

NATIONAL TECHNICAL UNIVERSITY OF ATHENS  
SCHOOL OF MECHANICAL ENGINEERING  
SECTOR OF INDUSTRIAL MANAGEMENT AND OPERATIONAL RESEARCH



Doctoral Thesis

---

Decision Making in Project Management:  
Multi Objective Extended Resource Constrained Project  
Scheduling

---

Eleni I. Rokou

*BSc Computer Science Univ. of Ioannina, MSc Financial and Management Engineering Univ. Aegean*

**Supervisor:**

Assist. Prof. K. Kirytopoulos

**Supervisory Committee**

Prof. I. Tatsiopoulos

Assoc. Prof. V. Leopoulos

Assist. Prof. K. Kirytopoulos

Athens, 2013



*To the one who taught me how to learn,  
grazie mamma.*



## Acknowledgements

I would like to thank first and foremost my supervisor, professor K. Kirytopoulos for his patience, generous guidance, advice and the long hours spent on discussing each and every idea here in presented. I am especially grateful to my supervisor professor and the members of the supervisory committee professors I. Tatsiopoulos and V. Leopoulos for entrusting me with this endeavour. I would also like to thank professor D. Drivaliaris for his insightful comments and support on the validation of the mathematical model.

I am obliged to professor N.Marmaras and my colleagues V. Tsagkas and X. Vassilakopoulou for helping me see the world on a less deterministic way. I would also like to thank the administrative staff of the Sector of Industrial Management and Operational Research, especially Vicky Koulara, whose support and encouragement was vital for the success of the project.

It is a pleasure to thank the undergraduate students being supervised by professor K. Kirytopoulos and myself for the challenging discussions and their support during these years.

Finally, I would like to thank my father, my husband and the rest of the family and friends for their patience and support.



## Abstract

A holistic approach is proposed for defining the resource constrained project scheduling problem (RCPSp). The doctoral thesis' aim is to give a formulation of the project scheduling problem where all deterministic aspects that have been previously explored in the relevant literature are covered. Our goal is to provide a way to model and solve project scheduling problems as they actually are, without compromises other than the assumption that the given inputs are realistic. An appropriate mathematical formulation along with a concise solution process, covering both the single and multi-objective case, are presented. Based on this model an adaptive evolutionary algorithm is implemented to solve the unified version of the problem along with and Add in for MS Project to provide an easy to use interface to the project managers. The efficiency of the proposed approach is compared to existing implementations through a number of experiments. The experiments are grouped in two classes: in the first one the best known results from each variation and extension of the single objective RCPSp are compared to the results given by the proposed algorithm and in the second one the multi-objective approach is compared to the single-objective results given in the same test cases appropriately adapted. Finally, the application of the proposed approach in real situations is illustrated through a case study on a medium sized project (200 activities) taken from the GIS domain.

The results show that the usage of the holistic model doesn't affect the quality of results or the needed CPU-time when compared to the existing RCPSp formulations, whereas it adds the ability to describe more realistically any complex project scheduling problem. We overcome the raise of complexity and the infeasibilities by using penalty functions when relaxation of the constraints is needed. In the multi-objective case the algorithm is capable of providing multiple solution scenarios that are generated either based on the simple Pareto front or on a weighted approximation of it.





## **Declaration**

I Elena Rokou, hereby declare that to the best of my knowledge and belief, this PhD Thesis is my own work and all sources or work of other people have been properly acknowledged. The dissertation contains no plagiarism nor material that has already been used to any substantial extent for a comparable purpose.



# Contents

0.1	Εισαγωγή	xxvii
0.2	Δομή Διατριβής	xxxii
0.3	Χρονοδιάγραμμα Εκπόνησης της Διατριβής	xxxiv
0.4	Ορισμός Προβλήματος	xxxvi
0.5	Προτεινόμενη Μαθηματική Μοντελοποίηση	xxxix
0.5.1	Ορισμοί	xxxix
0.5.2	Αντικειμενικοί στόχοι	xlv
0.5.3	Περιορισμοί	xlvii
0.6	Μαθηματική Μοντελοποίηση	xlviii
0.7	Προτεινόμενη Μέθοδος Επίλυσης	1
0.8	Πειραματική Επαλήθευση Αποτελεσμάτων	Iv
0.9	Μελέτη Περίπτωσης	λυι
0.10	Συμπεράσματα - Σημεία Καινοτομίας	λυι
<b>1</b>	<b>Overview</b>	1
1.1	Introduction	1
1.2	Motivation	2
1.3	Research Objectives	4
1.4	Research Boundaries	4
1.5	Overall structure and contents of this Thesis	6
<b>2</b>	<b>Literature Review</b>	7
2.1	Project Management	7
2.2	Project Scheduling	10
2.3	Classification of Project Scheduling problems	11
2.4	The Resource Constrained Project Scheduling Problem	12
2.4.1	Problem Definition	15
2.4.2	Variations and Extensions	17
2.4.3	Complexity	28
2.4.4	Solution Methods	29
2.5	Multi-Criteria Decision Making	40
2.5.1	Multi-Objective Decision Making	41
2.5.2	Multi-Attribute Decision Making	44

<b>3</b>	<b>Research Method</b>	53
3.1	General Research Methods	53
3.2	Mathematical Modelling	53
3.3	Algorithm Design	55
3.4	Research Process	56
<b>4</b>	<b>Problem Definition</b>	59
4.1	Systems approach to project scheduling	59
4.1.1	Problem Structuring	60
4.1.2	System Dynamics view of project scheduling	62
4.2	Problem Description	65
4.2.1	Goals	65
4.2.2	Available Inputs and Constraints	68
<b>5</b>	<b>Proposed Holistic Mathematical Model</b>	71
5.1	Proposed Problem Formulation	71
5.1.1	Definitions	71
5.1.2	Objectives	76
5.1.3	Constraints	78
5.2	Mathematical Formulation	79
<b>6</b>	<b>Solution Process</b>	81
6.1	Overview	81
6.2	Decision Making using the Analytic Network Process	84
6.3	Proposed Solution Algorithm	93
6.3.1	Basic Scheme	93
6.3.2	Preprocessing	96
6.3.3	Proposed Schedule Generation Schemes for the extended RCPSP	98
6.3.4	Chromosomes	103
6.3.5	Initial Population	104
6.3.6	Operators	105
6.3.7	Selection Strategy	107
6.3.8	Multi-Objective Optimisation Process	107
6.3.9	Auxiliary Solution Algorithms	111
<b>7</b>	<b>Computational Results and Evaluation</b>	119
7.1	Implementation	119
7.2	Experiments design	121
7.3	Experimental Comparison to best in class algorithms	121
7.4	Experimental results for multi-objective optimisation	126
<b>8</b>	<b>Case Study</b>	129
8.1	Introduction	129
8.2	Initial data	129
8.2.1	Execution modes	130
8.2.2	Preemption	132
8.2.3	Variability of resource availabilities and requirements	133
8.2.4	Constraints	133
8.2.5	Objectives	134
8.3	Solution scenarios	135

<b>9</b>	<b>General Discussion &amp; Conclusions</b> .....	137
9.1	Summary of PhD Thesis contribution .....	137
9.2	Potential Impact and Significance .....	138
9.2.1	Implications for researchers .....	138
9.2.2	Implications for practitioners .....	139
9.3	Future Work .....	140
	<b>References</b> .....	143
<b>A</b>	<b>Experimental Results</b> .....	151
A.1	Exerpt of analytical results RCPSP .....	151
A.2	Exerpt of analytical results MRCPSP .....	158
A.3	Exerpt of analytical results MRCPSP/max .....	159
A.4	Case Study .....	160
A.4.1	Activities .....	160
A.4.2	Gantt chart .....	165
<b>B</b>	<b>Implemented Code – Main Modules</b> .....	175
<b>C</b>	<b>Ms Project 2013 – Add In for Multi-Objective Resource Constrained Project Scheduling</b> .....	197
	<b>Glossary</b> .....	203



## List of Figures

0.1	Σχηματική αναπαράσταση προς επίλυση προβλήματος	xxviii
0.2	Χρονοπρογραμματισμός Εκπόνησης Διατριβής	xxxv
0.3	(α) Δραστηριότητες, τρόποι εκτέλεσης, απαιτήσεις σε πόρους και σημεία διακοπής των δραστηριοτήτων του έργου, (β) <i>minimal</i> και <i>maximal lag</i> των δραστηριοτήτων ανά τρόπο εκτέλεσης	xliii
0.4	(α) διαθεσιμότητα πόρου P1 σε σχέση με το χρόνο, (β) διαθεσιμότητα πόρου P1 σε σχέση με το χρόνο και (γ) διαθεσιμότητα μη ανανεώσιμου πόρου NP1 σε σχέση με τις περιόδους $I_{10} = [0, t_{a1})$ , $I_{11} = [t_{a1}, t_{a2})$ και $I_{12} = [t_{a2}, T)$	xliiii
0.5	(α) Γράφος του έργου για επιλογή τρόπων εκτέλεσης $M_1(0, 0, 0, 0, 0, 0, 0, 0)$ και (β) $M_2(0, 0, 2, 1, 1, 2, 1, 0)$	xliv
0.6	Φάσεις προτεινόμενης μεθόδου επίλυσης	1
0.7	Ροή δεδομένων	lii
0.8	Βελτιστοποίηση πολλαπλών αντικειμενικών στόχων με ANP	liiii
0.9	<i>Pareto</i> ΓΑ	liv
0.10	<i>EMO – RCPSP – MSPProject</i>	lv
2.1	Project scheduling process	10
2.2	Typology of Scheduling problems	12
2.3	RCPSP: (a) Activity on Node representation of the project's network, (b) duration and resource requirements of each activity and (c) resulting schedule when the resource's availability is five units	14
2.4	P-RCPSP: (a) Activity on Node representation of the project's network, (b) duration and resource requirements of each activity and (c) resulting schedule when the resource's availability is five units	18
2.5	MRCPS: (a) Activity on Node representation of the project's network, (b) duration and resource requirements of each activity, (c) resulting schedule when the resource's availability is five units and mode assignment $\{0, 0, 0, 0, 0, 0, 0, 0\}$ and (d) resulting schedule when the resource's availability is five units and mode assignment $\{0, 0, 1, 1, 2, 0, 1, 0, 0\}$	20
2.6	Example digraph with time lags	23
2.7	RCPSP/max: (a) Representation of the project's network extended for GPRs using the $G(V, E)$ digraph, (b) duration and resource requirements of each activity	24
2.8	Classification of Schedules	33
2.9	Multi-objective optimisation procedure	41
3.1	Mathematical modelling process and its validation	54

3.2	Research Process	57
4.1	Project Scheduling Rich Picture	62
4.2	Project monitoring cycle	64
5.1	(a) Project activities, modes, resource requirements and preemption status, (b) minimal and maximal lag of activities per mode	75
5.2	(a) R1 renewable resource availability in relation to time, (b) R2 renewable resource availability in relation to time and (c) NR1 non-renewable resource availability in relation to periods $I_{l0} = [0, t_{a1})$ , $I_{l1} = [t_{a1}, t_{a2})$ and $I_{l2} = [t_{a2}, T)$	75
5.3	(a) Project network for mode set $M_1(0, 0, 0, 0, 0, 0, 0, 0)$ and (b) Project network for mode set $M_2(0, 0, 2, 1, 1, 2, 1, 0)$	76
6.1	Phases of the proposed approach	82
6.2	Flow of events	83
6.3	ANP model	87
6.4	Solution process moderator basic flow of events	94
6.5	Usage of auxiliary algorithms	94
6.6	(a) Example of graph with generalised precedence constraints, (b) initial distance matrix and (c) final distance matrix	100
6.7	GA chromosome	104
6.8	Example of initial population	105
6.9	Crossover operator	106
6.10	Mutation operator	106
6.11	Pareto GA	108
6.12	ANP weighted multi-objective optimisation	111
7.1	EMO-RCPSP: a Ms Project Add-In	120
7.2	Single objective execution of j30 instances	123
7.3	Single objective execution of MRCPSP instances	124
7.4	Single objective execution of MRCPSP/max instances	125
8.1	Initial data of the GIS project	130
8.2	Overallocated resources	131
8.3	Defining multiple execution modes	132
8.4	Defining task preemption	132
8.5	Defining task resource requirements per step	133
8.6	Weighting the objectives using the ANP Solver	135
8.7	Best schedule found by pareto optimal approach - 598 days	136
A.1	Single objective execution of j30 instances - part I	152
A.2	Single objective execution of j30 instances - part II	153
A.3	Single objective execution of j30 instances - part III	154
A.4	Single objective execution of j30 instances - part IV	155
A.5	Single objective execution of j30 instances - part V	156
A.6	Single objective execution of MRCPSP instances	158
A.7	Single objective execution of MRCPSP instances	159
C.1	Ribbon styled toolbox for EMO- RCPSP	197
C.2	Import text files formatted as in PSPLib to run experiments	197



C.3	Set up additional properties to handle multiple modes of execution, non renewable resource type and maximal time lags .....	198
C.4	Define one or more optimisation objectives .....	198
C.5	Select execution algorithm to generate solutions - schedules .....	198
C.6	Set up parameters for the genetic algorithm (auxiliary algorithm).....	199
C.7	Set up parameters for the simulated annealing algorithm (auxiliary algorithm).....	199
C.8	The given scheduled as was imported from <i>J301<sub>1</sub>.rcp</i> file from PSPLib .....	200
C.9	Schedule generated by the proposed algorithm - note that the "As Soon As Possible" constraint has been changed to "Must Start On".....	201
C.10	Resource availabilities - the overallocated resources are marked with red .....	201
C.11	Duration of proposed scheduled compared to the results of MS Project's levelling option .....	202



## List of Tables

0.1	Μαθηματικοί Συμβολισμοί . . . . .	xlii
0.2	Συγκριτικά αποτελέσματα για ελαχιστοποίηση διάρκειας έργου . . . . .	lvi
0.3	δμπαρατιε ρεσυλτς φορ μυλτι-οβθεςτιε ινστανζες . . . . .	lvii
2.1	Job-shop sequencing and assembly-line balancing problem compared to project scheduling as referred by Icmeli (1996) . . . . .	13
2.2	Priority Rules based on Kolisch and Hartmann (1999) . . . . .	36
5.1	Basic Notation . . . . .	74
7.1	Comparative results for single-objective instance . . . . .	126
7.2	Comparative results for multi-objective instances . . . . .	127
8.1	Comparative results for GIS project . . . . .	136



## Acronyms

ANP	Analytic Network Process
CPM	Critical Path Method
EMO-RCPSP	Extended Multi Objective Resource Constrained Project Scheduling
GA	Genetic Algorithm
MRCPSP	Multi mode Resource Constrained Project Scheduling Problem
OBS	Organizational Breakdown Structure
p-SGS	Parallel Schedule Generation Scheme
PERT	Program Evaluation Research Technique
P-RCPSP	Preemptive Resource Constrained Project Scheduling Problem
PSO	Particle Swarm Optimisation
RCPSP-GPR	Resource Constrained Project Scheduling Problem with Generalised Precedence Relations
RCPSP/max	Resource Constrained Project Scheduling Problem with maximal time lags
RCPSP/t	Resource Constrained Project Scheduling Problem with resource capacities and requests varying with time
SA	Simulated Annealing
SGS	Schedule Generation Scheme
s-SGS	Serial Schedule Generation Scheme
WBS	Work Breakdown Structure



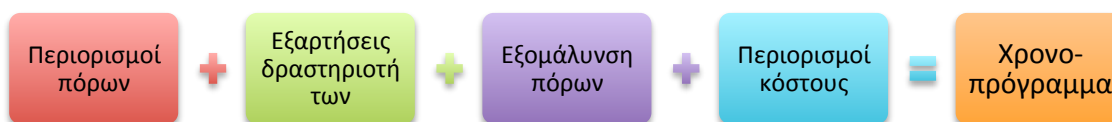
# Greek Synopsis

«Λήψη αποφάσεων στη διοίκηση έργων: Πολυστοχικός διευρυμένος χρονικός προγραμματισμός έργων υπό περιορισμένους πόρους»

## 0.1 Εισαγωγή

Η διαχείριση του χρόνου ή αλλιώς ο χρονικός προγραμματισμός των εργασιών ενός έργου αποτελεί μια από τις σημαντικότερες συνιστώσες της διοίκησης έργων. Κατά κανόνα συμπεριλαμβάνει διαδικασίες όπως τον καθορισμό των εργασιών που πρέπει να γίνουν ώστε να ολοκληρωθεί το έργο, τον καθορισμό της σειράς με την οποία πρέπει να εκτελεσθούν οι εργασίες, την εκτίμηση των απαιτούμενων πόρων για την πραγμάτωση κάθε μιας από τις εργασίες, την εκτίμηση του απαιτούμενου χρόνου με βάση τους διαθέσιμους πόρους και εν τέλει τη δημιουργία και παρακολούθηση του χρονικού προγράμματος που προκύπτει. Στόχος της παρούσας Διατριβής είναι η μελέτη της διαδικασίας ανάπτυξης χρονοδιαγραμμάτων έργων και η εύρεση μεθόδων για τη βελτιστοποίηση του τρόπου παραγωγής τους λαμβάνοντας υπόψη τους τιθέμενους περιορισμούς όσο αφορά την διαθεσιμότητα πόρων, το συνολικό κόστος, την απαιτούμενη ομαλότητα του προφίλ των χρησιμοποιούμενων πόρων. Στην παρούσα Διατριβή θα επικεντρωθούμε στον τρόπο σχηματισμού κατά το δυνατό βέλτιστων χρονοδιαγραμμάτων όσο αφορά κάποιες βασικές παραμέτρους όπως ο χρόνος, το κόστος και το προφίλ των πόρων. Συγκεκριμένα, το προς επίλυση πρόβλημα εκτείνεται κατά κύριο λόγο σε τρεις άξονες: χρόνο, κόστος και ομαλότητα προφίλ πόρων και αποσκοπούμε στην:

- ελαχιστοποίηση της διάρκειας του έργου δοθέντων προτεραιοτήτων για την εκτέλεση των δραστηριοτήτων και περιορισμένου πλήθους διαθέσιμων πόρων
- ελαχιστοποίηση του κόστους λαμβάνοντας υπόψη τη συσχέτιση διάρκειας δραστηριότητας – ποσότητας πόρων που της έχουν ανατεθεί και κόστους αλλά και των περιορισμένων χρηματικών πόρων που διατίθενται για την υλοποίηση του έργου
- εξομάλυνση των χρησιμοποιούμενων πόρων



Σχήμα 0.1 Σχηματική αναπαράσταση προς επίλυση προβλήματος

Συνεπώς, το παραπάνω πόνημα στρέφεται γύρω από τρεις βασικούς εννοιολογικούς άξονες: το κόστος, το χρόνο και τους πόρους και αποσκοπεί στην υποστήριξη του διαχειριστή του έργου στη προσπάθειά του να δημιουργήσει κατάλληλο χρονοδιάγραμμα για το εκάστοτε έργο με τέτοιο τρόπο ώστε να συγκεράσει τις συγκρουόμενες απαιτήσεις που αφορούν την ταχύτητα ολοκλήρωσης του έργου, το κόστος που θα απαιτείται και την βέλτιστη αξιοποίηση των διαθέσιμων πόρων και μόνον αυτών.

Ουσιαστικά, η παρούσα Διατριβή πραγματεύεται την ανάπτυξη πρότυπου πολυκριτήριου συστήματος αποφάσεων στη Διοίκηση Έργων και υβριδικής μεθόδου για την εύρεση του καλύτερου δυνατού χρονοπρογράμματος που να ικανοποιεί τις ανάγκες του λήπτη απόφασης.

Συγκεκριμένα αποσκοπεί στη σχεδίαση και ανάπτυξη ενός συστήματος για τη δημιουργία χρονοδιαγραμμάτων έργων υπό συνθήκες περιορισμένων πόρων και διαθέσιμου κεφαλαίου με την χρήση μεθόδων πολυκριτήριας ανάλυσης αποφάσεων για την βαθμονόμηση των προτεραιοτήτων κόστους, χρόνου και προφίλ πόρων.

Ένα έργο μπορεί να οριστεί ως ένα σύνολο από δραστηριότητες το οποίο έχει σαφώς καθορισμένη αρχή και συγκεκριμένο τέλος και αποσκοπεί στην επιτέλεση ενός συγκεκριμένου σκοπού κάνοντας χρήση καθορισμένων πόρων. Ένα χρονοπρόγραμμα συνήθως αποσκοπεί στον προγραμματισμό των δραστηριοτήτων του έργου, δηλαδή τον καθορισμό της έναρξής τους, με τέτοιο τρόπο ώστε να ικανοποιούνται οι σχέσεις προτεραιότητας και οι χρησιμοποιούμενοι πόροι να μην υπερβαίνουν την διατιθέμενη ποσότητα.

Παρόλο που το πρόβλημα του χρονοπρογραμματισμού έργου, μοιάζει να είναι σχετικά απλό, η μοντελοποίηση του με τρόπο που να καλύπτει όλες τις δυνατές περιπτώσεις που συναντώνται στην πράξη και η παροχή αποτελεσματικών τρόπων διαχείρισής τους, ισορροπώντας ανάμεσα στην πολυπλοκότητα του προβλήματος και στην ταχύτητα και αποτελεσματικότητα των προτεινόμενων λύσεων δεν είναι προφανής.

Το πρόβλημα του χρονοπρογραμματισμού έργων, ορίζεται ως η μέθοδος προγραμματισμού δραστηριοτήτων δοθέντων συγκεκριμένων ποσοτήτων διαθέσιμων πόρων για κάθε περίοδο της διάρκειας του έργου έτσι ώστε να ελαχιστοποιείται η αύξηση της συνολικής διάρκειας του έργου (Davis, 1974).

Βασικά συστατικά στοιχεία ενός χρονοπρογράμματος είναι τα ακόλουθα:

- Δραστηριότητες με χαρακτηριστικά γνώρισμα το αναγνωριστικό τους και τον τρόπο εκτέλεσης της δραστηριότητας, ο οποίος αφορά την διάρκεια της δραστηριότητας, το είδος και την ποσότητα πόρων που απαιτεί για την εκτέλεση της και εάν υπάρχουν χρηματοροές που σχετίζονται με αυτήν.
- Σχέσεις προτεραιότητας που καθορίζουν ποιες δραστηριότητες πρέπει να ολοκληρωθούν πριν η υπό εξέταση δραστηριότητα να μπορέσει να ξεκινήσει να εκτελείται
- Πόροι οι οποίοι ανήκουν σε μια από τις ακόλουθες κατηγορίες: α) ανανεώσιμοι – περιορισμένη διαθέσιμη ποσότητα ανά περίοδο του έργου πχ. εργάτες β) μη ανανεώσιμοι όπου έχουμε συγκεκριμένη ποσότητα διαθέσιμη για το σύνολο του έργου, πχ. προϋπολογισμός,



γ) διπλά περιορισμένοι και δ) μερικώς ανανεώσιμοι όπου έχουμε περιορισμό στις διαθέσιμες ποσότητες για ένα υποσύνολο της διάρκειας του έργου. Τέλος, οι πόροι χαρακτηρίζονται από τη λειτουργική τους χρήση και τη διαθέσιμη ποσότητα.

