          }
      }
      else
      {
          int max4 = 0; // to megisto finish time ton predecessors ths
drasthriothtas HighPriority
          for (int i = 0; i < NumberPredecessor[highPriority - 1]; i++)
          {
              for (int j = 0; j < h; j++)
              {
                  if (IdPredessesor[highPriority - 1, i] == PSh[j])
                  {
                      if (FinishTime[j] > max4)
                      {
                          max4 = FinishTime[j];
                      }
                  }
              }
          }

          EarlyStart[h] = max4; // Υπολογίζει το EarlyStart ως τη μέγιστη
διάρκεια ολοκλήρωσης των Predecessor .

          // FinishTime-d+1<= At <= FinishTime

          StartTime[h] = EarlyStart[h];
          FinishTime[h] = StartTime[h] + FinalDuration[h];

      b: At[h, 0] = FinishTime[h] - FinalDuration[h];
          At[h, 1] = FinishTime[h];

          for (int i = 0; i < ResourceType; i++)
          {
              ResourcesUsed[h, i] = ResourcesNeed[highPriority - 1, ya - 1, i];
              SumResources[i] = SumResources[i] + ResourcesUsed[h, i];
          }
          // Αν το συγκεκριμένο Individual υπερβαίνει τη διαθεσιμότητα των μη-
ανανεώσιμων το ακυρώνει. Του δίνει μεγάλο Fitness.

          for (int n = NumberOfRenewable; n < ResourceType; n++)
          {
              if (SumResources[n] > ResourceAvailability[n])
              {
                  for (int j = 0; j < JobNumber; j++)
                  {
                  N: SelectedModesSon[p, j] = ModeNumber[j,
rand1.Next(ModeNumberColumns)];
                      if (SelectedModesSon[p, j] == 0)
                      {
```

```
                        goto N;
                    }
                }
                goto bba;
            }
        }
        /* Εάν μία δραστηριότητα βρίσκεται στο χρονικό διάστημα που
εκτελείται μία άλλη δραστηριότητα , τότε πρέπει να ελεγθεί ότι
         *οι πόροι που καταλαμβάνουν οι δραστηριότητες αυτές σε κάθε χρονικό
διάστημα δεν υπερβαίνει το ResourceAvailability κάθε ResourceType.
        * Εάν το υπερβένει τότε στην προς προγραμματισμό δραστηριότητα ,
αυξάνουμε το StartTime (άρα και το FinishTime αφού είναι αλληλένδετα) ,
         *μέχρι να μην υπερβαίνεται το όριο των πόρων σε κάθε περίοδο.*/

        for (int j = 0; j < ResourceType; j++)
        {
            Prosthesh[h, j] = ResourcesUsed[h, j];
        }
        for (int i = 0; i < h; i++)
        {
            if (((At[h, 0] <= At[i, 0]) & (At[h, 1] >= At[i, 0])) || ((At[h,
0] >= At[i, 0]) & (At[h, 0] < At[i, 1])) || ((At[h, 0] >= At[i, 0]) & (At[h, 1]
<= At[i, 1])) || ((At[h, 0] <= At[i, 0]) & (At[h, 1] >= At[i, 1])))
            {
                for (int j = 0; j < NumberOfRenewable; j++)
                {
                    Prosthesh[h, j] = Prosthesh[h, j] + ResourcesUsed[i, j];

                    if (Prosthesh[h, j] > ResourceAvailability[j])
                    {
                        StartTime[h] = StartTime[h] + 1;
                        FinishTime[h] = StartTime[h] + FinalDuration[h];
                        for (int s = 0; s < ResourceType; s++)
                        {
      SumResources[s] = SumResources[s] - ResourcesUsed[h, s];
                        }
                        goto b;
                    }
                }
            }
        }
        FinishTime[h] = StartTime[h] + FinalDuration[h]; // ypologizei to
finish time
    }
    if (xa == 1)  // crashing way
    {
        for (int i = 0; i < NumberOfRenewable; i++)
        {
            // DirectCost είναι το CrashingCost Ck
            FitnessFunctionSon[p] = FitnessFunctionSon[p] +
(CrashingDuration[highPriority - 1, ya - 1] * DirectCost[i] * ResourcesUsed[h,
i]);
        }
    }
    if (xa == 0)  // normal way
    {
        for (int i = 0; i < NumberOfRenewable; i++)
        {
            // IndirectCost είναι το Normal Cost nk
            FitnessFunctionSon[p] = FitnessFunctionSon[p] +
(Duration[highPriority - 1, ya - 1] * IndirectCost[i] * ResourcesUsed[h, i]);
        }
```

```
      }
  }                   // Τέλος του Loop για κάθε στάδιο.