Στην πράξη όμως πέραν από την εύρεση χρονοπρογραμμάτων, όταν έχουμε συγκεκριμένες εξαρτήσεις μεταξύ των δραστηριοτήτων και συγκεκριμένο πλήθος διαθέσιμων προς χρήση πόρων, πρέπει να λαμβάνουμε υπόψη και το κόστος του έργου. Όμως συχνά η διάρκεια εκτέλεσης μιας δραστηριότητας μπορεί να μεταβληθεί με βάση το πόσο είμαστε διατεθειμένοι να πληρώσουμε για αυτήν, αφού ανάθεση περισσότερων ή διαφορετικών πόρων σε μια δραστηριότητα μειώνει τη συνολική της χρονική διάρκεια. Επομένως, προστίθεται άλλος ένας παράγοντας στο υπό μελέτη πρόβλημα του χρονοπρογραμματισμού, το κόστος και συνεπώς και ο αντίστοιχος περιορισμός που αφορά το σύνολο του διαθέσιμου προϋπολογισμού. Επιπλέον, γίνεται προφανές ότι στην περίπτωση αυτή οι δραστηριότητες θα έχουν πάνω από ένα δυνατούς τρόπους εκτέλεσης, ο οποίος θα προσδιορίζει το είδος και την ποσότητα χρησιμοποιούμενων πόρων και τη συνεπαγόμενη διάρκεια αλλά και χρηματική επιβάρυνση (κόστος).

Τέλος, δεν είναι επιθυμητό ούτε και πρακτικώς εφαρμόσιμο να έχουμε χρονοπρογράμματα στα οποία υπάρχουν απότομες αυξομειώσεις των πόρων οπότε προστίθεται στα παραπάνω και η απαίτηση για χρονοπρογράμματα στα οποία έχει γίνει εξισορρόπηση μεταξύ των σημείων μέγιστης και ελάχιστης ζήτησης πόρων για το σύνολο της χρονικής διάρκειας του έργου και μάλιστα χωρίς αυτό να οδηγεί σε αύξηση του χρόνου εκτέλεσης του ή του συνολικού πλήθους των χρησιμοποιούμενων πόρων και συνεπώς του κόστους του. Μελετώντας εις βάθος την βιβλιογραφία του συγκεκριμένου ερευνητικού πεδίου προκύπτει ότι ενώ υπάρχει πληθώρα αναλυτικών αλλά και ευρετικών μεθόδων για την εύρεση χρονοπρογραμμάτων όταν το ζητούμενο είναι η βελτιστοποίηση ως προς τον χρόνο ολοκλήρωσης ή το κόστος ή η ομαλότητα του προφίλ των πόρων, δεν υπάρχουν παρά περιορισμένες το πλήθος λύσεις για τις περιπτώσεις εκείνες όπου έχουμε συνδυασμό δυο εκ των τριών παραγόντων και πρακτικά καμία ολοκληρωμένη πρόταση για την περίπτωση που επιθυμούμε βελτιστοποίηση και ως προς τις τρεις παραμέτρους.

Στην παρούσα Διατριβή αποσκοπούμε στην εύρεση μιας συνολικής λύσης στο πρόβλημα του χρονοπρογραμματισμού έργων υπό συνθήκες περιορισμένων πόρων έτσι ώστε όταν εφαρμόσουμε στη πράξη τη μέθοδο αυτή το παραγόμενο αποτέλεσμα να δίνει την ελάχιστη δυνατή διάρκεια έργου στο ελάχιστο κόστος και με κατά το δυνατόν ομαλότερο προφίλ πόρων.

Επιπλέον, η ίδια η φύση του συγκεκριμένου προβλήματος οδηγεί σε μια σειρά από ερωτήματα που εξαρτώνται από το εκάστοτε πρόβλημα αλλά και από το συγκεκριμένο λήπτη απόφασης και την οπτική που έχει για το υπό εξέταση έργο και τις τρέχουσες συνθήκες που επικρατούν όσο αφορά την εργολήπτρια επιχείρηση, όπως για παράδειγμα εάν για το εκάστοτε υπό εξέταση έργο είναι το κόστος πιο σημαντικό από την ημερομηνία περάτωσης, είναι η ομαλότητα του προφίλ περισσότερο σημαντική από το κόστος ή τη διάρκεια του έργου, κ.α.

Επομένως πρόκειται για ένα πολυκριτήριο πρόβλημα με συγκρουόμενα κριτήρια απόφασης, οπότε η χρήση μηχανισμού υποστήριξης της διαδικασίας λήψης απόφασης κρίνεται απαραίτητη ώστε να προσδιοριστούν οι βαρύτερες των παραμέτρων κόστους χρόνο και ομαλότητας προφίλ.

Ως πολυκριτήριο πρόβλημα απόφασης μπορεί να επιλυθεί είτε συνδυάζοντας τις επί μέρους αντικειμενικές συναρτήσεις σε μια και χρησιμοποιώντας βάρη για την βαθμονόμηση είτε χρησιμοποιώντας το διάνυσμα που προκύπτει από τις αντικειμενικές και υπολογισμό της κατά Παρετο βέλτιστης λύσης. Στην προτεινόμενη προσέγγιση ο λήπτης απόφασης δύναται να επιλέξει να λάβει ως αποτελέσματα μείγμα λύσεων από κάθε μια από τις μονοκριτήριες αλλά και πολυκριτήριες προσεγγίσεις σύμφωνα με τις ανάγκες του, ώστε να επιλέξει την καταλληλότερη προσέγγιση για το προς χρονοπρογραμματισμό έργο.

Στην παρούσα Διατριβή προτείνεται μια ολιστική προσέγγιση για τον ορισμό του προβλήματος προγραμματισμού έργων υπό συνθήκες περιορισμένων πόρων (RCPSP). Στόχος είναι η

παροχή μιας ενιαίας εννοιολογικής έκφρασης του προβλήματος συμπεριλαμβάνοντας όλες τις διαφορετικές ντετερμινιστικές εκδοχές και παραλλαγές που απαντώνται στη βιβλιογραφία και έχουν πρακτική σημασία. Με βάση την ενιαία εννοιολογική προσέγγιση παρέχεται και η αντίστοιχη μαθηματική μοντελοποίηση του προβλήματος, όπως και η διαδικασία επίλυσης και οι απαιτούμενοι αλγόριθμοι για την υλοποίηση αυτής. Το πρόβλημα αντιμετωπίζεται ως βελτιστοποίηση ως προς μια ή περισσότερες μεταβλητές – στόχους. Ανώτερος στόχος είναι η παροχή ενός τρόπου μοντελοποίησης και επίλυσης των προβλημάτων χρονοπρογραμματισμού όπως αυτά συναντώνται στην πράξη χωρίς άλλους συμβιβασμούς και τροποποιήσεις των δεδομένων για να ταιριάζουν στο μοντέλο πέρα από την υπόθεση ότι τα εισαγόμενα στοιχεία προσεγγίζουν ικανοποιητικά την πραγματικότητα και είναι ντετερμινιστικά.

Στηριζόμενοι στο νέο ενιαίο μοντέλο, σχεδιάστηκε ένας υβριδικός αλγόριθμος, καλούμενος διαχειριστής (μοδερатор), που προσαρμόζεται στο μέγεθος και τα χαρακτηριστικά του προς επίλυση προβλήματος και βελτιστοποιεί τόσο ως προς έναν όσο και ως προς πολλαπλούς στόχους, υπολογίζοντας Παρετο βέλτιστες λύσεις και λαμβάνοντας ή μη υπόψη προτεραιότητες μεταξύ των αντικειμενικών στόχων.

Αποδείχθηκε πειραματικά ότι η χρήση του προτεινόμενου αλγορίθμου αυξάνει την ακρίβεια και σε κάποιες περιπτώσεις υπολογίζει και καλύτερες λύσεις χωρίς να επηρεάζει αρνητικά το χρόνο επίλυσης του προβλήματος. Επομένως, έχουμε έναν αξιόπιστο τρόπο για την επίλυση προβλημάτων χρονοπρογραμματισμού έργων είτε πρόκειται για απλά και κλασικά προβλήματα **RCPSP** είτε σύνθετους συνδυασμούς παραλλαγών και επεκτάσεων αυτού του τύπου προβλημάτων. Επιτυγχάνεται με αυτόν τον τρόπο η παροχή στους διευθυντές έργων ενός ευέλικτου μοντέλου το οποίο προσαρμόζεται στις ανάγκες τους αντί του ισχύοντος καθεστώτος όπου το πρόβλημα έπρεπε να προσαρμοστεί στο μοντέλο. Επιπλέον, παρέχεται και ένας διαφανής τρόπος επίλυσης του προβλήματος χωρίς ιδιαίτερα πολύπλοκα ή χρονοβόρα βήματα.

Η αυξημένη πολυπλοκότητα και η υψηλή πιθανότητα ανυπαρξίας εφικτών λύσεων στις πιο πολύπλοκες περιπτώσεις αντιμετωπίζεται με χαλάρωση των περιορισμών όταν αυτό κρίνεται αναγκαίο και με βάση πάντοτε τις επιλογές του διευθυντή έργου. Στην περίπτωση των πολλαπλών στόχων παράγονται πολλαπλά εναλλακτικά σενάρια με βάση τις Παρετο βέλτιστες λύσεις ή προσεγγίσεις αυτών, οι οποίες λαμβάνουν υπόψη και τα βάρη που ορίστηκαν για κάθε στόχο.

Η προτεινόμενη προσέγγιση αναπτύχθηκε σε τρεις φάσεις:

A) Ολιστική μοντελοποίηση του χρονοπρογραμματισμού έργου υπό περιορισμένους πόρους ώστε να συμπεριλαμβάνει όλες τις διαφορετικές εκδοχές του προβλήματος και να προσφέρει έναν εύκολο και κοντινό στην πραγματικότητα τρόπο μοντελοποίησης των έργων όπως αυτά συναντώνται στην πράξη. Το προτεινόμενο μαθηματικό μοντέλο καλύπτει τις περιπτώσεις που έχουμε πολλαπλούς τρόπους εκτέλεσης δραστηριοτήτων, χρονικά μεταβαλλόμενες διαθεσιμότητες ανανεώσιμων και μη ανανεώσιμων πόρων αλλά και μεταβαλλόμενες ανάγκες χρήσης των πόρων, δυνατότητα διακοπής ή μη των δραστηριοτήτων κατά την εκτέλεση τους, γενικευμένους τύπους προτεραιοτήτων (**FS, SS, SF, FF**) και παράθυρα χρόνου για την εκτέλεση των δραστηριοτήτων.

B) Ανάπτυξη γενετικού αλγορίθμου για τη διαχείριση πολυκριτήριας και μονοκριτήριας βελτιστοποίησης χρονοπρογραμμάτων με τα παραπάνω χαρακτηριστικά και δυνατότητας προσαρμογής στο εκάστοτε υπό επίλυση πρόβλημα και χρήσης κατάλληλου μετα-ευρετικού αλγορίθμου (**Genetic Algorithm, Simulated Annealing, Particle Swarm Optimization**, κ.λ.π.) και προσαρμογή υπαρχόντων εξελικτικών αλγορίθμων για την πολυκριτήρια βελτιστοποίηση τόσο στην περίπτωση του βαρυτικού αθροίσματος των επί μέρους κριτηρίων όσο και στην περίπτωση της Παρετο επίλυσης.

Γ) Ανάπτυξη κατάλληλης διεπαφής για τη χρήση των παραπάνω μεθόδων και πρακτική εφαρμογή τους σε πραγματικό έργο μεγάλου μεγέθους.

Συνοψίζοντας, προτείνεται ένα ενοποιημένο μαθηματικό μοντέλο και τρόποι εύρεσης λύσεων, αξιόπιστα και προσαρμοσμένα στις ανάγκες του διευθυντή έργων, όπως αυτές συναντώνται στην πράξη, ενώ ταυτόχρονα παρέχεται ευελιξία ως προς το είδος, το πλήθος και τα χαρακτηριστικά των παραγόμενων λύσεων αλλά και το βαθμό που κάθε αντικειμενικός στόχος θα πρέπει να βελτιστοποιηθεί.

## 0.2 Δομή Διατριβής

Η Διατριβή περιλαμβάνει ένα αρχικό κεφάλαιο στο οποίο εισάγονται τα βασικά στοιχεία του περιεχομένου της και στη συνέχεια διαρθρώνεται ως εξής:

Κεφάλαιο 2, όπου γίνεται τοποθέτηση του προβλήματος σε σχέση με τη βιβλιογραφία και ανάλυση των στοιχείων εκείνων που σχετίζονται άμεσα με το αντικείμενο της παρούσας Διατριβής. Η μελέτη εκτείνεται γύρω από τρεις άξονες: το χρονοπρογραμματισμό έργων, τη βελτιστοποίηση πολλαπλών στόχων και την πολυκριτήρια λήψη αποφάσεων. Ιδιαίτερη έμφαση δίνεται στις διάφορες εκδοχές του προβλήματος χρονοπρογραμματισμού και στις μεθόδους επίλυσης του.

Κεφάλαιο 3, όπου παρουσιάζεται η μεθοδολογική προσέγγιση του προβλήματος κυρίως σε σχέση με τη μαθηματική μοντελοποίηση και την αλγοριθμική σχεδίαση.

Κεφάλαιο 4, αφορά την ολιστική προσέγγιση του προβλήματος και αποσκοπεί στην απάντηση ερωτημάτων όπως: «ποιο είναι το ευρύτερο πλαίσιο του προς επίλυση προβλήματος», «ποιοι είναι οι στόχοι μας και πώς επηρεάζονται από περιβαλλοντικές παραμέτρους» και τελικά «τι δεδομένα μπορούμε να έχουμε όταν επιλύουμε το πρόβλημα». Περιγράφονται συνοπτικά τα βασικά στοιχεία του συστήματος, οι συσχετίσεις μεταξύ τους και οι τρόποι που αλληλοεπιδρούν. Η συστημική μεθοδολογία (*Soft Systems Methodology*) και η δυναμική συστημάτων (*System Dynamics*) χρησιμοποιούνται για να προσδιοριστεί το γενικότερο πλαίσιο του προβλήματος και τα βασικά στοιχεία του. Στη συνέχεια τα χαρακτηριστικά των επιθυμητών λύσεων χρησιμοποιούνται ως βάση για τον προσδιορισμό των στόχων της βελτιστοποίησης και η προηγούμενη ανάλυση για τον τελικό καθορισμό του προς επίλυση προβλήματος και των χαρακτηριστικών που πρέπει να έχει η μοντελοποίησή του για να μπορεί να είναι πρακτικά εφαρμόσιμη σε κάθε περίπτωση.

Κεφάλαιο 5, στο οποίο παρουσιάζεται η εννοιολογική και μαθηματική μοντελοποίηση του προβλήματος ως δυαδικό πρόβλημα γραμμικού προγραμματισμού.

Κεφάλαιο 6, στο οποίο γίνεται αναλυτική περιγραφή τόσο της προτεινόμενης διαδικασίας επίλυσης όσο και του νέου αλγόριθμου (*moderator*) και των βοηθητικών αλγόριθμων επίλυσης του προβλήματος. Πρόκειται για μια διαδικασία που απαρτίζεται από τρεις φάσεις, στην 1η γίνεται καθορισμός των δεδομένων, όπως εναλλακτικών τρόπων εκτέλεσης των δραστηριοτήτων, δυνατότητα διακοπής εκτέλεσης δραστηριοτήτων, διαθεσιμότητες και ανάγκες πόρων αντικειμενικών στόχων του προβλήματος και εάν απαιτείται, χρήση συστήματος υποστήριξης λήψης απόφασης για την εύρεση των σχετικών προτεραιοτήτων μεταξύ τους. Στη 2η φάση τα αρχικά δεδομένα και οι επιλογές του διευθυντή του έργου τροποποιούνται και προσαρμόζονται κατάλληλα για την διευκόλυνση της διαδικασίας επίλυσης. Στην 3η φάση καλείται ο διαχειριστής (*moderator*) αλγόριθμος για να ρυθμίσει τη διαδικασία επίλυσης, επιλέγοντας από γενιά σε γενιά τους αλγόριθμους επίλυσης που έχουν υψηλότερη απόδοση στο συγκεκριμένο στιγμιότυπο του προβλήματος και στους στόχους και τις προτεραιότητες αυτών, όπως καθορίστηκαν από τον διευθυντή έργου.

Κεφάλαιο 7, το οποίο παρουσιάζει τις πειραματικές διατάξεις και τα αποτελέσματα που επιτεύχθηκαν με τη χρήση του προτεινόμενου μοντέλου και αλγόριθμου και τα συγκρίνει με τις καλύτερες λύσεις, όπως αυτές προκύπτουν από τη βιβλιογραφία.

Κεφάλαιο 8, στο οποίο παρουσιάζεται η εφαρμογή του προτεινόμενου μοντέλου και της διαδικασίας επίλυσης σε πραγματικό πρόβλημα χρονοπρογραμματισμού έργου για την κτηματογράφηση συγκεκριμένων περιοχών της Ελλάδας. Συγκεκριμένα, παρουσιάζεται βήμα-βήμα η διαδικασία από τον αρχικό προσδιορισμό του προβλήματος ως την τελική επιλογή της βέλτιστης εναλλακτικής λύσης για τον προγραμματισμό του έργου. Ο προσδιορισμός των δεδομένων του έργου, δραστηριότητες, συσχετίσεις, ανάγκες σε πόρους και διαθεσιμότητα αλλά και οι προτε-

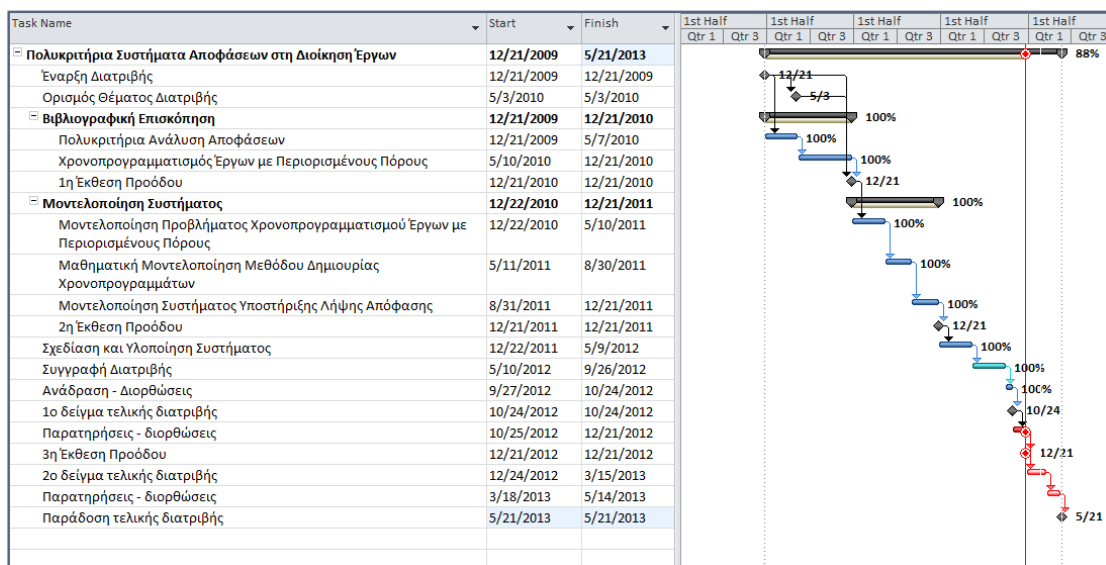
ραιότητες των αντικειμενικών στόχων και το είδος και πλήθος των ζητούμενων εναλλακτικών λύσεων, προέκυψαν κατόπιν σειράς συνεντεύξεων με το διευθυντή και την ομάδα έργου.

Τέλος, στο κεφάλαιο 9, παρουσιάζονται τα συμπεράσματα που προέκυψαν από την έρευνα, την ερευνητική συμβολή και την πρακτική συνεισφορά αυτής αλλά και προτάσεις για περαιτέρω ερευνητικές κατευθύνσεις.

### 0.3 Χρονοδιάγραμμα Εκπόνησης της Διατριβής

Το χρονοδιάγραμμα της εκπόνησης της διατριβής δίνεται στο σχήμα 0.2 και παρακάτω επεξηγούνται οι βασικότερες επιμέρους δραστηριότητες όπως αυτές έχουν πραγματοποιηθεί:

- Βιβλιογραφική επισκόπηση: αποσκοπεί στην οργάνωση των βασικών εννοιών που θα πραγματευτεί η διατριβή ώστε να παρουσιαστούν με σαφήνεια οι υπό εξέταση ερευνητικές περιοχές, τα κρίσιμα τεχνικά ή/και θεωρητικά ζητήματα που άπτονται αυτών σε σχέση με το κεντρικό θέμα της τρέχουσας διατριβής και οι μέθοδοι αντιμετώπισης των θεμάτων αυτών, όπως έχουν παρουσιαστεί στη διεθνή βιβλιογραφία. Η εργασία αυτή απαιτεί συνεχή ενημέρωση και ανανέωση των πηγών καθ' όλη την διάρκεια εκπόνησης της διατριβής.  
Αποτελέσματα:
  - Καταγραφή της τρέχουσας κατάστασης στο πεδίο έρευνας όπως αυτή αποτυπώνεται στην επιστημονική αρθρογραφία.
  - Κωδικοποίηση των τριών καλύτερων μεθόδων χρονοπρογραμματισμού (**Particle Swarm Optimisation, Simulated Annealing, Genetic Algorithm**) με περιορισμένους πόρους και σύγκριση αποτελεσμάτων με δημοσιευμένα αντίστοιχα αποτελέσματα (**PSPLib**)
- Μοντελοποίηση συστήματος: Στο στάδιο αυτό με βάση το θεωρητικό υπόβαθρο που αναλύθηκε στις προηγούμενες ενότητες, γίνεται ο σαφής καθορισμός του προς επίλυση προβλήματος και εν συνεχεία η μαθηματική μοντελοποίηση τόσο της μεθόδου δημιουργίας χρονοπρογραμμάτων όσο και η καθεαυτό μοντελοποίηση του προτεινόμενου συστήματος και των συνεπαγόμενων διαδικασιών. Αποτελέσματα:
  - Μαθηματική μοντελοποίηση του υπό εξέταση προβλήματος για τη δημιουργία νέου ολιστικού μοντέλου που να επιτρέπει την συνύπαρξη όλων των διαφορετικών παραμέτρων που παρουσιάζονται στην πράξη
  - Επιλογή οικογένειας αλγορίθμων για την επίλυση του προβλήματος και σχεδιασμός νέου εξειδικευμένου προσαρμοστικού αλγόριθμου επίλυσης.
  - Μοντελοποίηση του πολυκριτήριου προβλήματος και της διαδικασίας υποστήριξης λήψης απόφασης.
  - Πειραματικός έλεγχος ορθής λειτουργίας του αλγόριθμου.



Σχήμα 0.2 Χρονοπρογραμματισμός Εκπόνησης Διατριβής

- Υλοποίηση συστήματος: Ιδιαζούσης σημασίας στάδιο αποτελεί η σχεδίαση του προτεινόμενου συστήματος υποστήριξης λήψης απόφασης και η υλοποίηση αυτού ώστε να εφαρμοστούν στη πράξη οι μέθοδοι και διαδικασίες στις οποίες καταλήξαμε στα προηγούμενα στάδια και να ελεγχθεί τόσο η αποτελεσματικότητα όσο και η ορθότητα των παραγόμενων αποτελεσμάτων σε σύγκριση με τα αντίστοιχα δημοσιευμένα αποτελέσματα των προϋπαρχόντων συστημάτων. Αποτελέσματα
  - Υλοποίηση και έλεγχος ορθής λειτουργίας επιλεγθέντος αλγόριθμου για κάθε μια από τις παραμέτρους του προβλήματος βελτιστοποίησης και ανά ζεύγη.
  - Ανάπτυξη προσαρμοστικού αλγόριθμου για την επίλυση του προτεινόμενου ενοποιημένου μαθηματικού μοντέλου.
  - Υλοποίηση και έλεγχος ορθής λειτουργίας του προτεινόμενου αλγόριθμου για το σύνολο των παραμέτρων και των περιορισμών του ντετερμινιστικού προβλήματος εύρεσης βέλτιστου χρονοπρογράμματος.
  - Υλοποίηση και έλεγχος ορθής λειτουργίας αλγορίθμων διαχείρισης πολλαπλών αντικειμενικών στόχων.
  - Ενσωμάτωση των αλγορίθμων σε γραφικό περιβάλλον ως προσάρτημα του Μικροσοφτ Προθεστ.
  - Μελέτη Περίπτωσης: Χρήση του παραχθέντος μοντέλου και εργαλείων για τον χρονοπρογραμματισμό συγκεκριμένου έργου, «Εργασίες Κτηματογράφησης για τη Δημιουργία Ψηφιακής Κτηματολογικής Βάσης».

## 0.4 Ορισμός Προβλήματος

Ζητούμενο είναι τόσο η εύρεση ενός τρόπου που να διευκολύνει το διευθυντή του έργου να καθορίσει με ακρίβεια και σαφήνεια τα χαρακτηριστικά του υπό εξέταση έργου και των περιβαλλοντικών παραμέτρων που επηρεάζουν την επίλυση του, όσο και η παροχή μιας διαδικασίας επίλυσης του προβλήματος που θα δύναται να παράγει λύσεις προσαρμοσμένες στις ιδιαίτερες συνθήκες που προσδιορίζουν το συγκεκριμένο έργο, το μέγεθος της εργοληπτικής επιχείρησης, τη γενικότερη στρατηγική που έχει αυτή επιλέξει αλλά ταυτόχρονα να είναι και κλιμακούμενου μεγέθους με βάση το ίδιο το έργο και την κρισιμότητα αυτού. Επιπρόσθετα, η διαδικασία αυτή θα πρέπει να είναι αρκετά απλή και γρήγορη ώστε να επιτρέπει επαναληπτικές εκτελέσεις της, για την διευκόλυνση της παραγωγής εναλλακτικών σεναρίων, ώστε ο διευθυντής του έργου ή/και η ομάδα που είναι υπεύθυνη για τη λήψη των τελικών αποφάσεων και την επιλογή του χρονοδιαγράμματος που θα χρησιμοποιηθεί, να έχουν ένα επαρκές πλήθος εναλλακτικών λύσεων προς σύγκριση, συζήτηση και εν τέλει επιλογή.

Επομένως, πρώτο βήμα αποτελεί ο καθορισμός των χαρακτηριστικών εκείνων που θα πρέπει να έχουν τα παραγόμενα χρονοπρογράμματα ώστε να καλύπτουν τις κατά περίπτωση ανάγκες αλλά και ο καθορισμός των δομικών στοιχείων του έργου. Τα χαρακτηριστικά του χρονοπρογράμματος, είτε έμμεσα είτε άμεσα, συσχετισμένα με περιβαλλοντικές παραμέτρους, όπως για παράδειγμα τους στρατηγικούς στόχους της επιχείρησης, που συχνά περιλαμβάνουν το κέρδος, την ικανοποίηση των πελατών και την ελαχιστοποίηση των κινδύνων. Βέβαια οι ίδιοι οι παράγοντες αλλά και η σημασία καθενός σε σχέση με τους υπόλοιπους, συσχετίζονται με την ίδια την επιχείρηση και την τρέχουσα κατάσταση. Ένα 'καλό' χρονοπρόγραμμα για την εργοληπτική επιχείρηση, θα πρέπει να της δίνει τη δυνατότητα να:

- (α) ικανοποιεί τους πελάτες της, με το να οδηγεί στην παροχή του συμφωνηθέντος προϊόντος: το οποίο έχει τη ζητούμενη ποιότητα, έχει παραχθεί στον προσυμφωνηθέντα χρόνο, παρουσιάζει όλα εκείνα τα χαρακτηριστικά που είχαν συμφωνηθεί κατά την ανάθεση του έργου και βέβαια με το προϋπολογισθέν κόστος,
- (β) χρησιμοποιεί με βέλτιστο τρόπο το διαθέσιμο προϋπολογισμό και να ελαχιστοποιεί το κόστος, χωρίς όμως αυτό να λειτουργεί εις βάρος του (α) ,
- (γ) διαχειρίζεται τους ανθρώπινους πόρους με τέτοιο τρόπο ώστε να τηρούνται οι κείμενες νομοθεσίες αλλά και οι όροι των αντίστοιχων συμβάσεων, όσο αφορά το χρόνο και το είδος εργασίας αλλά και να γίνεται ισορροπημένη χρήση των υλικών και των μηχανημάτων,
- (δ) είναι κατά το δυνατόν εύρωστο ώστε μικρές αλλαγές σε διάρκειες δραστηριοτήτων ή διαθεσιμότητες πόρων να μπορούν να απορροφηθούν και να μην προκαλούν σοβαρές αλλαγές στο συνολικό χρονοπρόγραμμα.

Συνεπώς, ένα χρονοπρόγραμμα κρίνεται με βάση τα χαρακτηριστικά του, όπως τη διάρκεια, το κόστος, το πλήθος, το είδος και τον τρόπο χρήσης των πόρων και την ευρωστία του αλλά και το κατά πόσο κάθε ένα από αυτά τα στοιχεία είναι επιθυμητό και σε ποιο βαθμό από την πλευρά της εργοληπτικής επιχείρησης τη συγκεκριμένη χρονική περίοδο. Τα χαρακτηριστικά αυτά αποτελούν κατ' ουσία και τους αντικειμενικούς στόχους στην προσπάθεια βελτιστοποίησης ενός χρονοπρογράμματος. Είναι συχνά αντικρουόμενα μεταξύ τους και η προτεραιοποίησή τους είναι σχετικά πολύπλοκη αφού απαιτεί τον συνυπολογισμό διάφορων ποσοτικών αλλά και ποιοτικών κριτηρίων.

Τα δομικά στοιχεία ενός έργου είναι οι δραστηριότητες που το αποτελούν, οι συσχετίσεις μεταξύ τους και οι διαθέσιμοι πόροι, τύποι και ποσότητες. Ένα χρονοπρόγραμμα καθορίζει



τον τρόπο που θα εκτελεστεί η κάθε δραστηριότητα και τη χρονική στιγμή που αυτή θα αρχίσει να εκτελείται.

Κάθε δραστηριότητα έχει έναν ή περισσότερους τρόπους εκτέλεσης, κάθε ένας από τους οποίους μπορεί να συνεπάγεται διαφορετικές ποσότητες χρησιμοποιούμενων πόρων και αντίστοιχη διάρκεια ή διαφορετικούς τύπους πόρων και άλλη διάρκεια εκτέλεσης. Κάθε τρόπος εκτέλεσης ενδέχεται να οδηγεί σε διαφορετικό κόστος για την ίδια δραστηριότητα.

Οι δραστηριότητες μπορεί να επιτρέπεται να διακόπτονται κατά την εκτέλεση τους και να επανεκκινούν με μηδενικό κόστος ή και όχι. Τα σημεία διακοπής κάθε δραστηριότητας μπορεί να είναι προκαθορισμένα ή τυχαία με μόνο στόχο τη διευκόλυνση του προγραμματισμού των δραστηριοτήτων υπό συνθήκες περιορισμένων πόρων.

Οι δραστηριότητες μπορεί να απαιτούν είτε σταθερές είτε μεταβαλλόμενες ποσότητες πόρων κατά την εκτέλεση τους. Η ανάγκη μεταβαλλόμενων ποσοτήτων πόρων κάποιες φορές μπορεί να αντιμετωπισθεί με την περαιτέρω ανάλυση της εκάστοτε δραστηριότητας σε υποδραστηριότητες που έχουν σταθερή απαίτηση σε πόρους, αλλά όχι πάντα, μιας και οδηγεί σε αύξηση του πλήθους των δραστηριοτήτων και άρα του μεγέθους του προβλήματος.

Μια δραστηριότητα μπορεί να προγραμματιστεί οποιαδήποτε στιγμή μετά την ολοκλήρωση των προαπαιτούμενων της αλλά όχι νωρίτερα. Στην περίπτωση που υπάρχουν παράθυρα χρόνου για την έναρξη ή/και τη λήξη της, τότε πρέπει στο χρονοπρόγραμμα να λαμβάνονται υπόψη.

Διακρίνουμε τρεις διαφορετικούς τύπους πόρων: ανανεώσιμους, μη ανανεώσιμους και διπλά περιορισμένους. Στην κατηγορία των ανανεώσιμων πόρων ανήκουν εκείνοι οι πόροι που η διαθεσιμότητα τους ορίζεται ανά μονάδα χρόνου, όπως οι ανθρώπινοι πόροι, αν έχω 5 εργάτες σημαίνει ότι τους αναθέτω σε μια εργασία και μετά το πέρας της είναι ξανά διαθέσιμοι ή αλλιώς είναι πόροι που χρησιμοποιούνται αλλά δεν καταναλώνονται. Αντίθετα οι μη ανανεώσιμοι πόροι καταναλώνονται έτσι για παράδειγμα μη ανανεώσιμος πόρος είναι τα χρήματα, για τα οποία όταν λέμε ότι έχουμε 50.000 ευρώ τότε αυτό το ποσό είναι για ολόκληρο το έργο και κάθε φορά που χρησιμοποιείται μέρος του, όπως όταν γίνεται μια πληρωμή τότε αυτό το ποσό δεν θα είναι ξανά διαθέσιμο, αφαιρείται από το συνολικό ποσό. Τέλος, οι διπλά περιορισμένοι πόροι, έχουν όριο τόσο στη συνολική διαθέσιμη ποσότητα όσο και στην ανά χρονική μονάδα διαθέσιμη ποσότητα, όπως για παράδειγμα όταν έχουμε 3 μηχανές που μπορούν να λειτουργούν το πολύ 11 ώρες την ημέρα.

Το κόστος των ανανεώσιμων και των διπλά περιορισμένων πόρων είτε υπολογίζεται ως το γινόμενο του χρόνου εργασίας επί την τιμή αυτής είτε έχει τη μορφή μισθού/μισθώματος. Η διαφοροποίηση αυτή επηρεάζει ιδιαίτερα το κατά πόσο θα έχει ή όχι ιδιαίτερη σημασία η επίτευξη ομαλού προφίλ για τον συγκεκριμένο πόρο ή η ανάγκη ελαχιστοποίησης της μέγιστης χρήσης του.

Η διαθεσιμότητα των πόρων συνηθίζεται να θεωρείται σταθερή και αμετάβλητη για ολόκληρη τη διάρκεια του έργου. Η παραδοχή αυτή δεν προσεγγίζει ικανοποιητικά την πραγματικότητα μιας και συχνά στα έργα η διαθεσιμότητα των πόρων μεταβάλλεται, είτε πρόκειται για ανθρώπινους πόρους, οπότε έχουμε μεταβολές λόγω ασθένειας, αδειών άλλων έργων που διεκδικούν τους ίδιους πόρους, κ.α. αλλά και στην περίπτωση των μηχανημάτων, συντηρήσεις, βλάβες, κλπ.

Τέλος, οι περιορισμοί που τίθενται στο συγκεκριμένο πρόβλημα, μπορούν να ομαδοποιηθούν σε δυο κατηγορίες: αυτούς που προκύπτουν από τη λογική του προβλήματος και αυτούς που εκφράζουν συγκεκριμένες απαιτήσεις σε σχέση με το ζητούμενο σύνολο χρονοπρογραμμάτων. Στην πρώτη κατηγορία ανήκουν οι περιορισμοί που αφορούν τις σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων είτε προέρχονται από τον ορισμό του έργου είτε λόγω εισαγωγής δυνατότητας διακοπής της εκτέλεσης κάποιων δραστηριοτήτων. Στη δεύτερη ανήκουν περιορισμοί που αφορούν το κόστος/προϋπολογισμό του έργου, τις δια-

θεσιμότητες των πόρων και τις ημερομηνίες ολοκλήρωσης συγκεκριμένων δραστηριοτήτων. Οι περιορισμοί που ανήκουν στη 2η κατηγορία, μπορούν ενδεχομένως σε περιπτώσεις αδυναμίας εύρεσης καλών λύσεων στο πρόβλημα να "χαλαρώσουν" για να μπορέσουμε να οδηγηθούμε σε κάποιες εφικτές λύσεις ή σημαντικά καλύτερες όσο αφορά κάποιο από τα επιθυμητά χαρακτηριστικά, αφού για παράδειγμα αν και δεν είναι επιθυμητό θα μπορούσαμε ενδεχομένως σε περίπτωση ανάγκης να κάνουμε κάποιες προσλήψεις ή αγορά κάποιου μηχανήματος ή αύξηση του προϋπολογισμού.

## 0.5 Προτεινόμενη Μαθηματική Μοντελοποίηση

## 0.5.1 Ορισμοί

Η προτεινόμενη ολιστική μοντελοποίηση του προβλήματος του χρονοπρογραμματισμού έργων υπό περιορισμένους πόρους, με βάση τα παραπάνω μπορεί να εκφρασθεί ως εξής:

- Όλα τα δεδομένα θεωρούνται ντετερμινιστικά και εκ των προτέρων γνωστά.
- Ορίζουμε μοναδικό έργο αποτελούμενο από  $n$  δραστηριότητες συν μια βοηθητική δραστηριότητα, την 0 που αναπαριστά την έναρξη του έργου και μια βοηθητική δραστηριότητα την  $n+1$  που εκφράζει τη λήξη του έργου, με μηδενικές διάρκειες και απαιτήσεις σε πόρους. Ορίζουμε το σύνολο των δραστηριοτήτων ως  $V = \{0, 1, \dots, n, n+1\}$
- $T$  είναι ο χρονικός ορίζοντας του έργου, που υπολογίζεται ως το άθροισμα της μέγιστης διάρκειας κάθε δραστηριότητας του έργου.
- Το σύνολο των ανανεώσιμων πόρων συμβολίζονται με  $R^P$ . Για κάθε ανανεώσιμο πόρο  $k \in R^P$  η διαθέσιμη ποσότητα ανά χρονική περίοδο είναι μεταβλητή και ορίζεται ως  $\alpha_{kt}^P$ ,  $t = 0, 1, \dots, T-1$ .
- Το σύνολο των μη ανανεώσιμων πόρων ορίζεται ως  $R^V$ . Σε κάθε μη ανανεώσιμο πόρο  $l \in R^V$  αντιστοιχούμε ένα υποσύνολο  $\{t_{lx} | x = 0, \dots, X_l\}$  οφ  $\{0, 1, \dots, T\}$  όπου

$$0 = t_{l0} < \dots < t_{lx} < t_{l(x+1)} < \dots < t_{lX_l} = T.$$

Το υποσύνολο αυτό ορίζει μια διαμέριση του διαστήματος  $[0, T)$  το οποίο αποτελείται από τα υποδιαστήματα  $I_{lx} = [t_{lx}, t_{l(x+1)})$ ,  $x = 0, \dots, X_l - 1$ . Η συνολική κατανάλωση του μη ανανεώσιμου πόρου  $l$  για την περίοδο  $I_{lx}$  του έργου δεν μπορεί να υπερβάνει το  $\alpha_{lI_{lx}}^V$ .

- Κάθε δραστηριότητα  $i$  αντιστοιχίζεται σε ένα σύνολο  $M_i$  εναλλακτικών τρόπων εκτέλεσης.
  - Κάθε δραστηριότητα  $i$  μπορεί να εκτελεστεί με ακριβώς ένα τρόπο  $m \in M_i$  σε κάθε διακριτό χρονοπρόγραμμα.
  - Κάθε τρόπος εκτέλεσης  $m$  έχει διάρκεια  $d_{im}$  χρονικές μονάδες.
  - Η δραστηριότητα  $i$  με τρόπο εκτέλεσης  $m$  απαιτεί  $r_{imk}^P$  ανανεώσιμους πόρους του τύπου  $k \in R^P$  στην  $\tau_{im}$ -στη περίοδο εκτέλεσης της,  $\tau_{im} = 0, \dots, d_{im} - 1$ . Οι απαιτούμενοι πόροι δεν καταναλώνονται, απλά χρησιμοποιούνται και επιστρέφονται στη δεξαμενή πόρων μετά την ολοκλήρωση της αντίστοιχης δραστηριότητας.
  - Η δραστηριότητα  $i$  με τρόπο εκτέλεσης  $m$  απαιτεί την κατανάλωση  $r_{iml}^V$  μη ανανεώσιμων πόρων του τύπου  $l \in R^V$  στην  $\tau_{im}$ -στη περίοδο εκτέλεσης της,  $\tau_{im} = 0, \dots, d_{im} - 1$ .
- Κάθε τρόπος εκτέλεσης  $m$  μιας δραστηριότητας  $i$  καθορίζει την δραστηριότητα είτε ως διακοπτόμενη είτε όχι.
  - Το σύνολο των μη διακοπτόμενων τρόπων εκτέλεσης μιας δραστηριότητας  $i$  ορίζεται ως  $M_i^P$  και το σύνολο των διακοπτόμενων τρόπων εκτέλεσης ως  $M_i^V$ .
  - Η διάρκεια  $d_{im}$  μιας δραστηριότητας  $i$  μπορεί να διακοπεί σε  $z_{im} + 1$  διαστήματα μοναδιαίας ή μεγαλύτερης ακέραιας διάρκειας. Κάθε τμήμα ορίζεται ως  $p_{imq}$ ,  $q = 0, \dots, z_{im}$  και έχει διάρκεια  $d_{imq}$ . Κάθε τμήμα λαμβάνει τιμή έναρξης εκτέλεσης  $s_{imq}$  και ολοκλήρωσης  $f_{imq}$ .
  - Για την απλοποίηση της παρακάτω μοντελοποίησης όλες οι δραστηριότητες θεωρούνται διακοπτόμενες και απλά όσοι τρόποι εκτέλεσης δεν είναι διακοπτόμενοι έχουν μηδενικό πλήθος σημείων διακοπής, όπως για παράδειγμα η δραστηριότητα έναρξης του έργου  $z_{im} = 0$ . Αν  $s_{imq}$ ,  $q = 0, \dots, z_{im}$ , είναι γνωστά, τότε μπορούμε να μετατρέψουμε τις

περιόδους εκτέλεσης της δραστηριότητας  $i$  που εκτελείται με τον τρόπο  $m$ ,  $\tau_{im}$ , σε περιόδους  $t$  του έργου ως εξής:

$$t = \begin{cases} \tau_{im} + s_{im0}, & \tau_{im} = 0, \dots, d_{im0} - 1 \\ \tau_{im} + s_{im1}, & \tau_{im} = d_{im0}, \dots, d_{im1} - 1 \\ \vdots \\ \tau_{im} + s_{imz_{im}}, & \tau_{im} = d_{im(z_{im}-1)}, \dots, d_{imz_{im}} - 1 \end{cases} . \quad (0.1)$$

- Με βάση τα παραπάνω:
  - $s_{im0}$  είναι η χρονική στιγμή έναρξης της δραστηριότητας  $i \in V$  εκτελούμενης με τον τρόπο  $m$  και το πρώτο της τμήμα  $p_{im0}$ .  $f_{imz_{im}}$  είναι η χρονική στιγμή ολοκλήρωσης της δραστηριότητας  $i \in V$  εκτελούμενης με τον τρόπο  $m$  και το τελευταίο της τμήμα  $p_{imz_{im}}$ .
  - Η δραστηριότητα έναρξης του έργου έχει μοναδικό τρόπο εκτέλεσης  $m = 0$ , διάρκεια  $0$  χρονικών μονάδων και δεν μπορεί να διακοπεί. Επομένως,  $z_0 = 0$  και  $s_{00z_0} = s_{000}$ . Συνεπώς, θέτοντας τη χρονική έναρξη του έργου στη χρονική στιγμή  $0$  δίνει ότι  $s_{000} = 0$ .
  - Αντίστοιχα για τη δραστηριότητα λήξης του έργου έχουμε  $z_{n+1} = 0$  και  $f_{(n+1)0z_{n+1}} = f_{(n+1)00}$  δίνει τη συνολική διάρκεια του έργου.
- Ορίζουμε τις ακόλουθες σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων: **start-to-start**  $SS_{imjn}$ , **finish-to-finish**  $FF_{imjn}$ , **finish-to-start**  $FS_{imjn}$ , **start-to-finish**  $SF_{imjn}$ , με ελάχιστες και μέγιστες υστερήσεις μεταξύ των δραστηριοτήτων  $i$  και  $j$  που εκτελούνται με τους τρόπους  $m$  και  $n$  αντίστοιχα.
- Μετά τον προσδιορισμό της διάρκειας και των χρονικών υστερήσεων μεταξύ των δραστηριοτήτων, μετατρέπονται όλες οι σχέσεις προτεραιότητας σε ένα μοναδικό τύπο, τον  $SS$ , χρησιμοποιώντας τους ακόλουθους κανόνες μετατροπής:

**Start to Start:**

$$s_{im0} + SS_{imjn}^{\min} \leq s_{jn0} \rightarrow s_{im0} + \delta_{imjn} \leq s_{jn0},$$

$$\text{with } \delta_{imjn} = SS_{imjn}^{\min},$$

$$s_{im0} + SS_{imjn}^{\max} \geq s_{jn0} \rightarrow s_{jn0} + \delta_{jnim} \leq s_{im0},$$

$$\text{with } \delta_{jnim} = -SS_{imjn}^{\max}.$$

**Start to Finish:**

$$s_{im0} + SF_{imjn}^{\min} \leq f_{jnz_{jn}} \rightarrow s_{im0} + \delta_{imjn} \leq s_{jn0},$$

$$\delta_{imjn} = SF_{imjn}^{\min} - d_{jn},$$

$$s_{im0} + SF_{imjn}^{\max} \geq f_{jnz_{jn}} \rightarrow s_{jn0} + \delta_{jnim} \leq s_{im0},$$

$$\delta_{jnim} = -(SF_{imjn}^{\max} - d_{jn}).$$

(0.2)

**Finish to Start:**

$$f_{imz_{im}} + FS_{imjn}^{\min} \leq s_{jn0} \rightarrow s_{im0} + \delta_{imjn} \leq s_{jn0},$$

$$\text{with } \delta_{imjn} = FS_{imjn}^{\min} + d_{im},$$

$$f_{imz_{im}} + FS_{imjn}^{\max} \geq s_{jn0} \rightarrow s_{jn0} + \delta_{jnim} \leq s_{im0},$$

$$\delta_{jnim} = -(FS_{imjn}^{\max} + d_{im}).$$

**Finish to Finish:**

$$f_{imz_{im}} + FF_{imjn}^{\min} \leq f_{jnz_{jn}} \rightarrow s_{im0} + \delta_{imjn} \leq s_{jn0},$$

$$\delta_{imjn} = FF_{imjn}^{\min} + d_{im} - d_{jn},$$

$$f_{imz_{im}} + FF_{imjn}^{\max} \geq f_{jnz_{jn}} \rightarrow s_{jn0} + \delta_{jnim} \leq s_{im0},$$

$$\delta_{jnim} = -(FF_{imjn}^{\max} + d_{im} - d_{jn}).$$

- Το διάνυσμα  $S = (s_{imq})_{i=0,1,\dots,n,n+1 \ q=0,\dots,z_{im}}$  ορίζει ένα χρονοπρόγραμμα του έργου. Το χρονοπρόγραμμα  $S$  καλείται εφικτό ως προς το χρόνο και τους πόρους αν οι περιορισμοί που αφορούν τη διαθεσιμότητα των πόρων και τις προτεραιότητες των δραστηριοτήτων τηρούνται.
- $Act(t)$  ορίζεται ως το σύνολο των υπό εκτέλεση δραστηριοτήτων τη χρονική περίοδο  $t$ ,  $t = 0, 1, \dots, T$ .