  #endregion // Multi Mode Serial Generation Scheme


public static int priority(int []Eh ,int JobNumber,int h , int p , int
[,]PositionValues)
        {
            int max = 0;
            int max2 = 0;

                if (h == 0)
                {
                    max2 = Eh[0]; // dhladh 1 afou tha einai h dummy panta.
                }
                else
                {
                    for (int i = 0; i < JobNumber; i++)
                    {
                       if (Eh[i]!=0)
                       {
                        if ( PositionValues[p,Eh[i]-1]>max  )
                        {
                            max = PositionValues[p, Eh[i] - 1];
                            max2 = Eh[i];
                        }
                       }
                    }
            }
            int highPriority = max2;
            return highPriority;
        }



Selection
int minIndex;
for (int j = 0; j < pop; j++)
{
   minIndex = Array.IndexOf(FitnessFunctionFinishTime2Pop,
   FitnessFunctionFinishTime2Pop.Min());
   FitnessFunctionFinishTime[j] = FitnessFunctionFinishTime2Pop[minIndex];
   FitnessFunction[j] = FitnessFunction2Pop[minIndex];
   FitnessFunctionFinishTime2Pop[minIndex] = 999999999999999999;
      for (int m = 0; m < JobNumber; m++)
     {
         PositionValues[j, m] = PositionValues2Pop[minIndex, m];
         SelectedModes[j, m] = SelectedModes2Pop[minIndex, m];
         SelectedWays[j, m] = SelectedWays2Pop[minIndex, m];
         PrecedenceList[j, m] = PrecedenceList2Pop[minIndex, m];
         StartTimeList[j, m] = StartTimeList2Pop[minIndex, m];
         FinalDurationList[j, m] = FinalDurationList2Pop[minIndex, m];
     }
}
```

```
#region Crossover των SelectedModes για τα Children (Son-Daughter)
    //one point crossover
    Random rand5 = new Random();

    for (int i = 0; i < pop / 2; i++)
    {
        int random111 = rand5.Next(0, JobNumber);

        for (int j = 0; j < random111; j++)
        {
            SelectedModesSon[i, j] = SelectedModes[Pairs[i, 0] - 1, j];
            SelectedModesDaughter[i, j] = SelectedModes[Pairs[i, 1] - 1, j];
        }
        for (int j = random111; j < JobNumber; j++)
        {
            SelectedModesSon[i, j] = SelectedModes[Pairs[i, 1] - 1, j];
            SelectedModesDaughter[i, j] = SelectedModes[Pairs[i, 0] - 1, j];
        }
    }
    #endregion


  #region PositionValues Mutation

   int change1 = 0;                          // gia ta Sons
   int change2 = 0;

   Random rand12 = new Random();
   Random rand13 = new Random();
   int random6 = rand12.Next(0, JobNumber);
        c2: int random7 = rand13.Next(0, JobNumber);
   if (random6 == random7)
   {
       goto c2;
   }

   change1 = PositionValuesSon[i, random6];
   change2 = PositionValuesSon[i, random7];

   PositionValuesSon[i, random6] = change2;
   PositionValuesSon[i, random7] = change1;


   int change3 = 0;                          // gia ta Daughters
   int change4 = 0;

   Random rand14 = new Random();
   Random rand15 = new Random();
   int random8 = rand14.Next(0, JobNumber);
        c3: int random9 = rand15.Next(0, JobNumber);
   if (random6 == random7)
   {
       goto c3;
   }

   change3 = PositionValuesDaughter[i, random8];
   change4 = PositionValuesDaughter[i, random9];

   PositionValuesDaughter[i, random8] = change4;
   PositionValuesDaughter[i, random9] = change3;
   #endregion
```

```
#region Preprocessing
                    // Βγάζει εκτός τους Inefficient Modes
 // Στους inefficient το περιεχόμενο τους μετατρέπεται σε 0, όπως επίσης και το
ResourceNeed τους.

 for (int i = 0; i < JobNumber; i++)
 {
     for (int j = 0; j < NumberOfModes[i] ; j++)
         {
             for (int k = 0; k < NumberOfRenewable; k++)
             {
                 if (ResourcesNeed[i, j, k] > ResourceAvailability[k])
                 {
                     ModeNumber[i, j] = 0;
                     ResourcesNeed[i, j, k] = 0;
                 }
             }
         }
 }
for (int i=0;i<JobNumber;i++)
 {
     for (int k = 0; k < NumberOfRenewable; k++)
     {
         if (NumberOfModes[i] > 1)//inefficient
         {
             for (int j = 0; j < NumberOfModes[i]-1; j++)
             {
                 for (int m = j + 1; m < NumberOfModes[i]; m++)
                 {
  if ((ResourcesNeed[i, j, k] != 0) & (ResourcesNeed[i, m, k] != 0))
  {
      if (Duration[i, j] >= Duration[i, m] &
ResourcesNeed[i,j,k]>=ResourcesNeed[i,m,k])
      {
          ModeNumber[i, j] = 0;
          ResourcesNeed[i, j, k] = 0;
      }
      else if (Duration[i, j] <= Duration[i, m] & ResourcesNeed[i, j, k] <=
ResourcesNeed[i, m, k])
      {
          ModeNumber[i, m] = 0;
          ResourcesNeed[i, m, k] = 0;
      }
  }
                 }
             }
         }

     }
 }
```