Στόχος είναι ο προσδιορισμός των τρόπων εκτέλεσης  $m$  και των ενάρξεων  $s_{imq}$  όλων των δραστηριοτήτων  $i = 1, \dots, n$  και όλων των τμημάτων τους  $q = 0, \dots, z_{im}$  με τέτοιο τρόπο ώστε να επιτυγχάνεται βελτιστοποίηση των αντικειμενικών στόχων υπό τους δοθέντες περιορισμούς.

Ο πίνακας 0.1 συνοψίζει τους συμβολισμούς που εισήχθησαν σε αυτή την ενότητα.

Στη συνέχεια οι παραπάνω ορισμοί παρουσιάζονται με τη χρήση ενός σύντομου αριθμητικού παραδείγματος, όπως φαίνεται στις εικόνες: 0.3 - 0.5, όπου περιγράφεται ένα έργο με 6 δραστηριότητες πλέον της έναρξης (0) και λήξης του έργου (7). Για κάθε δραστηριότητα προσδιορίζονται η διάρκεια, οι απαιτήσεις σε ανανεώσιμους και μη ανανεώσιμους πόρους, αν και σε ποια σημεία μπορεί να διακοπεί η εκτέλεση της και τις σχέσεις προτεραιότητας, συμπεριλαμβανομένων των χρονικών παραθύρων.

Τέλος, στην εικόνα 0.5, παρουσιάζεται ο γράφος που προκύπτει για το έργο σε δυο διαφορετικές, τυχαίες, επιλογές τρόπων εκτέλεσης των δραστηριοτήτων:  $M_1$  (0,0,0, 0,0,0,0,0) και  $M_2$  (0,0,2,1,1,2,1,0).

Πίνακας 0.1 Μαθηματικοί Συμβολισμοί

Σύμβολο	Περιγραφή
$V = \{0, 1, \dots, n, n+1\}$	το σύνολο των δραστηριοτήτων $i$
$n$	πλήθος πραγματικών δραστηριοτήτων
$G(V, A)$	κατευθυνόμενος γράφος χρονικών περιορισμών
$T$	ο χρονικός ορίζοντας, άθροισμα της μέγιστης διάρκειας κάθε δραστηριότητας
$t$	περίοδοι, δείκτης του $T$
$[t, t+1)$	χρονικό διάστημα που αντιστοιχεί στην περίοδο $t$
$Act(t)$	σύνολο όλων των δραστηριοτήτων που είναι υπό εκτέλεση τη χρονική στιγμή $t$ , $t = 0, 1, \dots, T$
$R^p$	σύνολο ανανεώσιμων πόρων
$\alpha_{kt}^p$	μεταβλητή ποσότητα διαθέσιμων πόρων τύπου $k$ , $t = 0, \dots, T-1$
$R^v$	σύνολο μη ανανεώσιμων πόρων
$t_{lx}$	κάθε μη ανανεώσιμος πόρος $l \in R^v$ αντιστοιχίζεται σε ένα υποσύνολο $\{t_{lx}   x = 0, \dots, X_l\}$ , $\{0, 1, \dots, T\}$ με $0 = t_{l0} < \dots < t_{lx} < t_{l(x+1)} < \dots < t_{lX_l} = T$
$I_{lx}$	υποδιαστήματα $I_{lx} = [t_{lx}, t_{l(x+1)})$ , $x = 0, \dots, X_l - 1$ που συνιστούν μια διαμέριση του $[0, T)$
$\alpha_{I_{lx}}^v$	μεταβλητό πλήθος διαθέσιμης ποσότητας μη ανανεώσιμων πόρων $l$
$M_i$	σύνολο εναλλακτικών τρόπων εκτέλεσης της δραστηριότητας $i$
$M_i^p$	σύνολο μη διακοπτόμενων τρόπων εκτέλεσης της $i$
$M_i^m$	σύνολο διακοπτόμενων τρόπων εκτέλεσης της $i$
$d_{im}$	διάρκεια της δραστηριότητας $i$ που εκτελείται με τον τρόπο $m$
$r_{imk}^p$	για τη δραστηριότητα $i$ ανά περίοδο χρήσης του πόρου τύπου $k$ όταν εκτελείται με τον τρόπο $m$
$r_{iml}^v$	για τη δραστηριότητα $i$ ανά περίοδο χρήσης του πόρου τύπου $l$ όταν εκτελείται με τον τρόπο $m$
$z_{im}$	πλήθος διακοπών της δραστηριότητας $i$ όταν εκτελείται με τον τρόπο $m$ , $z_{im} = 0, \dots, d_{im} - 1$
$p_{imq}$	τμήμα της διακοπτόμενης δραστηριότητας $i$ με $q = 0, 1, 2, \dots, z_{im}$
$d_{imq}$	διάρκεια του τμήματος $q$ της δραστηριότητας $i$ όταν εκτελείται με τον τρόπο $m$
$s_{imq}$	χρονική στιγμή έναρξης του τμήματος $q$ της δραστηριότητας $i$ όταν εκτελείται με τον τρόπο $m$
$f_{imq}$	χρονική στιγμή ολοκλήρωσης του τμήματος $q$ της δραστηριότητας $i$ όταν εκτελείται με τον τρόπο $m$
$s_{im0}$	έναρξη της δραστηριότητας $i$
$f_{imz_m}$	χρονική στιγμή ολοκλήρωσης της δραστηριότητας $i$
$s_{000}$	χρονική στιγμή έναρξης του έργου
$f_{(n+1)00}$	χρονική στιγμή λήξης του έργου
$S = (s_{imq})$	χρονοπρόγραμμα, διάγραμμα των χρόνων έναρξης όλων των τμημάτων όλων των δραστηριοτήτων του έργου

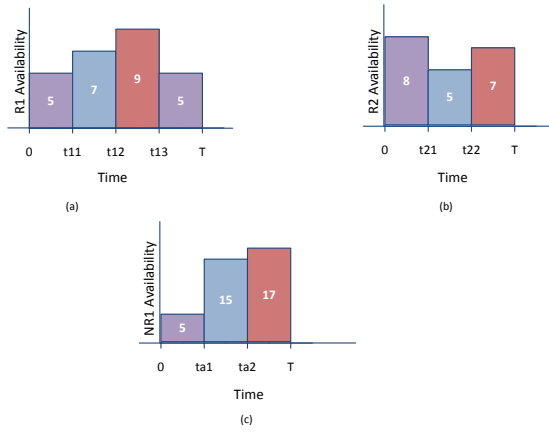
id	mode	dur	R1			R2			NR1		Preemption
			$r^p$	$r^m$	$r^k$	$r^p$	$r^m$	$r^k$	$r^p$	$r^m$	
$i$	$m_i$	$d_{im}$	$t_{i1}$	$t_{i2}$	$t_{i3}$	$t_{21}$	$t_{22}$	$ta_1$	$ta_2$	Y/N	
0	0	0	0	0	0	0	0	0	0	0N	
1	0	5	3	2	1	2	1	1	2	2N	
1	1	9	2	1	1	1	2	2	2	1N	
2	6	1	2	3	1	2	3	2	3	2N	
2	0	5	1	1	1	1	2	1	1	3N	
2	1	7	2	0	2	1	2	1	2	2Y	
2	9	0	2	2	2	1	1	1	1	1Y (30%)	
3	0	4	1	2	1	4	1	2	4	4N	
1	5	2	1	2	2	4	2	2	1	1N	
2	5	1	1	3	1	2	2	2	2	2Y (20%)	
4	0	5	2	1	2	1	2	3	2	2Y (50%)	
1	4	1	2	2	2	1	3	2	1	2N	
2	4	1	1	2	1	2	4	2	1	2N	
5	0	5	2	1	2	1	3	2	3	3N	
1	8	2	1	2	1	3	1	2	1	2Y (25%)	
2	4	2	1	2	3	1	1	1	3	3N	
6	0	4	1	2	1	2	1	3	1	1N	
1	5	1	2	1	1	2	3	2	1	2Y (20%)	
2	8	1	2	1	2	1	2	2	1	2Y (75%)	
7	0	0	0	0	0	0	0	0	0	0N	

(a)

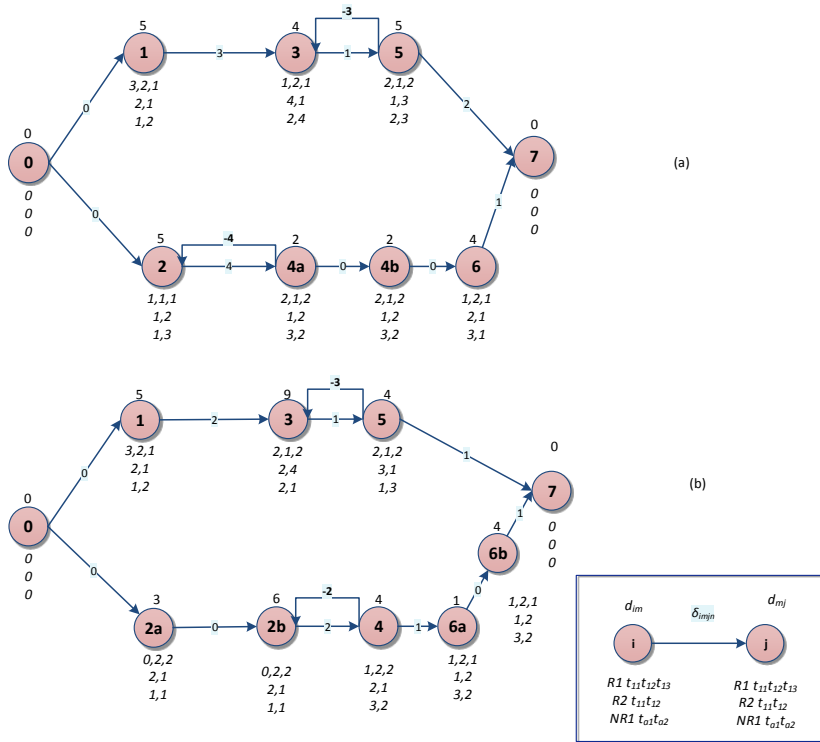
id	m	0	1	2	3	4	5	6	7								
0	0	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0
1	0	0		0													
1	0					3	2	1									
1	1					6	4	1									
2	2					1	2	3									
2	0								4	4	1						
1	1								1	1	4						
2	2								2	2	1						
3	0											1	1	2			
1	1											2	2	1			
2	2											4	1	2			
4	0					-4	-4	-6							0	4	1
1	1					-2	-1	-2							2	1	7
2	2					-6	-2	-2							1	1	4
5	0																2
1	1								-3	-1	-4						2
2	2								-4	-4	-6						2
6	0																1
1	1																1
2	2																4
7	0																0

(b)

Σχήμα 0.3 (α) Δραστηριότητες, τρόποι εκτέλεσης, απαιτήσεις σε πόρους και σημεία διακοπής των δραστηριοτήτων του έργου, (β) *minimal* και *maximal lag* των δραστηριοτήτων ανά τρόπο εκτέλεσης



Σχήμα 0.4 (α) διαθεσιμότητα πόρου P1 σε σχέση με το χρόνο, (β) διαθεσιμότητα πόρου P1 σε σχέση με το χρόνο και (γ) διαθεσιμότητα μη ανανεώσιμου πόρου NP1 σε σχέση με τις περιόδους  $I_{t0} = [0, t_{a1})$ ,  $I_{t1} = [t_{a1}, t_{a2})$  και  $I_{t2} = [t_{a2}, T)$



Σχήμα 0.5 (α)Γράφος του έργου για επιλογή τρόπων εκτέλεσης  $M_1(0,0,0,0,0,0,0,0)$  και (β)  $M_2(0,0,2,1,1,2,1,0)$



### 0.5.2 Αντικειμενικοί στόχοι

Στο χρονοπρογραμματισμό έργων αναζητάμε ένα πρόγραμμα αναφοράς όπου ο χρόνος ολοκλήρωσης του έργου, το κόστος, η ομαλότητα του προφίλ των πόρων, η ελάχιστη μέγιστη χρήση συγκεκριμένων πόρων και η ευρωστία, είναι οι κύριοι στόχοι αναφορικά με τη διαδικασία βελτιστοποίησης.

Η διάρκεια του έργου είναι ένα κανονικό μέτρο απόδοσης (γνησίως μονότονο) του χρονοπρογράμματος ενός έργου και εκφράζεται ως τη χρονική στιγμή έναρξης της δραστηριότητας τέλους του έργου:

$$\min s_{(n+1)0}. \quad (0.3)$$

Επιπλέον, μπορεί να προστεθεί ως "τιμωρία" στην συνάρτηση βελτιστοποίησης ένας παράγοντας που εκφράζει το χρόνο απόκλισης από συγκεκριμένες ημερομηνίες που αντιστοιχούν π.χ. σε παραδοτέα ή ορόσημα του έργου:

$$\min s_{(n+1)0} + T_{over}, \quad (0.4)$$

όπου  $T_{over}$  είναι το πλήθος των χρονικών περιόδων που καθυστέρησε η ολοκλήρωση της αντίστοιχης δραστηριότητας. Ο παράγοντας αυτός μπορεί να συνδέεται με συγκεκριμένες βαρύτητες που αντικατοπτρίζουν την κρισιμότητα της καθυστέρησης:

$$T_{over} = \sum_{i=1}^n w_i (T_i - fimz_{im}), \quad (0.5)$$

όπου  $T_i$  είναι η ημερομηνία παράδοσης (*deadline*) της δραστηριότητας  $i$ .

Οι αντικειμενικοί στόχοι που σχετίζονται με τους πόρους, κατά κύριο λόγο αφορούν τη μείωση της μέγιστης χρήσης πόρων υψηλού κόστους ή χαμηλής διαθεσιμότητας αλλά τις έντονες διακυμάνσεις στις ποσότητες χρησιμοποιούμενων πόρων σε συνάρτηση με το χρόνο. Όταν στόχος είναι η μείωση των διακυμάνσεων και επομένως η εξομάλυνση του προφίλ των πόρων, γίνεται υπολογισμός της ιδανικής μέσης χρήσης του κάθε πόρου και στη συνέχεια άθροιση των αποκλίσεων ανά χρονική μονάδα και πόρο:

$$\min \sum_{k \in R^p} \sum_{t=0}^{f_{(n+1)00}} \left| \left( \sum_{i=1}^N r_{imk}^p \tau_{im} \right) - \frac{\sum_{t=0}^{f_{(n+1)00}} \sum_{i=1}^N r_{imk}^p \tau_{im}}{f_{(n+1)00}} \right|. \quad (0.6)$$

Σημειώνεται εδώ, ότι τόσο στη παραπάνω εξίσωση όσο και στις επόμενες με  $\tau_{im}$  συμβολίζουμε την αντίστοιχη τιμή του  $t$  όπως προκύπτει από την αντίστοιχη εξίσωση (0.1).

Αν  $c_k$  είναι το μοναδιαίο κόστος που σχετίζεται με τον ανανεώσιμο πόρο  $k$  και  $c_0$  είναι το άθροισμα των έμμεσων κοστών που αντιστοιχούν στο έργο, τότε το συνολικό κόστος μπορεί να εκφραστεί ως το άθροισμα των ποσοτήτων αυτών:

$$\min c_0 + \sum_{k \in R^p} c_k \left( \sum_{t=0}^T \sum_{i=1}^N r_{imk}^p \tau_{im} \right). \quad (0.7)$$

Στην περίπτωση που υπάρχει συγκεκριμένος προϋπολογισμός, αυτός μπορεί να χρησιμοποιηθεί είτε ως περιορισμός είτε ως ποινή στην αντικειμενική συνάρτηση, όπως και προηγουμένως.

Η μείωση της μέγιστης κατανάλωσης ενός ή περισσότερων πόρων, χρησιμοποιείται συνήθως για πόρους που είτε το κόστος τους είναι τέτοιο ώστε και ακόμα και μια μόνο μονάδα επιπλέον να χρησιμοποιηθεί επηρεάζει σημαντικά τον προϋπολογισμό είτε είναι ιδιαίτερα δυσέυρετες. Σε αυτές τις περιπτώσεις στοχεύουμε στην ελαχιστοποίηση της μέγιστης κατανάλωσης του πόρου:

$$\min \max \left\{ \sum_{i=1}^N r_{imk}^p \tau_{im} \mid t = 0, 1, \dots, T-1 \right\}. \quad (0.8)$$

Τέλος, όσο αφορά την αύξηση της ευρωστίας του χρονοπρογράμματος προτείνεται η μεγιστοποίηση είτε του συνολικού είτε του ελεύθερου περιθωρίου του έργου. Στην προτεινόμενη προσέγγιση θέτουμε το συνολικό περιθώριο, την διαφορά νωρίτερης και αργότερης έναρξης των δραστηριοτήτων ως αντικειμενικό στόχο με σκοπό την ελαχιστοποίηση του:

$$\max \sum_{i=1}^n (LS_{im} - ES_{im}), \quad (0.9)$$

όπου  $ES_{im}$  είναι η ενωρίτερη έναρξη και  $LS_{im}$  η αργότερη έναρξη της δραστηριότητας  $i$  όταν εκτελείται με τον τρόπο  $m$ — στην περίπτωση διακοπτόμενης δραστηριότητας λαμβάνονται υπόψη οι τιμές του πρώτου και του τελευταίου τμήματος αυτής.

Οι παραπάνω στόχοι συχνά είναι αλληλοσυγκρουόμενοι και εκφράζουν διαφορετικές οπτικές και απαιτήσεις σε σχέση με το υπό σχεδίαση χρονοδιάγραμμα. Ο κατάλληλος συνδυασμός στόχων αλλά και ενδεχομένως η ιεράρχηση τους εξαρτάται από το εκάστοτε στιγμιότυπο του έργου αλλά και τα περιβαλλοντικά χαρακτηριστικά και την ίδια την εργολήπτρια επιχείρηση.

### 0.5.3 Περιορισμοί

Οι δραστηριότητες πρέπει να εκτελούνται με τέτοιο τρόπο ώστε να ικανοποιούνται οι γενικευμένες σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, που μπορούν να μετατραπούν σε έναν τύπο σχέσεων προτεραιότητας χρησιμοποιώντας τους κανόνες των εξισώσεων (2.32), οπότε προκύπτει ο γενικευμένος περιορισμός:

$$\begin{aligned} \delta_{imjn} &\leq s_{jn0} - s_{im0}, \\ \forall (i, j) \in A, \forall m \in M_i, \forall n \in M_j. \end{aligned} \quad (0.10)$$

Η έναρξη κάθε τμήματος  $q$  μιας διακοπτόμενης δραστηριότητας  $i$ , πρέπει να ξεκινάει μετά το πέρας του προηγούμενου τμήματος της ίδιας άρα τουλάχιστον  $d_{im(q-1)}$  μονάδες χρόνου από την έναρξη του τμήματος  $q-1$  της ίδιας:

$$\begin{aligned} s_{im(q-1)} + d_{im(q-1)} &\leq s_{imq}, \\ \forall i = 1, \dots, n, \forall m \in M_i, \forall q = 1, \dots, z_{im}. \end{aligned} \quad (0.11)$$

Επιπρόσθετα, αυτός ο περιορισμός καλύπτει και την ανάγκη συσχέτισης των τμημάτων μιας δραστηριότητας με  $FS$  σχέσεις για να διατηρηθεί η σειρά εκτέλεσης των τμημάτων της δραστηριότητας.

Προφανώς, η δραστηριότητα έναρξης του έργου πρέπει να είναι η 1η που προγραμματίζεται τη χρονική στιγμή  $t = 0$ :

$$s_{000} = 0, \quad (0.12)$$

και η δραστηριότητα τέλους πρέπει να προγραμματιστεί μετά την ολοκλήρωση όλων των υπόλοιπων δραστηριοτήτων:

$$\begin{aligned} s_{(n+1)00} &\geq f_{imz_{im}}, \\ \forall i = 0, 1, \dots, n, \forall m \in M_i. \end{aligned} \quad (0.13)$$

Οι χρησιμοποιούμενες ποσότητες ανανεώσιμων και μη ανανεώσιμων πόρων κάθε χρονική στιγμή  $t$  πρέπει να είναι μικρότερες ή ίσες της αντίστοιχης διαθέσιμης ποσότητας εκείνη τη χρονική στιγμή:

$$\begin{aligned} \sum_{i \in Act(t)} r_{imk}^p \tau_{im} &\leq \alpha_{kt}^p, \\ \forall k \in R^p, \forall t = 0, 1, \dots, T-1, \forall m \in M_i, \end{aligned} \quad (0.14)$$

$$\begin{aligned} \sum_{t=0}^{t_{l(x+1)}-1} \sum_{i \in Act(t)} r_{iml}^v \tau_{im} &\leq \alpha_{lIx}^v, \\ \forall l \in R^v, \forall x = 0, \dots, X_l-1, \forall m \in M_i. \end{aligned} \quad (0.15)$$

Στην περίπτωση που είτε κάποια ορόσημα είτε ο προϋπολογισμός του έργου θέλουμε να ληφθούν υπόψη ως περιορισμοί και όχι ως παράμετροι τιμωρίας τότε προστίθενται οι αντίστοιχοι περιορισμοί. Βέβαια, για την παροχή περισσότερων βαθμών ελευθερίας προτιμάται και προτείνεται η χρήση των ποινών ώστε να διευκολυνθεί και να επιταχυνθεί η διαδικασία εύρεσης εφικτών λύσεων.

## 0.6 Μαθηματική Μοντελοποίηση

Οι προτεινόμενες παραλλαγές του προβλήματος *RCPSP* μπορούν να μοντελοποιηθούν μαθηματικώς εισάγοντας τις δυαδικές μεταβλητές  $x_{imqt}$  που ορίζονται ως εξής:

$$x_{imqt} = \begin{cases} 1, & \text{if the segment } p_{imq} \text{ of } i \text{ in mode } m \text{ starts at } t \\ 0, & \text{otherwise} \end{cases}. \quad (0.16)$$

Η μαθηματική μοντελοποίηση που παρουσιάζεται στις εξισώσεις (0.17)-(0.25), είναι μια επέκταση του μοντέλου που αρχικά παρουσιάστηκε από τους Πριτσκερ et al. (1969) ώστε να συμπεριλαμβάνει διακοπτόμενες δραστηριότητες, πολλαπλούς τρόπους εκτέλεσης δραστηριοτήτων, γενικευμένες σχέσεις προτεραιότητας και μεταβλητή ζήτηση και απαίτηση σε πόρους. Επιπλέον, δανείζεται στοιχεία και από τις μοντελοποιήσεις των *DeReyck* (1999) και *Hartmann* (2013) για τα προβλήματα *MRCPSP – GPR* και *RCPSP/t* αντίστοιχα.

$$\min f(x_{imqt}), \quad (0.17)$$

υπό τους περιορισμούς:

$$\sum_{m \in M_i} \sum_{t=0}^{T-1} \sum_{q=0}^{z_{im}} x_{imqt} = 1, \quad (0.18)$$

$$\forall i = 0, 1, \dots, n+1,$$

$$\left( \sum_{m \in M_i} \sum_{t=0}^{T-1} tx_{im0t} \right) + \delta_{imjn} \leq \sum_{n \in M_j} \sum_{t=0}^{T-1} tx_{jn0t}, \quad (0.19)$$

$$\forall (i, j) \in A,$$

$$\sum_{i=1}^n \sum_{m \in M_i} \sum_{q=0}^{z_{im}} r_{imk}^p x_{imqt} \leq \alpha_{kt}^p, \quad (0.20)$$

$$\forall k \in R^p, \forall t = 0, 1, \dots, T-1,$$

$$\sum_{t=0}^{l_{(x+1)}-1} \sum_{i=1}^n \sum_{m=1}^{m_{i\lambda}} \sum_{q=0}^{z_{im}} r_{iml}^v x_{imqt} \leq \alpha_{lx}^v, \quad (0.21)$$

$$\forall l \in R^v, \forall x = 0, \dots, X_l - 1,$$

$$x_{0000} = 1, \quad (0.22)$$

$$\sum_{t=0}^{T-1} tx_{imz_{im}t} \leq \sum_{t=0}^{T-1} tx_{(n+1)00t}, \quad (0.23)$$

$$\forall i = 0, 1, \dots, n,$$

$$\sum_{t=0}^{T-1} tx_{im(q-1)t} + d_{imq} \leq \sum_{t=0}^{T-1} tx_{imqt}, \quad (0.24)$$

$$\forall i = 0, 1, \dots, n, \forall m \in M_i, \forall q = 1, \dots, z_{im},$$

$$\begin{aligned} x_{imqt} &\in \{0, 1\}, \\ \forall i = 0, 1, \dots, n+1, \forall m \in M_i, \forall t = 0, 1, \dots, T-1. \end{aligned} \quad (0.25)$$

Η αντικειμενική συνάρτηση (0.17) ελαχιστοποιεί τους επιλεχθέντες αντικειμενικούς στόχους, για παράδειγμα τον στόχο ελαχιστοποίησης της συνολικής διάρκειας του έργου γράφεται ως:

$$\min \sum_{t=0}^{T-1} tx_{(n+1)00t}.$$

Οι περιορισμοί (0.18) διασφαλίζουν ότι κάθε δραστηριότητα και κάθε τμήμα στο οποίο διασπάται, εκτελείται ακριβώς μιας φορά και με ένα συγκεκριμένο τρόπο εκτέλεσης.

Οι περιορισμοί (0.19) αφορούν τις σχέσεις προτεραιότητας με ελάχιστη και μέγιστη χρονική υστέρηση, όπου οι τιμές της μέγιστης ή ελάχιστης υστέρησης δίνονται από το  $\mathit{dimjn}$ .

Οι περιορισμοί στη διαθεσιμότητα ανανεώσιμων και μη ανανεώσιμων πόρων δίνονται στις εξισώσεις (0.20) και (0.21), αντιστοίχως.

Οι εξισώσεις (0.22) και (0.23) διασφαλίζουν ότι η πρώτη δραστηριότητας που προγραμματίζεται είναι η έναρξη και η τελευταία μετά την ολοκλήρωση όλων των άλλων είναι η δραστηριότητα τέλους.

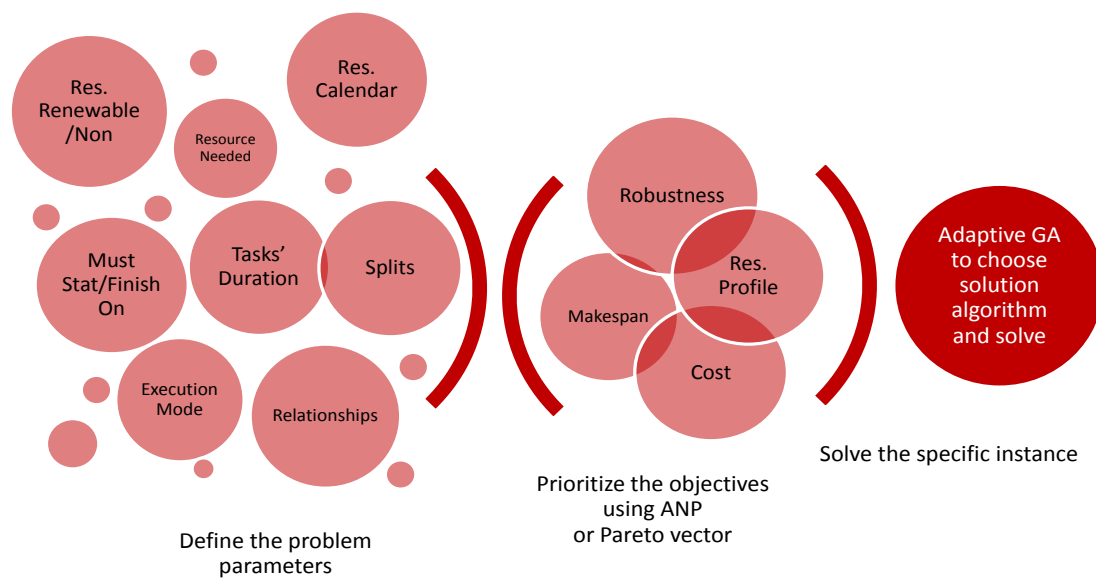
Οι περιορισμοί (0.24) εξασφαλίζουν ότι τα τμήματα κάθε δραστηριότητας θα εκτελεστούν με τη σωστή σειρά.

Ο περιορισμός (0.25) ορίζει ότι οι μεταβλητές μας είναι δυαδικές και λαμβάνουν τις τιμές 0 ή 1.

## 0.7 Προτεινόμενη Μέθοδος Επίλυσης

Η προτεινόμενη μέθοδος επίλυσης απαρτίζεται από τρεις φάσεις, όπως φαίνεται στην εικόνα 0.6. Αρχικά, το πρόβλημα αναλύεται ώστε να καθοριστούν τα δομικά στοιχεία του υπό εξέταση έργου, ποιες είναι οι δραστηριότητες, με ποιο τρόπο συσχετίζονται, αν είναι διακοπτόμενες και πώς ή όχι, ποιοι είναι οι εναλλακτικοί τρόποι εκτέλεσης, τι είδους και ποιες ποσότητες πόρων και σε ποιες χρονικές στιγμές εκτέλεσης της δραστηριότητας απαιτούνται και αντίστοιχα οι διαθέσιμότητες για τον χρονικό ορίζοντα του έργου. Το σύνολο αυτών των στοιχείων αποτελούν τις εισόδους της διαδικασίας επίλυσης του προβλήματος.

Η δεύτερη φάση αφορά τους αντικειμενικούς στόχους. Αποφασίζεται ποιοι είναι οι αντικειμενικοί στόχοι, επιλέγοντας ανάμεσα στις διαθέσιμες εναλλακτικές, κόστους, χρόνου, προφίλ πόρων, ευρωστίας χρονοπρογράμματος και ελαχιστοποίησης μέγιστης χρήσης. Με βάση αυτή την επιλογή, στη συνέχεια εάν έχουμε πάνω από έναν αντικειμενικό στόχο τότε η ομάδα λήψης απόφασης ή και ο διευθυντής του έργου επιλέγει αν θα γίνει ιεράρχηση τους με χρήση της *ANP* ή θα ακολουθηθεί η πιο παραδοσιακή πορεία της εύρεσης των κατά *Pareto* βέλτιστων λύσεων, όπως φαίνεται στην εικόνα 0.7.

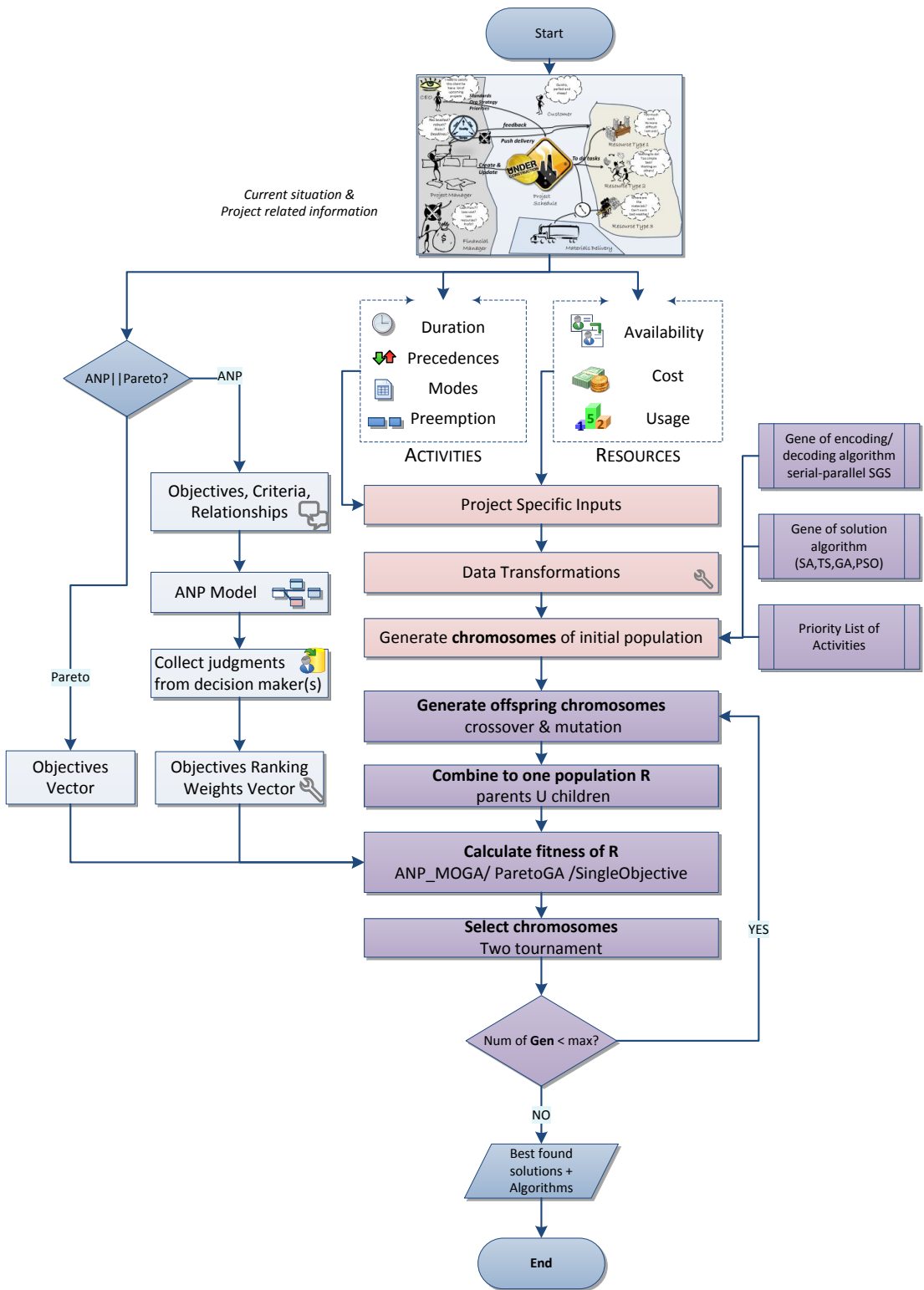


Σχήμα 0.6 Φάσεις προτεινόμενης μεθόδου επίλυσης

Στην τελευταία φάση πραγματοποιούνται όλες οι απαιτούμενες διεργασίες για την μετατροπή των δεδομένων εισόδων στην μορφή που απαιτεί ο αλγόριθμος επίλυσης για την εκτέλεση του και την παραγωγή των διάφορων εναλλακτικών λύσεων. Εφόσον ο αλγόριθμος - ενορχηστρωτής της όλης διαδικασίας είναι ένας υβριδικός γενετικός αλγόριθμος, έχουμε κωδικοποίηση των δραστηριοτήτων και των αντίστοιχων τρόπων εκτέλεσης σε ένα χρωμόσωμα, μαζί με γονίδια για τον προσδιορισμό του αλγόριθμου αποκωδικοποίησης και του καθ' αυτό αλγόριθμου επίλυσης (*SA, TS, PSO, GA*). Ο αρχικός πληθυσμός παράγεται τυχαία ενώ με χρήση τελεστών διασταύρωσης και μετάλλαξης προκύπτει ο πληθυσμός των παιδιών που προστίθεται στους γονείς. Ο ενοποιημένος πληθυσμός κατανέμεται με βάση το αν λύνουμε μονής, πολλών αντικειμενικών συναρτήσεων *Pareto* ή *ANP* πρόβλημα στην

αντίστοιχη διαδικασία και εν συνεχεία ελέγχεται το γονίδιο αλγόριθμου επίλυσης για την ανάθεση σε έναν από τους αλγόριθμους επίλυσης. Τα χρωμοσώματα τροποποιούνται μέσω αυτής της διαδικασίας και επιστρέφονται στον αλγόριθμο - ενορχηστρωτή της διαδικασίας για την ταξινόμηση τους με βάση την τιμή της συνάρτησης καταλληλότητας καθενός από αυτά και επιλογή των καλύτερων για να μεταβούν στην επόμενη γενιά. Η διαδικασία επαναλαμβάνεται για προκαθορισμένο πλήθος κύκλων ή ως το πέρας κάποιου μέγιστου χρονικού διαστήματος από την έναρξη.

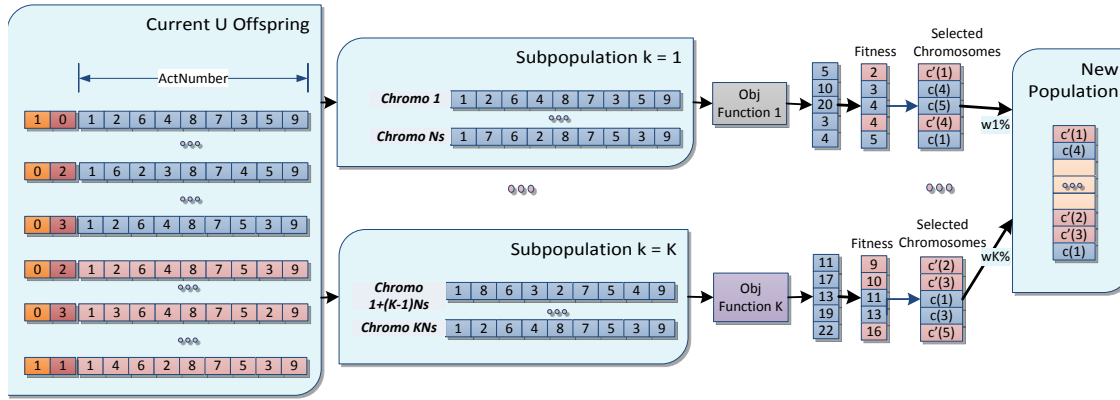
Η ροή των δεδομένων από τον καθορισμό των στοιχείων του έργου ως την παραγωγή του ζητούμενου συνόλου των εναλλακτικών συνοψίζεται στην εικόνα 0.7.



Σχήμα 0.7 Ροή δεδομένων



Ιδιαίτερο ενδιαφέρον παρουσιάζουν οι περιπτώσεις βελτιστοποίησης με πολλαπλούς αντικειμενικούς στόχους. Στην περίπτωση της ANP, η διαδικασία βελτιστοποίησης βασίζεται στον διαχωρισμό του δοθέντος αρχικού πληθυσμού σε υποπληθυσμούς.



Σχήμα 0.8 Βελτιστοποίηση πολλαπλών αντικειμενικών στόχων με ANP

Ο κάθε ένας υποπληθυσμός χρησιμοποιείται για την επιδίωξη του αντίστοιχου αντικειμενικού στόχου ανεξάρτητα από τους υπόλοιπους. Στο τέλος του κύκλου εργασιών επιλέγεται αριθμός λύσεων από τον κάθε υποπληθυσμό ανάλογα με τις βαρύτητες που είχαν δοθεί στους αντικειμενικούς στόχους μέσω της μεθόδου ANP, όπως φαίνεται στην εικόνα 0.8. Ο αντίστοιχος αλγόριθμος περιγράφεται ως εξής:

---

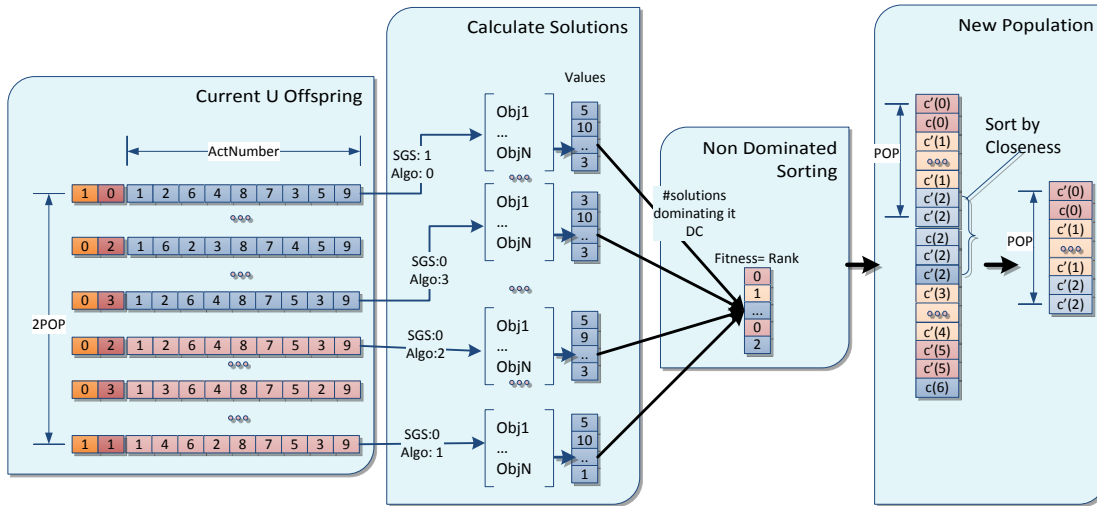
#### Algorithm 0.1: ANP

---

**input** :  $P_g$ , ANPweights= $[w_1, \dots, w_k]$ ,  $K$  = total number of objectives,  $N_s = 2POP/K$   
**output**:  $P_{g+1}$   
 // for each objective  $k$   
**for**  $k = 1 \dots K$  **do**  
   // for each chromosome  $i$   
   **for**  $i = 1 + (k-1)N_s \dots kN_s$  **do**  
     Fitness ( $i$ ) = ObjectiveFunction ( $k, i$ );  
   **end**  
    $P_{sub_k}$  = FormSubPopulation ( $P_g, 1 + (k-1)N_s \dots kN_s$ );  
**end**  
 Select from  $P_g$  by subpopulations;  
 $P_{g+1}$  = Form ( $w_1 \times P'_{sub_1}, \dots, w_k \times P'_{sub_k}$ );

---

Η δεύτερη δυνατότητα που δίνεται για πολυστοχική βελτιστοποίηση αφορά την εύρεση βέλτιστων κατά *Pareto* λύσεων. Χρησιμοποιείται μια αντικειμενική συνάρτηση για κάθε στόχο προς βελτιστοποίηση και οι τιμές σχηματίζουν το *Pareto* διάνυσμα. Η ιδέα βασίζεται στην ταξινόμηση του πληθυσμού με βάση έναν προκαθορισμένο κανόνα κυριαρχίας και την τιμή της αντίστοιχης συνάρτησης καταλληλότητας (εικόνα 0.9).



Σχήμα 0.9 Pareto GA

Εδώ πρέπει να σημειωθεί ότι η συνάρτηση καταλληλότητας δεν βασίζεται στις αντικειμενικές συναρτήσεις αλλά σχετίζεται άμεσα με το σε ποιο *Pareto* μέτωπο αντιστοιχεί, όπως φαίνεται αναλυτικά στον αντίστοιχο αλγόριθμο 2.

---

#### Algorithm 0.2: Pareto

---

```

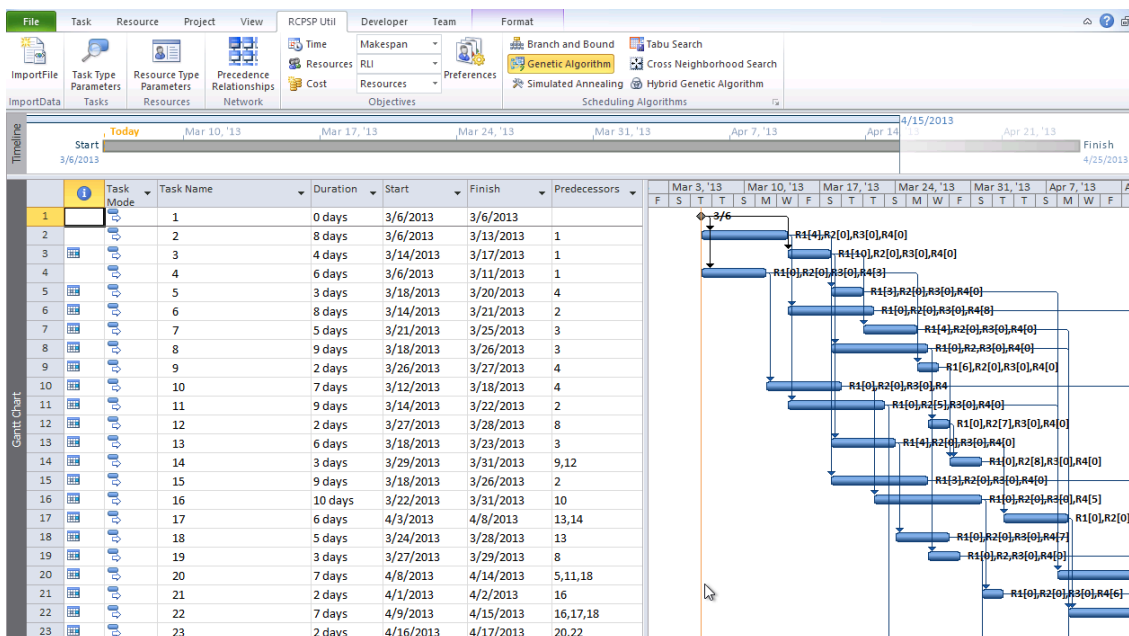
InputFininput set (input  $R_g$ ) set (output  $P_{g+1}$ ) // calculate non dominated
sets for population  $R_g$ 
Non-Dominated-Sort ( $R_g$ , out  $F_i$ , out  $rank[ ]$ );
// select chromosomes to pass to the next generation
foreach  $F_i$  do
  set  $sizeF = count(F_i)$ ;
  if  $emptySlots \geq sizeF$  then
    forall the solutions  $j \in F_i$  do
      Copy (chromo( $j$ ),  $P_{g+1}$ );
      emptySlots=emptySlots - sizeF;
    end
  else
    SortClosenessDesc ( $F_i$ );
    while  $emptySlots > 0$  do
      Take chromosome in desc order of closeness;
      Copy (chromo( $j$ ),  $P_{g+1}$ );
      emptySlots = emptySlots - 1;
       $j = j + 1$ ;
    end
  end
end
set emptySlots=POP;
end

```

---

## 0.8 Πειραματική Επαλήθευση Αποτελεσμάτων

Η προτεινόμενη μαθηματική μοντελοποίηση και η αντίστοιχη μέθοδος επίλυσης υλοποιήθηκε κάνοντας χρήση της γλώσσας προγραμματισμού *C#.NET*. Ο παραχθέν κώδικας παρέχει ένα διπλό περιβάλλον υλοποίησης των αλγορίθμων, αφενός έχουμε μια απλή εφαρμογή χωρίς γραφική διεπαφή για την εκτέλεση των αλγορίθμων με στόχο την βέλτιστη ταχύτητα και αποτελεσματικότητα και αφετέρου παράχθηκε ένα πλήρες περιβάλλον διεπαφής σαν επέκταση του γνωστού εργαλείου διαχείρισης έργων *Microsoft Project*. Στο εργαλείο αυτό, που από εδώ και στο εξής θα καλούμε *EMO – RCPSP*, παρέχονται στον τελικό χρήστη όλα εκείνα τα εργαλεία που απαιτούνται για τον λεπτομερή προσδιορισμό των χαρακτηριστικών και των δομικών στοιχείων ενός έργου αλλά και η επιλογή και παραμετροποίηση τόσο των αντικειμενικών στόχων και του πλήθους των παραγόμενων λύσεων όσο και των ίδιων των αλγορίθμων επίλυσης, όπως φαίνεται στην εικόνα 0.10.



Σχήμα 0.10 *EMO – RCPSP – MSPProject*

Δυο διακριτά πειράματα σχεδιάστηκαν και υλοποιήθηκαν:

- το πρώτο πείραμα αποσκοπεί στην επαλήθευση των προτεινόμενων αλγορίθμων συγκρίνοντας τα αποτελέσματα που δίνουν σε σχέση με τα αντίστοιχα αποτελέσματα σε σχετικά μικρά (15, 30 δραστηριοτήτων) προβλήματα που υπάρχουν δημοσιευμένα και είναι εκ των προτέρων γνωστά τα βέλτιστα αποτελέσματα. Επίσης, κάνοντας χρήση των ίδιων αλλά και μεγαλύτερου μεγέθους (120 δραστηριοτήτων) προβλημάτων γίνονται συγκρίσεις για τη διαπίστωση της αποτελεσματικότητας των προτεινόμενων μεθόδων. Τελικός στόχος είναι να αποδειχθεί ότι το προτεινόμενο ολιστικό μοντέλο και οι αντίστοιχοι αλγόριθμοι δίνουν τουλάχιστον το ίδιο καλές λύσεις και σε αντίστοιχους χρόνους με τους αναφερόμενους στην πρόσφατη βιβλιογραφία του χώρου δηλαδή ότι δεν οδηγεί σε απώλεια ποιότητας ή ταχύτητας επίλυσης των προβλημάτων.

- το δεύτερο πείραμα έχει περισσότερο διερευνητικό χαρακτήρα μιας και αφορά την πολυστοχική βελτιστοποίηση για την οποία δεν υπάρχουν διαθέσιμα στη βιβλιογραφία στοιχεία για να γίνει σύγκριση, οπότε γίνεται προσέγγιση των προβλημάτων βελτιστοποίησης του προηγούμενου πειράματος, αυτή τη φορά κάνοντας χρήση πολλαπλών αντικειμενικών στόχων και οι λύσεις ελέγχονται απλά ως προς την εφικτότητα τους και το βαθμό ικανοποίησης των τεθέντων στόχων αλλά και την απόκλιση τους από τα αντίστοιχα παραδείγματα με ένα αντικειμενικό στόχο.

Στον πίνακα 0.2, συνοψίζονται τα αποτελέσματα του 1ου πειράματος ενώ στον πίνακα 0.3 του 2ου πειράματος. Για το 1ο πείραμα έχουμε στην πρώτη στήλη το στιγμιότυπο που εκτελέστηκε, στη δεύτερη η ελάχιστη διάρκεια έργου, στην τρίτη στήλη η μέγιστη διάρκεια έργου, στη συνέχεια δίνεται η μέγιστη απόκλιση από το καλύτερο δημοσιευμένο αποτέλεσμα και η συχνότητα που το καλύτερο γνωστό αποτέλεσμα βρέθηκε κατά τις επαναλήψεις του πειράματος. Τα αποτελέσματα οδηγούν στο συμπέρασμα ότι ο προτεινόμενος αλγόριθμος δίνει το ίδιο καλά αποτελέσματα με τους καλύτερους εξειδικευμένους ανά τύπο προβλήματος αλγόριθμους της βιβλιογραφίας και μάλιστα έχει στις περισσότερες περιπτώσεις ποσοστό απόκλισης από αυτή την τιμή κάτω του 2%. Επομένως, ο στόχος του πειράματος, η απόδειξη ότι επιτυγχάνουμε το ίδιο καλά αποτελέσματα με τους υπάρχοντες αλγόριθμους, επιτεύχθηκε.

Πίνακας 0.2 Συγκριτικά αποτελέσματα για ελαχιστοποίηση διάρκειας έργου

Όνομα	Μέση Απόκλιση	Μέγιστη Απόκλιση	Βέλτιστο/ΥΒ)
R <sup>*</sup> ΠΣΠ Θ30	0.25%	3%	96.7%
R <sup>*</sup> ΠΣΠ Θ120	1.42%	8%	34.46%
ΠR <sup>*</sup> ΠΣΠ Θ30	0.12%	2.5%	98.7%
ΠR <sup>*</sup> ΠΣΠ Θ120	1.21%	5%	42.73%
MR <sup>*</sup> ΠΣΠ 15	0.23%	1%	98.9%
MR <sup>*</sup> ΠΣΠ 21	0.01%	1%	99.9%
MR <sup>*</sup> ΠΣΠ Θ10	0.01%	0%	99.9%
R <sup>*</sup> ΠΣΠ-τ Θ30	0.05%	1%	99.7%
R <sup>*</sup> ΠΣΠ-τ Θ120	0.22%	1.5%	99.5%
R <sup>*</sup> ΠΣΠμαξ Θ30	0.12%	1.8%	90.12%

Τα αποτελέσματα του 2ου πειράματος δείχνουν ότι η συνδυασμένη χρήση των δυο πολυστοχικών τρόπων βελτιστοποίησης δίνουν στο διευθυντή έργου μια πιο σφαιρική εικόνα όσο αφορά τις δυνατές εναλλακτικές λύσεις και τους διάφορους ισοδύναμους δρόμους που μπορεί να ακολουθήσει για τον προγραμματισμό του εκάστοτε έργου. Επιπρόσθετα, συγκρίνοντας τα αποτελέσματα της πολυστοχικής βελτιστοποίησης με την βελτιστοποίηση μονού αντικειμενικού στόχου, συχνά (57% των περιπτώσεων) η πολυστοχική αντιμετώπιση οδηγεί σε σημαντική βελτίωση των μη κυρίαρχων αντικειμενικών στόχων με συγκριτικά μικρή απώλεια στον βασικό στόχο (λιγότερο από 5%). Το ιδανικό θα ήταν η σύγκριση και αυτών των αποτελεσμάτων με αντίστοιχα της βιβλιογραφίας αλλά δεν υπάρχουν διαθέσιμα στοιχεία.

Πίνακας 0.3 Διπαράτιοι ρεσουλτς φορ μλτι-οβθεςτιε ινστανςες

Ινστανςε	Αλγοριτημ	Μακροσπαν	ΡΑΙ	δστ Ροβυστνεςς	
Θ301.1.1	Παρετο	43	124.10	4900	30
Θ301.1.1	ΑΝΠ	45	133.10	4500	32
Θ301.1.1	Σινγλε Οβθ.	43	124.10	4900	30
Θ301.2.2	Παρετο	47	176.18	5500	32
Θ301.2.2	ΑΝΠ	47	173.28	5600	30
Θ301.2.2	Σινγλε Οβθ.	47	176.18	5500	32
Θ301.3.6	Παρετο	47	153.02	5200	46
Θ301.3.6	ΑΝΠ	47	158.65	5300	48
Θ301.3.6	Σινγλε Οβθ.	47	161.65	5600	42
Θ301.4.7	Παρετο	62	185.10	6600	35
Θ301.4.7	ΑΝΠ	64	187.63	5800	28
Θ301.4.7	Σινγλε Οβθ.	62	185.09	6800	21
Θ3034.9.3	Παρετο	60	207.33	5400	40
Θ3034.9.3	ΑΝΠ	60	208.32	5400	40
Θ3034.9.3	Σινγλε Οβθ.	60	207.82	5300	37

## 0.9 Μελέτη Περίπτωσης

Η προτεινόμενη μοντελοποίηση και μέθοδος επίλυσης του προβλήματος χρονοπρογραμματισμού έργου με περιορισμένους πόρους χρησιμοποιήθηκε στην πράξη για την κτηματογράφηση συγκεκριμένων περιοχών της Ελλάδας. Συγκεκριμένα, η προτεινόμενη διαδικασία ακολουθήθηκε βήμα-βήμα από τον αρχικό προσδιορισμό του προβλήματος ως την τελική επιλογή της βέλτιστης εναλλακτικής λύσης για τον προγραμματισμό του έργου. Ο προσδιορισμός των δεδομένων του έργου, δραστηριότητες, συσχετίσεις, ανάγκες σε πόρους και διαθεσιμότητα αλλά και οι προτεραιότητες των αντικειμενικών στόχων και το είδος και πλήθος των ζητούμενων εναλλακτικών λύσεων, προέκυψαν κατόπιν σειράς συνεντεύξεων με το διευθυντή και την ομάδα έργου.

## 0.10 Συμπεράσματα - Σημεία Καινοτομίας

Συνοψίζοντας, η παρούσα Διατριβή παρουσιάζει τα ακόλουθα στοιχεία καινοτομίας:

- Ολιστικό μαθηματικό μοντέλο που ενσωματώνει όλες τις γνωστές επεκτάσεις και παραλλαγές του προβλήματος χρονοπρογραμματισμού έργων υπό περιορισμένους πόρους. Συγκεκριμένα καλύπτει τις περιπτώσεις, δραστηριοτήτων με πολλαπλούς τρόπους εκτέλεσης, γενικευμένες σχέσεις προτεραιότητας μεταξύ των δραστηριοτήτων, δυνατότητα διακοπής της εκτέλεσης συγκεκριμένων δραστηριοτήτων σε ένα ή περισσότερα σημεία, μεταβλητή διαθεσιμότητα πόρων αλλά και μεταβλητή ανάγκη σε πόρους ανά δραστηριότητα.
- Προσαρμοσμένο υβριδικό εξελικτικό αλγόριθμο που διαχειρίζεται την επιλογή τόσο του τρόπου αποκωδικοποίησης των λύσεων όσο και των συνδυασμών αλγορίθμων αναζήτησης που χρησιμοποιούνται σε κάθε γενιά για τον υπολογισμό των λύσεων με βάση την αποδοτικότητα τους στο συγκεκριμένο, υπό επίλυση πρόβλημα.
- Επίτευξη καλών αποτελεσμάτων βελτιστοποίησης, ίδια ή και καλύτερα σε σύγκριση με τα αντίστοιχα που αναφέρονται στη βιβλιογραφία.
- Ανάπτυξη πρόσθετου εργαλείου για το ΜΣ Προθεστ, το οποίο υλοποιεί τους προαναφερθέντες αλγορίθμους, με σκοπό την παροχή στον διευθυντή έργου μιας σειράς εργαλείων για τον καθορισμό των δεδομένων του έργου ( πολλαπλούς τρόπους εκτέλεσης, κυμαινόμενη απαίτηση σε πόρους, καθορισμός μη ανανεώσιμων πόρων, κ.α.), των στόχων της βελτιστοποίησης με τρόπο απλό και εύληκτο, αλλά και την παραγωγή εναλλακτικών λύσεων αντί μιας μοναδικής λύσης.
- Λαμβάνονται υπόψη συστημικοί παράγοντες που επηρεάζουν το χρονοπρογραμματισμό του έργου τόσο στη μοντελοποίηση του έργου όσο και στον τρόπο σχεδιασμού.

Εν κατακλείδι, απώτερος στόχος της Διατριβής είναι η αντιμετώπιση του προβλήματος του χρονοπρογραμματισμού έργων και κατά συνέπεια η κατά στοιχείο (μοδულάρ) μοντελοποίησή του, η οποία δίνει τη δυνατότητα στο διευθυντή έργου να συνδυάζει, προσθέτει, αφαιρεί χαρακτηριστικά, για να πετύχει όσο γίνεται πιο ρεαλιστική αναπαράσταση του έργου που πρέπει να προγραμματίσει, των συνθηκών που επικρατούν αλλά και του είδους των επιθυμούμενων λύσεων. Τελικός στόχος η παραγωγή λύσεων από τις οποίες ο διευθυντής έργου θα επιλέξει την καταλληλότερη, που ουσιαστικά αντικατοπτρίζει καλύτερα το πραγματικό πρόβλημα που πρέπει να λυθεί και ταιριάζει στις συνθήκες που επικρατούν και τους στόχους που ο ίδιος έχει θέσει.

# Chapter 1

## Overview

### 1.1 Introduction

Managing projects dates back thousands of years. The builders of the pyramids in Egypt and the Acropolis in Greece are often cited as some of the world's first project managers. They had no computers nor planning software to assist them, but they managed complex projects, using the simplest of tools. A project can be defined as a set of activities which have a defined start point and a defined end state and which pursue a defined goal and use a defined set of resources. The most common goal is to assign starting times to each activity in a way that all the precedence relationships among the activities are obeyed, there are no overallocations of resources and the project finishes as soon as possible.

Although this problem might seem quite simple, trying to model it in such a way as to cover all the different situations that are encountered in practice and provide an effective way to handle them, balancing between the complexity of the problem and the efficiency of the provided solutions, is not straightforward.

In this thesis a holistic approach is proposed for defining the so called resource constrained project scheduling problem (RCPSP). The aim is to give a conceptual formulation of the project scheduling problem as a whole, where all deterministic aspects that have been previously explored in the relevant literature are covered. Moreover, an appropriate mathematical formulation along with a concise solution process covering both the single and the multi-objective case, are presented. The final goal is to provide a way to model and solve project scheduling problems as they actually are, without compromises other than the assumption that the given inputs are realistic.

Based on this model an adaptive algorithm the moderator, handling single objective and multi-objective cases either with prioritisation of the objectives and/or pareto optimality, was proposed. The moderator algorithm has as its main role to regulate the process and select the best algorithm to be used based on the instance currently being used.

It was experimentally proven that the usage of the algorithm raises the accuracy of the results without harming the execution time. Therefore we have a reliable way for solving any scheduling problem having features spanning from the standard RCPSP to any combination of existing variations. This way the project managers have a way of modelling their project in a single step following a transparent process. We overcome, the raise of complexity and the infeasibilities by using penalty functions when relaxation of the constraints is needed. In the multi-objective case the algorithm is capable of providing multiple solution scenarios that are generated either based on the Pareto front or on a weighted approximation of it.

Summarising, we have a unified model, that is reliable and accommodates the needs of project scheduling in practice, keeping at the same time great flexibility on what kind of solutions and at what degree each objective should be pursued.

The next step on this research is to enhance it by adding a mechanism for automatically dealing with infeasibilities instead of interactively doing it. Further experimentation on the multi-objective side of the problem focusing on the comparison with the existing approaches it is expected to give valuable insight.

## 1.2 Motivation

Scheduling problems have been investigated since the late fifties, motivated by the need to improve and facilitate project management. Project scheduling is a complex problem that every project manager faces in the beginning of each project and the consequences of an ill designed schedule can seriously endanger the successful project execution and completion. Applications can be found in diverse industries such as construction management, software development, etc. In addition, project scheduling is very attractive for researchers, mainly those related to operational research, because the models in this area are rich and, hence, difficult to solve.

Project scheduling involves the development of a project base plan (baseline schedule) which specifies for each activity the precedence and resource feasible start and completion dates, the amounts of the various resource types that will be needed during each time period and as a result the corresponding budget required for the execution of the project (Brucker et al., 1999). The fundamental issue for relevant problems is to generate a schedule that is precedence and resource feasible, that is to fulfil the initially set precedence constraints and respects the available capacity of the resources involved. But, apart from this major issue, it is also desirable to come up with a schedule of minimal total project duration and cost, smooth profiles for the resource types used and increase robustness, in order to minimise the effect of possible perturbations in the duration of the activities and the resource availabilities during the execution. Project managers depending on the project and the situation at hand give more or less importance to each of the above objectives, therefore, a multidimensional approach is implicitly or explicitly used in practice (Viana and Pinho de Sousa, 2000). These different aspects, are often conflicting and all of them need to be taken into consideration as they play different roles in the schedule generation process based on the specific organisation and its priorities, the size and the budget of the project, the customer and other environmental parameters.

Although the project scheduling problem, was initially faced as a "hard" problem assuming that it is fully observable, governed by well-defined laws of behaviour and closed to the environment, this is not the case. Still, when trying to take into consideration all the parameters defining and affecting a good project schedule a very complex system emerges. Therefore, it is essential to try to define the project scheduling problem taking into consideration conflicts and uncertainties but in a level of abstraction that will keep it general and permit its modelling and solution.

The vast majority of the research efforts in project scheduling over the past years has concentrated on the development of procedures for the generation of an effective baseline schedule, often called pre-schedule, predictive schedule or proactive, assuming that the environment is deterministic and all the needed information exists and is accurate (Herroelen and Demeulemeester, 1996). The baseline schedule serves very important functions like the allocation of



starting times and resources as to optimise some measure of performance (e.g. project duration) (Aytug et al., 2005; Demirkol et al., 1998), planning external activities, such as material procurement, preventive maintenance and committing to external deadlines (Xu and Cheng, 2008), cash flow projections and even as a measure of the performance of the project team. Indeed, baseline schedule enables good scheduling and resource allocation decisions that in turn allow quoting competitive and reliable due dates (Herroelen and Leus, 2004).

The early work in the project scheduling field addressed scheduling problems with finish to start precedence constraints among activities assuming that sufficient resources are available to perform the activities. Later on, resources were taken into account leading to the so-called resource-constrained project scheduling problem (RCPSP). More recently, multiple optimisation objectives and variations and extensions of the initial problem, mainly related to additional constraints and types of precedence relations, have been investigated.

Due to the complexity of the problem and the difficulties encountered when solving even the simple RCPSP problem and much more when handling its various extensions and variations, even today, there is a lack of generic models that integrate all the different facets of a project that should be scheduled and provide a solution process. However, in practice projects often fail to fall precisely in a sole case of those studied in the literature. For example a project can have some tasks that are splittable but not all and at the same time some activities with variable resource demands and a few tasks with hard deadlines. In such a case, either some of the features should be omitted to fall in one of the existing problem types or a multi-phase approach, handling each situation separately, should be followed. Furthermore, there is the issue of whether a single or multi-objective approach should be followed, when even if it seems a straightforward decision, it is not, as managers used on "what-if" scenarios would prefer to have both options and evaluate the different results against the case at hand to work using just one or two objectives.

To fill in this research gap, a holistic model is proposed in order to provide a way to define all the desired characteristics, and provide a solution process that will generate project schedules adaptable to different project settings, organisational sizes and strategies and scalable according to the size and criticality of the undergoing project. Furthermore, a solution process that is simple and quick enough to permit immediate re-runs for the generation of alternative scenarios, is proposed to give the opportunity to the project manager (and/or the group of decision makers) responsible for the definition and final selection of the baseline schedule to have a satisfactory number of alternatives to choose from.

Therefore, a generic model for project scheduling is proposed so as to cover holistically the majority of existing variations and extensions of the RCPSP and give project managers a straightforward way for scheduling their projects. This model is conceptually and mathematically formulated to support further extension and be the basis of a single unified project scheduling model. Based on this model, an adaptable evolutionary solution algorithm is implemented to handle single and multi-objective instances of scheduling problems. This algorithm is used on the one hand to prove that existing variations of the RCPSP problem are efficiently handled by the generic model giving as good results as the best known solutions and on the other hand that actual projects can be modelled and feasibly solved using the proposed set of model and algorithm.

### 1.3 Research Objectives

The general research objective can be stated as:

*to formulate a holistic model, that is reliable and capable of accommodating the needs of project scheduling in practice, keeping at the same time extended flexibility on what kind of solutions and at what degree one or more objectives should be pursued.*

This research objective leads to the following specific objectives:

- holistic modelling of the project scheduling problem so as to cover the majority of cases encountered in practical situations,
- generation of a conceptual and mathematical model of the problem having as a basis the resource constrained project scheduling problem and including all the deterministic extensions and variations of this problem,
- propose a solution process able to appropriately handle any situation from small and every day projects to large scale problems with complex activity relations, splittable activities, multiple execution modes, time windows and variability on the resource requirements and availability.
- a solution algorithm capable of adapting itself and providing different ways of handling the specific instance of the problem at hand, based on:
  - the number of objectives to be pursued,
  - the type of the problem: simple, multi-mode, with generalised precedence constraints, minimal and maximal time lags, preemption or any mix of these features and
  - the number and type of solution scenarios that are desired: from single objective optimisation of a range of objectives to multi-objective optimisation using weighted sum and/or pareto-optimal solutions, or a mix of the above,
- experimental verification of the efficacy of the proposed algorithm by comparing the results of the best in class algorithms for each of the extensions and variations of the resource constrained project scheduling problem to those given by the proposed algorithm, having in mind that the goal is to get results at least as good as the best known results based on the recent literature.
- experimental verification of the usability of the proposed model and its accompanying process by following it for the scheduling of a real project and comparing the process and the results to those given without its usage.

### 1.4 Research Boundaries

The area of project scheduling is vast, however, the present study focuses on proactive scheduling of single projects where the activities and their attributes have already been deterministically defined and likewise the optimisation objectives to be pursued.

This study is about generating a number of schedules (solution scenarios) based on the activities, the different modes that each activity can be executed, the relationships among the activities and the related execution time windows on their execution time, the renewable and non renewable resource availability and demand, the duration, as well as the selected objectives. All the parameters beside the resource availability depend on the selected execution mode, therefore each combination of execution modes reflects a different set up of the same project. The provided solutions can either be the result of a single objective optimisation of

one of the four available optimisation objectives (duration, cost, resource profile, robustness) or the result of multi-objective optimisation, where either the weights of two or more objectives are taken into consideration during the optimisation process or the pareto-optimal solution of a vector consisting of the selected objectives is calculated. Furthermore, due to the fact that the project manager is provided with a set of solutions and not a single one, any mix of the above solution scenarios can be requested as to compare the different solutions and choose the one that best fits or even combine solutions.

Having defined, what this study is about the boundaries of this research should also be clearly defined. First of all, the aim is to provide proactive scheduling of the project and not reactive. This means that the result is a baseline schedule meant to act as guideline during the project execution and it is not expected that the project will be executed exactly as it was scheduled, or at least often this doesn't happen. The provided schedules are proactive, as opposed to reactive schedules that include mechanisms for reacting on changes usually related to the activities duration and resource availability (e.g activities taking less or more time to be executed or resources being available sooner or later than the expected or in different quantities). Nevertheless, a way of elevating the chances of successful reaction in cases of delay in the execution of some of the activities through the robustness objective, is provided. Schedules that are created by using some robustness measure are able to partially absorb unanticipated disruptions.

Aspects of project scheduling concerning issues related to the existence of multiple projects either contending or not the same resources, are out of the scope of this study. Even so, the limitations on the availability of the resources due to their usage in other projects that are being executed concurrently, are handled by the variability of the availability of the resources over time. More specifically, when there is a priori knowledge of the projects that will be executed in parallel and when the common resources will be needed then either this resource can be made unavailable or at least have limited availability for that period of time and therefore the new project's schedule will not create any conflicts.

Furthermore, in this work the durations of the activities are taken as deterministic and known in advance, although the difficulties and risks lurking in expressing the activities' duration in a strictly deterministic way are recognised. Even so, in small and medium projects or for specific activities whose duration could make a great difference on the final results it is advised to have multiple executions of the project scheduling algorithm with different duration values (e.g. min, max and most likely) so as to be able to have a more precise view of the situation.

It is known that project scheduling is a multi-facet problem affected by a plethora of systemic parameters that cannot be easily taken into consideration in a quantitative model. This study aims in giving a tool to support the project manager on scheduling the project by providing a number of alternative solution scenarios to select the one that best fits the situation and not just a unique solution that can act as panacea. It is expected that the inputs and objectives are set based on the current situation, taking into consideration all those uncountable parameters that are not part of the model but play a role on how the schedule should be. Even so, the provision of tools for ranking the objectives using quantitative and qualitative criteria (through an ANP model) and the ability to have mixed solution scenarios are valuable tools in handling the complexity of this problem.

## 1.5 Overall structure and contents of this Thesis

This thesis is organised as follows:

- In this chapter the motivation of this work, the research objectives and boundaries were defined.
- In chapter 2, an introduction to the research domain of this thesis is effectuated through the literature review of the last decades. The study is developed around three axes: project scheduling, multi-objective optimisation and multi-criteria decision making. The focus on the different aspects of existing RCPSP extensions and variations along with the most commonly used solution methods.
- Chapter 3 is about the research method that was followed, focusing on the mathematical modelling and algorithmic design concepts.
- In chapter 4, a holistic approach is followed in order to define the project scheduling problem. The analysis is expanded around three axes: "What is the context of the problem that we want to cope with?", "What do we want to achieve?" and "What inputs do we have?". The interrelations among the components of the system and the flows of influence are briefly presented. Soft Systems Methodology and System Dynamics are used in some extend to initially frame and define the issues constituting the problem. After that, the features of the desired solution are used to define the objectives that should be pursued during the solution process. Finally, the problem is defined in terms of available inputs and desired outputs.
- In chapter 5, based on the problem definition, first a conceptual formulation of the problem is analytically described and then its mathematical formulation as a binary linear programming problem is presented.
- In chapter 6, the solution process is described and its main components explained in detail. It is a three phase process. It begins with the project manager defining which objectives (one or more) should be pursued and if there is some kind of prioritisation of these objectives. Following, the inputs are transformed in order to simplify the search for a good feasible solution. The final phase consists in executing the proposed adaptive genetic algorithm, that iteratively leads to the selection of the most proper solution algorithm from a predefined set of algorithms along with the best solutions/schedules calculated in the given time frame.
- In chapter 7, the results of experiments concerning the validity of the proposed process are shown. This is achieved by comparing its results to the best known solutions of the major RCPSP problem types.
- In chapter 8, the preliminary design of a real project for the development of large scale spatial data infrastructure for terrestrial areas network is presented to illustrate the proposed approach. Based on the proposed model the project manager was interviewed about the constraints, objectives and their weighting and degrees of freedom. Having modelled the project and defined all the needed inputs, the proposed algorithm with three different settings: a) as a single objective, b) using the given weights for the requested objectives and c) looking for pareto-optimal schedules, was executed. Then the best schedules got by each method, were presented to the project manager and the results were analysed.
- Finally, in chapter 9, potential impact and significance of the conducted study, implications for researchers and practitioners and possible directions for further research on this subject, are discussed.

## Chapter 2

# Literature Review

### 2.1 Project Management

The field of project management has taken decisive steps forward in the past decades. In today's competitive environment it is crucial to deliver quality products on time and within budget. It is not surprising that project management has become a hot research topic.

Nowadays, the word 'project' is very often used by practitioners and it implies different things to different people depending on the context. It originates from the Latin verb *proicere*, "to throw something forward" meaning "something that comes before anything else happens" thus initially, it referred to planning (Herroelen et al., 1998). A well accepted definition of a project is given by the ISO where project is defined as a "unique process, consisting of a set of coordinated and controlled activities with start and finish dates, undertaken to achieve an objective, conforming to specific requirements including constraints of time, cost and resources" (21500, 2012). An alternative definition is provided by the Project Management Book of Knowledge where a project is defined as "a temporary endeavour undertaken to create a unique product, service, or result" (PMBOK, 2012). The notion of project can be formally defined (Tavares, 2002) as "any purposeful transformation leading a system,  $X$ , from an initial state,  $s$ , to a specific state,  $s_0$  and so  $s_0$  should represent the targets to be achieved. This means that the concept of project implies: the identification of the system,  $X$ , to be transformed, b) the description of the initial state,  $s$  and c) the description of the new state,  $s_0$ , that should represent the targets of the project".

Analysing the project's definition, its temporary nature is indicated by the existence of a definite beginning and end. The end is reached when the project's objectives have been achieved or when the project is terminated because its objectives will not or cannot be met, or the need for the project no longer exists. The project's duration is always finite and defined by the timespan between the project's start and end. Furthermore, the temporary nature of a project indicates that a concentrated use of resources is needed to carry out the project. The unique nature of projects expresses the fact that every project creates a specific product, service, or result that differentiates it from other products, services, or results (PMBOK, 2012). A successful project is a project that is finished on time, within the budget and according to the preset specifications. Summarising, each project is characterised by:

- A goal or objective: A definable end product, result or output that is typically defined in terms of cost, quality and timing of the output from the project activities.
- Uniqueness: A project is a one-at-a-time, not a repetitive undertaking.
- Temporary nature: Projects have a defined start and end.

- Uncertainty: Projects are planned before they are executed and therefore carry an element of risk.
- Life cycle: A project passes through a life cycle that consists of different phases from conceptual design, definition, planning and scheduling to execution and delivery of the results.

A project consists of a number of events (milestones) and activities or tasks that have to be performed in accordance with a set of precedence constraints. An event (milestone) refers to a stage of accomplishment of activities, associated with a certain point in time and has a zero duration and thus no resource requirements. Each activity has a duration and requires a certain amount of one or more types of resources in order to be executed.

An activity can be *preemptive* (splittable), when the activity can be interrupted during its execution and started later on, with or without additional cost or time penalty, or non preemptive when interruption is not allowed.

The *duration* of each activity can either be deterministic or stochastic. In the first case, the duration is a single constant value, usually calculated as estimate using the average time the activity should take, excluding uncontrollable contingencies or stochasticity. In very specific situations, which involve imprecision rather than uncertainty, activity duration can be expressed using fuzzy numbers.

An activity can either be *multi-mode*, that is executable in different discrete modes, with each mode having different duration, resource type and/or amount requirements or it can have a single duration and set of resource requirements and then it is called *single mode*.

Resources may be of different types, including financial resources, manpower, machinery, equipment, materials, energy, space, etc. Resources are classified as renewable, non renewable, doubly constrained and partially renewable.

*Renewable resources* are available on a period-by-period basis. Only the total amount of resource used within each period is constrained and the per period availability of resource type  $k$  is  $R_k^p$ . Typical examples of renewable resources, include manpower, machines, tools, equipment, space, etc. The per period renewable resource units of type  $k$  required by activity  $i$ ,  $r_{ik}^p$  may be a constant number for all the activity duration, a variable related to the activity's execution stage or it can be a stochastic variable.

*Nonrenewable resources* are available on a total project basis, with a limited consumption availability for the entire project. Typical examples of nonrenewable resources include money, raw materials and energy. The set of nonrenewable resources is denoted as  $R^v$ . The availability of nonrenewable resource type  $k$  is denoted as  $R_k^v$  and the required amount for the execution of activity  $i$  is  $r_{ik}^v$  and it can be deterministic, constant or variable, or stochastic.

*Doubly-constrained resources* are constrained per period as well as for the whole project. Capital with restricted period cash flow and limited total cash amount is a typical example. Man-hours per day in combination with a constraint on the total number of man-hours for the project is another. Doubly-constrained resources can be incorporated by a combination of a renewable and a nonrenewable resources (Blazewicz et al., 1986) thus usually are not handled separately.

*Partially (non)renewable resources* are resources whose availability is defined for a specific time interval (subset of periods). This resource type was introduced by researchers recently and is not used very often (Böttcher et al., 1996, Schirmer and Drexl, 1996) although it is a generalisation of the above resource types and can be used to define both renewable and non renewable resources using a single resource type.

A project can be represented as a project network, Gantt chart (Clark (1954)), track planning (Herroelen (1998)) and line of balance (Lumsden (1968)). A project network can be described as the graphical representation of events, activities and precedence relationships.

A project network is a graph  $G = (N, A)$ , consisting of a set of nodes  $N$  and a set of arcs  $A$ . There are two possible modes of representation of a project network: the activity-on-arc representation which uses the set of arcs  $A$  to represent the activities and the set of nodes  $N$  to represent events, and the activity-on-node representation which uses the set of nodes  $N$  to denote the activities or events and the set of arcs  $A$  to represent the precedence relations (Demeulemeester and Herroelen, 1992).

Having defined what is a project and its main components, we can pass to project management, that essentially is a set of processes aiming at a successful project accomplishment. More formally, project management: "involves planning, scheduling and control of project activities to achieve performance, cost and time objectives for a given scope of work, while using resources efficiently and effectively" (PMBOK, 2012). Planning calls for the definition of a listing of activities that must be performed along with requirements for the various types of resources and estimates for the duration and costs of the various activities. Scheduling is the laying out of the actual activities of the project in the time order in which they have to be performed. This way the actual resources needed at each stage in the project are calculated, along with the expected completion time of each of the activities. Finally, control focuses on the difference between the schedule and actual performance once the project has started (Lewis, 1998). All project management processes are mapped into ten Project Management Knowledge Areas (PMBOK, 2012):

- Scope management refers to the process of directing and controlling the entire scope of a project with respect to a specific goal.
- Quality management involves ensuring that the performance of a project conforms to the specifications of the project stakeholders and participants
- Schedule/time management involves the effective and efficient use of time to facilitate the execution of a project.
- Cost management deals with methods used to keep a project within its budget.
- Risk management is the process of identifying, analysing, and recognising the various risks and uncertainties that might affect a project.
- Human resources management involves the function of directing human resources throughout the life cycle of a project.
- Contract/procurement management involves the process of acquiring the necessary resources to successfully accomplish project goals.
- Communications management involves having the proper skills to communicate to the right people at the right time, using the proper organisation, routing and control of information.
- Project integration management includes the processes required to ensure that the various elements of the project are properly coordinated.
- Stakeholder management involves four processes: identifying stakeholders, planning stakeholder management, managing stakeholder engagement and controlling stakeholder engagement.

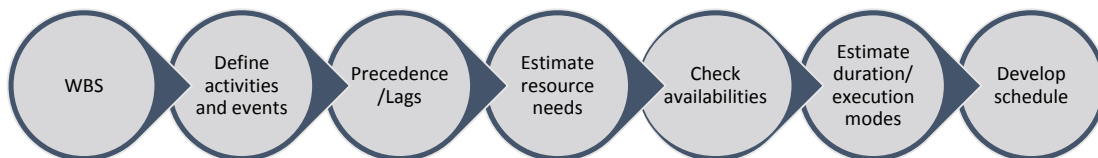
Herein the focus is given to schedule/time management that is often identified by practitioners as one of the most common causes of project failure (Duarte et al., 1995; Goldratt, 1997). Schedule/time management involves the effective and efficient use of time to facilitate the execution of a project and its effectiveness is reflected in the schedule performance as it is calculated by comparing the actual progress and/or cost of the project to the initial baseline schedule. Primary goal is to generate a feasible schedule, that is a schedule that respects the logic of the project network (e.g. the precedence relations and time lags) and the availability of resources and then optimise it under some objective either regular (monotone function of the starting/finishing times), or non-regular.

## 2.2 Project Scheduling

Scheduling problems have been investigated since the late fifties, motivated by the need to improve and facilitate new products, mainly military, delivery. Applications can be found in diverse industries such as construction engineering, software development, etc. Project scheduling is very attractive for researchers, because the models in this area are rich and, hence, difficult to solve.

Project scheduling involves the construction of a project base plan (baseline schedule) which specifies for each activity the precedence and resource feasible start and completion dates, the amounts of the various resource types that will be needed during each time period and as a result the corresponding budget required for the execution of the project (Brucker et al., 1999). The development of a realistic baseline schedule is critical to the successful accomplishment of a project. Fundamental issue is to generate a schedule that is not only precedence and resource feasible, having fulfilled the resource and precedence constraints initially set, but also robust, having minimised the effect of possible perturbations in the duration of the activities and the resource availabilities during the execution. The project activities are usually scheduled under one or more regular objectives (e.g. project duration) or non-regular objectives (e.g. net present value of the project).

The project scheduling process can be roughly summarised as follows: a) design of the work breakdown structure (WBS) and organisational breakdown structure (OBS) for the specific project, b) definition of activities and events, available amounts and types of resources, along with the estimation of the tasks duration, c) the definition of the precedence relationships and time lags and d) selection of start times for the activities in order to fulfil the constraints set in the previous steps, as shown in Figure 2.1.



**Fig. 2.1** Project scheduling process

The vast majority of the research efforts in project scheduling over the past years has concentrated on the development of procedures for the generation of an effective baseline schedule, often called pre-schedule, predictive schedule or proactive, assuming that the environment is deterministic and all the needed information exists and it is accurate (Vanhoucke et al., 2002). The baseline schedule aids very important processes like the allocation of starting times and resources for optimising performance measures (e.g. project duration) (Aytug et al., 2005; Varma et al., 2007), planning external activities such as material procurement, preventive maintenance, committing to external deadlines (Liang et al., 2012), and cash flow projections. Indeed, baseline schedules enable good scheduling and resource allocation decisions that in turn allow quoting competitive and reliable due dates (Herroelen and Leus, 2004). However, during execution, the project is subject to considerable uncertainty, which may lead to schedule disruptions due to activities that took more or less time than originally estimated, resources that became unavailable, material supplies that arrived behind schedule and changes in scope that cause addition of new activities, merging or splitting of activities,



abandonment of existing activities, etc. The recognition that uncertainty lies at the heart of project planning induced a number of research efforts in the field of project scheduling under uncertainty (Herroelen and Leus, 2005). One research track involves the development of baseline schedules that are protected as well as possible against schedule disruptions that may occur during project execution. On the other hand, another track focuses on reactive scheduling that is about the revision and re-optimisation of the baseline schedule after one or more unexpected events have occurred (Vieira et al., 2003).

Early work in the project scheduling field investigated scheduling situations with precedence constraints between activities assuming that sufficient resources are available to perform the activities. Following, scarce resources have been taken into account leading to so-called resource-constrained project scheduling problems (RCPS). More recently, multiple optimisation objectives and variations and extensions of the initial model have been investigated. Finally, the last few years the proactive - reactive schedule has started emerging. In the following sections, after presenting the most common way that project scheduling problems are classified, an overview of the RCPS problem will be given, including its definition, the most common mathematical formulations, existing variations and extensions along with popular, optimisation objectives and exact and heuristic solution approaches.

## 2.3 Classification of Project Scheduling problems

The growing research efforts in the area of project scheduling have led to a wide and growing variety of problem types, as shown in Figure 4.2. This motivated the introduction of a specific classification scheme. The extensive scheme commonly used in project scheduling (Demeulemeester and Herroelen, 1992) resembles the standard scheme for machine scheduling problems. In machine scheduling problems (Graham et al., 1979; Blazewicz et al., 1983) there is a three fields classification scheme,  $\alpha|\beta|\gamma$ , where the first field [U+FFFC] describes the machine environment, the second field is used to describe the task and resource characteristics and the third field [U+FFFC] denotes the optimality criterion (performance measure). In project scheduling, field  $\alpha$  is used to describe the resource characteristics. It contains at most four elements [U+FFFC]  $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ . The machining process (no machines, dedicated, parallel identical, uniform in parallel, unrelated, etc.) is specified by the parameter  $\alpha_1$ . However when dealing with a pure project scheduling problem the specification of structural resources is irrelevant. This is denoted by using the empty symbol  $\alpha_1 = \circ$ . The number of the resources of a project scheduling problem (other than machines) is specified by parameter  $\alpha_2 = \{\circ, 1, m\}$ . When no resources are available  $\alpha_2 = \circ$ , when only 1 resource type is available then  $\alpha_2 = 1$  and in case of multiple resource types then  $\alpha_2 = m$ , with  $m$  representing the number of available resource types. Parameter  $\alpha_3$  denotes the specific resource types used ( $\circ$  no resources, 1 renewable,  $T$  non renewable,  $1T$  both renewable and non renewable,  $v$  partially (non) renewable, etc.). Finally,  $\alpha_4$  [U+FFFC] describes the resource availability characteristics ( $\circ$  constant arbitrary amount,  $k$  constant amount of  $k$  units,  $v$  variable over time,  $\alpha$  stochastic,  $\tilde{\alpha}$  fuzzy, etc.).

The second field  $\beta$  specifies the activity characteristics of a project scheduling problem. It contains at most nine elements  $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9$ . Parameter  $\beta_1$ , [U+FFFC] indicates the possibility of activity preemption ( $\circ$ ,  $pmtn$  for preemption and resume at another time point,  $pmtn - rep$  for preemption and resume at the same point). The second parameter  $\beta_2$  concerns the precedence constraints and whether minimal and/or maximal time lags are allowed. The third parameter  $\beta_3$  refers to activities' deadlines. Parameter [U+FFFC]  $\beta_4$  de-

scribes the duration type of the project activities, whether discrete, continuous, stochastic or fuzzy. Parameter  $\beta_5$  describes the existence or not of activities and project deadlines. Parameter  $\beta_6$  denotes the nature of the resource requirements of the project activities (constant, variable, etc.). The type and number of possible execution modes for the project is denoted by parameter  $\beta_7$ . Parameter  $\beta_8$  is used to describe the financial implications of the project activities by associating the cash flows with the activities. Parameter  $\beta_9$  is used to denote change-over times (no change-over, sequence-dependent, stochastic or fuzzy) that is the time needed to pass from the execution of activity  $i$  using resource type  $r_x$  to the execution of activity  $j$  using the same or another resource type.

The third field  $\gamma$  is reserved to denote optimality criteria, which are either regular performance measures, involving functions which are nondecreasing in activity completion times (Erenguc et al., 2001), like minimisation of the project duration (makespan), of the project lateness or tardiness, of the sum of the direct and indirect project costs, etc. and non regular performance measures as the maximisation of the net present value of a project characterised by arbitrary cash flow values.

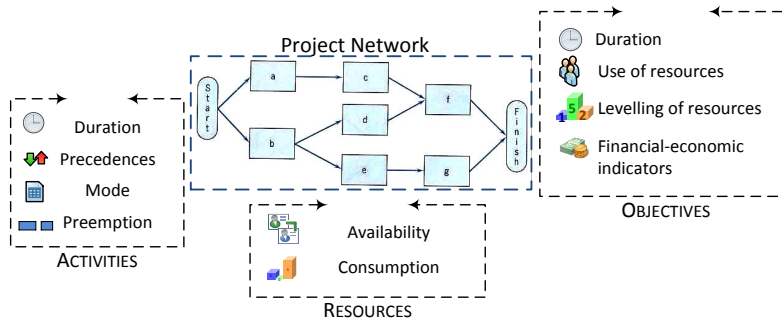


Fig. 2.2 Typology of Scheduling problems

## 2.4 The Resource Constrained Project Scheduling Problem

In the late 1950s the development of PERT (Program Evaluation Research Technique) and CPM (Critical Path Method) techniques allowed projects to be described using network diagrams where, either activities are represented by nodes, and the inter-relations between the activities are defined by the network structure (Activity on Node - AoN) or activities are represented by arcs (Activity on Ark - AoA). However, this way it is possible to deal only with the time aspect assuming that there are no resource restrictions. In practical situations it is uncommon to be able to follow a schedule generated using either PERT or CPM due to insufficient resource availability (Icmeli and Erenguc, 1994).

The first complete survey of this area was performed by (Davis, 1973) who categorised the resource allocation problems into three types: time/cost trade-off problems, problems in which resource demands are levelled and project scheduling problems with fixed resource limits. In addition, (Davis and Patterson, 1975) remarked on the strong similarities that exist between project scheduling problems and job-shop sequencing and assembly-line balancing problems. The correspondence between project scheduling and assembly-line balancing problems was summarised by (Icmeli and Erenguc, 1996c) as in Table 2.1:

**Table 2.1** Job-shop sequencing and assembly-line balancing problem compared to project scheduling as referred by Icmeli (1996)

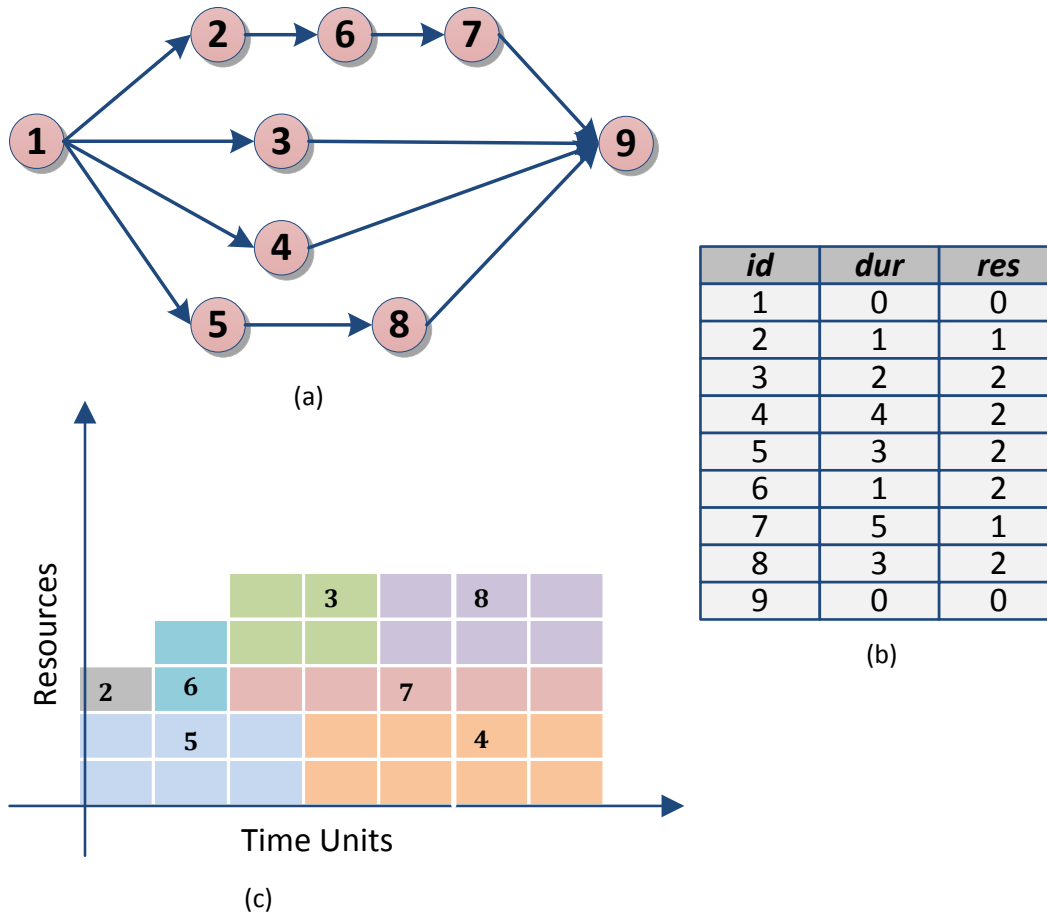
Assembly-line balancing	Project scheduling
Work elements	Activities
Work element times	Activity resource requirements
Work stations	Days
Cycle time	Maximum available units of resource

This particular problem of scheduling a project's activities under precedence and resource constraints is known as the "resource-constrained project scheduling problem" in the literature. It was firstly proposed by Kelley 1963 while solving the project scheduling problem with resource constraints in 1963. It is a very general scheduling problem which may be used to model many applications in practice like production process, school timetable, construction projects and it is a combinatorial optimisation NP-hard problem.

The problem is built upon three main axes: activities, resources and performance measures. A project consists of single activities, the execution of each activity requires resources from a predefined total available amount, so it is resource constrained and one or more performance measures are used to compare the generated schedules, that are the optimisation objectives (Liu et al., 2009). Therefore, Resource Constrained Scheduling consists of scheduling activities on scarce resources, with each activity requiring one or more resource types at a time, and each resource being available in the same quantity throughout the planning period (Leon and Balakrishnan, 1995).

The first optimisation objective used for the resource-constrained project scheduling problem, was the project's makespan, that is the project's total duration. The corresponding optimisation problem was defined as "finding precedence and resource feasible start times for all activities such that the makespan of the project is minimised" (Davis, 1973). Later on, the RCPSP was defined in more detail as the problem consisting in scheduling all the activities of a project so as to minimise its total duration subject to zero-lag feasible precedence of the PERT/CPM type and constant availability constraints on the required set of renewable resources (Herroelen et al., 1998).

Summarising, as illustrated in Figure 2.3, RCPSP involves the scheduling of project activities subject to finish-start precedence constraints with zero time lag and constant renewable resource constraints in order to minimize the project duration. Activities have a single execution mode with a fixed integer duration, preemption is not allowed and renewable resource requirements are constant throughout the duration of an activity. This problem is denoted as  $m, 1|cpm|C_{max}$  using the classification scheme presented in Section 2.3 (Herroelen and Demeulemeester, 1996).



**Fig. 2.3** RCPSP: (a) Activity on Node representation of the project's network, (b) duration and resource requirements of each activity and (c) resulting schedule when the resource's availability is five units

This problem deals with the optimum allocation of scarce resources over time and results in the definition of which activities are to be performed at which particular time. The allocation of scarce resources over time has been the subject of extensive research since the early days of operations research in the mid 1950s (Tavares, 2002). The result is a vast and not easy to digest literature and there is a considerable gap between scheduling theory and its application in practice. Practitioners often blame scheduling theoreticians for studying toy problems that oversimplify the reality. On the other hand theoreticians blame practitioners for their reluctance in applying the recent developments in practice and give valuable feedback (Demeulemeester et al., 1994). Despite the valid arguments about its simplified formulation, the resource-constrained project scheduling problem has become a standard problem for project scheduling in the literature.

### 2.4.1 Problem Definition

The resource constrained project scheduling problem (RCPSP) may be formulated as follows (Christofides et al., 1987; Demeulemeester et al., 1994):

- There is a single project consisting of  $n$  activities  $i = 1, \dots, n$  plus, a dummy source activity 0 representing the “project start” and a dummy sink activity  $n + 1$  representing the “project end”, both with zero duration and resource requirements.
- Each activity  $i$ , has a duration of  $d_i$  time units. Setup times are not taken into consideration separately but are included in the duration.
- There are two kinds of constraints, precedence and resource related.
- The activities should be processed in a specific order given by the precedence constraints, where each activity  $i$  should start after the completion of all its immediate predecessors. Precedence constraints are given by relations  $i \rightarrow j$ , where  $i \rightarrow j$  means that activity  $j$  cannot start before activity  $i$  is completed.
- When the structure of the project is represented by an activity-on-node network  $G = (V, A)$ , then the vertex set  $V = \{0, 1, \dots, n, n + 1\}$  contains all activities and the set of arcs  $A = \{(i, j) | i, j \in V; i \rightarrow j\}$  represents the precedence constraints. For each activity  $i$  we define the set of predecessors of activity  $i$  as  $Pred(i) := \{j | (j, i) \in A\}$
- Performing an activity requires resources, which have limited capacity. There is a set of  $K$  renewable resource types  $k = 1, \dots, r$ , and each resource type has a limited capacity  $R_k$  that is constantly available at any time.
- Each activity  $i$  in order to be executed, requires  $r_{ik}$  units of resource type  $k$ . The required resources are not consumed but used for the time period of the activities duration and then returned to the resource pool.
- The activities are assumed not preemptive, thus their processing cannot be stopped once it has been started.
- All data is assumed to be deterministic and known in advance.
- The objective is to determine starting times  $S_i$  for all the activities  $i = 1, \dots, n$  in such a way that:
  - at each time  $t$  the total resource demand is less than or equal to the resource availability of each resource  $k = 1, \dots, r$
  - the given precedence constraints are fulfilled so each activity should start after the completion of all its predecessors and
  - the makespan, that is the total project duration, which is the completion time of the dummy sink activity representing the “project end”, is minimised.
- The vector  $S = \overrightarrow{S_i}$  defines a schedule of the project, under the condition that preemption of activities is not allowed. A schedule  $S$  is called feasible if all resource and precedence constraints are fulfilled.

The RCPSP can be conceptually formulated (Christofides et al., 1987; Demeulemeester and Herroelen, 1992), as follows:

$$\min f_{n+1} \quad (2.1)$$

$$S_i + d_i \leq S_j \quad j = 1, \dots, n, \quad i \in Pred(j) \quad (2.2)$$

$$\sum_{j \in Act(t)} r_{jk} \leq R_k \quad Act(t) = \{j | j = 1, \dots, n \quad S_{j+1} \leq t \leq S_j\} \quad (2.3)$$

$$S_j \geq 0 \quad j = 0, \dots, n + 1 \quad (2.4)$$

The objective function of Equation 2.1 minimises the completion time of the project's end activity and thus the makespan of the project. Constraints defined in Equation 2.2 take into consideration the precedence relations, that are of "finish to start with no time lag" type. The constraints set in Equation 2.4 limits the total resource usage within each period to the available amount. By relaxing the resource constraints, set in Equation 2.4, the problem reduces to the CPM case and can be easily solved in polynomial time by forward recursion. Equations 2.1 to 2.4 do not give a mechanism to compute  $Act(t)$ , hence the problem with the above formulation cannot be solved using linear programming. To overcome this deficiency, the RCPSP has to be modelled with 0-1 variables as outlined in Pritsker et al. 1969.

Following is presented this 0-1 integer programming formulation of RCPSP. This formulation can be used to directly solve small instances of the problem as it requires the use of  $nT_{max}$  at most) binary decision variables and  $n + n(n-1)/2 + kT_{max}$  number of constraints that is  $\mathcal{O}(n^2) + kT_{max}$  restrictions (Maniezzo and Mingozzi, 1999).

$$\min \sum_{t=es_n}^{ls_n} t \cdot \xi_{nt} \quad (2.5)$$

$$s.t. \sum_{t=es_i}^{ls_i} \xi_{it} = 1 \quad (2.6)$$

$$\sum_{t=es_j}^{ls_j} t \cdot \xi_{jt} - \sum_{t=es_i}^{ls_i} t \cdot \xi_{it} \geq d_i \quad (i, j) \in A \quad (2.7)$$

$$\sum_{i \in Act(t)} r_{ik} \sum_{\tau=\sigma(t,i)}^t \xi_{i\tau} \leq R_k \quad t = 0, \dots, T_{max} \quad T_{max} = \sum_i d_i \quad (2.8)$$

$$\sigma(t, i) = \max(0, t - d_i + 1) \quad (2.9)$$

$$\xi_{it} \in \{0, 1\} \quad t = es_i, \dots, ls_i \quad (2.10)$$

In this formulation, the binary variable  $\xi_{it}$  gets the value 1 when the corresponding activity starts at the beginning of period  $t$ , assuming that time period  $t$  corresponds to the time interval  $[t, t + 1]$  and the value 0 otherwise. The time window  $[es_i, ls_i]$  of earliest and latest start times for each activity  $i$  is computed by performing a forward and backward recursion on the graph  $G$ , by setting  $es_1 = 0$  and  $ls_n = T_{max}$  where  $T_{max}$  is the time horizon that equals a feasible finish time of the project as it can be calculated by any heuristic method (Elmaghraby, 1977).

The objective function of Equation 2.5 minimises the project's makespan by minimising the start time of the project's end activity, as  $\xi_{nt}$  has value 1 when  $t = S_n$  thus  $\min \sum_{t=es_n}^{ls_n} t \cdot \xi_{nt} \Rightarrow \min (0 + 0 + \dots + S_n \cdot 1 + 0 + \dots + 0)$ . Constraints set in Equation 2.6 secure that each activity is executed exactly once, while constraints (2.7) take care of the standard precedence constraints, since if  $t_j = S_j$  and  $t_i = S_i$  then  $S_j - S_i \geq d_i$  for all activities  $i$  that are predecessors of activity  $j$ , these constraints are of "finish to start with no time lag" type. Constraints (2.8) reflect the resource availability restrictions, by calculating for each resource type the amount of resources being used by the in progress activities each time period and comparing it to the amount available. An activity is being executed in period  $t$  if it has been started in period  $q$  where  $0 \leq q \leq t - d_i + 1$ . Finally, constraints (2.10) define the binary decision variables.

Here should be noted that the size of the formulation is favourably affected by an increased amount of sequencing, by activities with relatively long duration and by close proximity of the scheduling horizon to the optimal project completion date (Pritsker et al., 1969).

Other formulations are presented by: a) Alvarez Valdes and Tamarit (1989) that is based on the definition of a set of all minimal resource incompatible sets where a resource incompatible set is a set of activities between which no precedence relation exists and that can be resource feasibly scheduled in parallel and it is called minimal if it is impossible to remove an activity and still have a resource incompatible set, this formulation requires  $\mathcal{O}(n^2)$  decision variables and  $\mathcal{O}(2^n)$  restrictions, b) Mingozzi (1998) that defines feasible subsets, which are subsets of activities between which no precedence relation is specified and that, if scheduled in parallel, do not violate the resource constraints and requires [U+FFFC]  $\mathcal{O}(2^{nT_{max}})$  binary decision variables and  $\mathcal{O}(n^2, nT_{max})$  restrictions and c) Klein (2000), where the binary decision variable  $x_{it}$  is 1 if activity  $i$  is in progress at  $t$  or has been in progress before  $t$  and 0 otherwise, there are needed  $\mathcal{O}(nT_{max})$  decision variables and  $\mathcal{O}(n^2T_{max})$  restrictions.

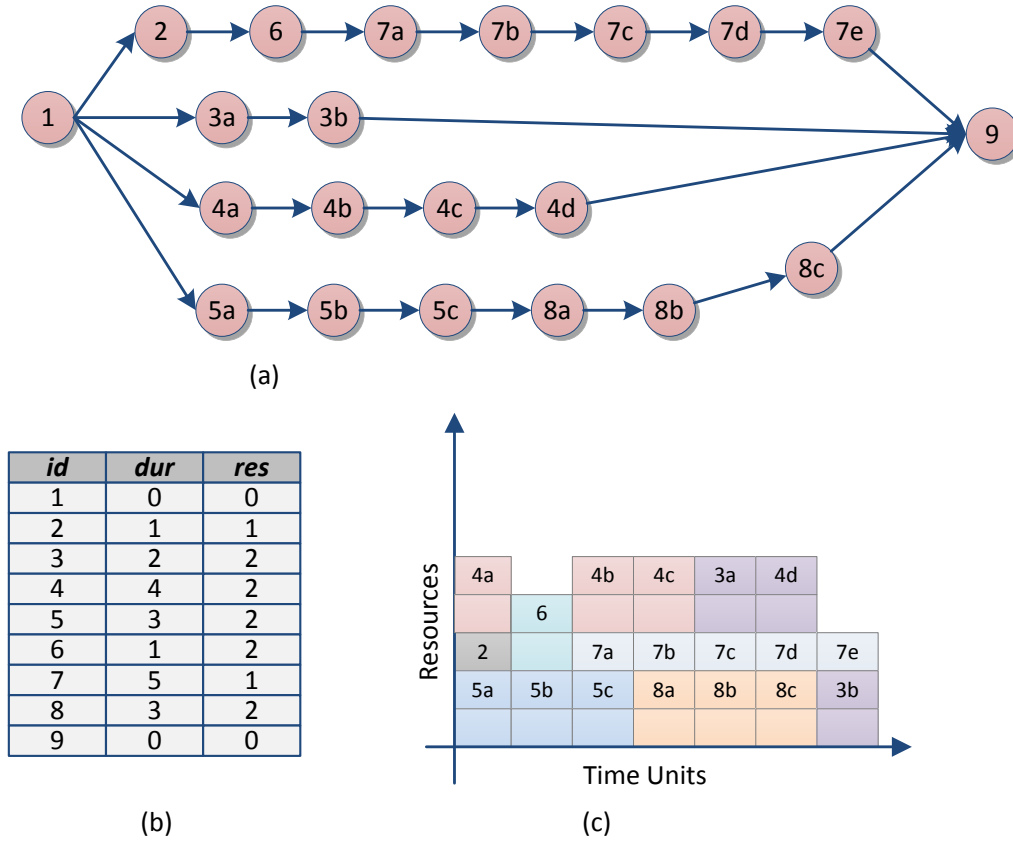
### 2.4.2 Variations and Extensions

While the RCPSP as given above is already a powerful model, it cannot cover all situations that occur in practice. Therefore, many researchers have developed more general project scheduling problems, often using the standard RCPSP as a starting point. Generalisations of the activity concept, precedence constraints and network characteristics as well as extensions of the resource concept and multiple objectives have been proposed the last few years (Hartmann and Briskorn, 2010).

The common point of all the variants of the RCPSP problem is the fact that each time one or two facets of the actual problem are included in the problem definition letting out the rest of them and conditions or axioms usually are applied to all the activities of the projects although this is not the case in the real world. In the approach proposed in this thesis all the following variants are integrated as alternative ways of describing each task composing the project and the problem and solution process are defined accordingly.

#### 2.4.2.1 Preemptive scheduling

The preemptive resource-constrained project scheduling problem (PRCPSP) allows activities to be preempted at any integer time instant and restarted later on at no additional cost (Demeulemeester and Herroelen, 1992; Bianco et al., 1998; Brucker et al., 1999; Debels and Vanhoucke, 2005). This problem is denoted as  $m, 1|pmtn, cpm|C_{max}$  using the classification scheme presented in Section 2.3 (Herroelen et al., 1999). In this case, the duration  $d_i$  of an activity  $i$  may be split in [U+FFFC]  $d_i$  duration units, as shown in Figure 2.4.



**Fig. 2.4** P-RCPSA: (a) Activity on Node representation of the project's network, (b) duration and resource requirements of each activity and (c) resulting schedule when the resource's availability is five units

Each duration unit  $j = 1, 2, \dots, d_i$  of activity  $i$  is assigned a finish time  $f_{ij}$ . [U+FFFC] To simplify the conceptual formulation, a variable  $f_{i,0}$  denoting the earliest time that an activity  $i$  can be finished is used. In the PRCPSA only relations of "finish - start" type with 0 time-lag are allowed. Therefore  $f_{i,0}$  equals to the latest finish time of all its predecessors. An activity  $i$  belongs to the set of activities in progress at time  $t$ ,  $Act(t)$  if and only if one of its duration units finishes at time  $t$ . Having in mind the above, the PRCPSA can be modelled conceptually in the following way:

$$\min f_{n,0} \quad (2.11)$$

$$s.t. f_{i,d_i} \leq f_{j,0} \quad \forall (i,j) \in A \quad (2.12)$$

$$f_{i,j-1} + 1 \leq f_{i,j} \quad \text{for } i = 1, \dots, n \text{ and } j = 1, \dots, d_i \quad (2.13)$$

$$f_{1,0} = 0 \quad (2.14)$$

$$\sum_{i \in Act(t)} r_{ik} \leq a_k \quad \text{for } k = 1, \dots, m \text{ and } t = 1, \dots, f_{n,0} \quad (2.15)$$

The objective function 2.11 minimises the project makespan by minimising the earliest start time of the dummy end activity which by assumption has a duration of 0. Equation 2.12



ensures that all precedence relations are satisfied by requiring the earliest start time of each activity  $j$  to be larger than the finish time of the last unit of duration of each predecessor  $i$ . In Equation 2.13 it is specified that the finish time for every unit of duration of an activity has to be at least one time unit larger than the finish time that the previous unit of duration has been assigned. The dummy start activity is assigned an earliest start time of 0 in Equation 2.14. Equation 2.15 ensures that the resource constraints will not be violated by requiring the total amount of used resources at each time instant to be less or equal to the available amount per resource type. Another, well known formulation of the RCPSP, is the binary formulation proposed by Kaplan (1988) where the binary variable  $x_{it}$  is defined to be 1 if  $i$  is in progress in period  $t$  and to be 0 otherwise.

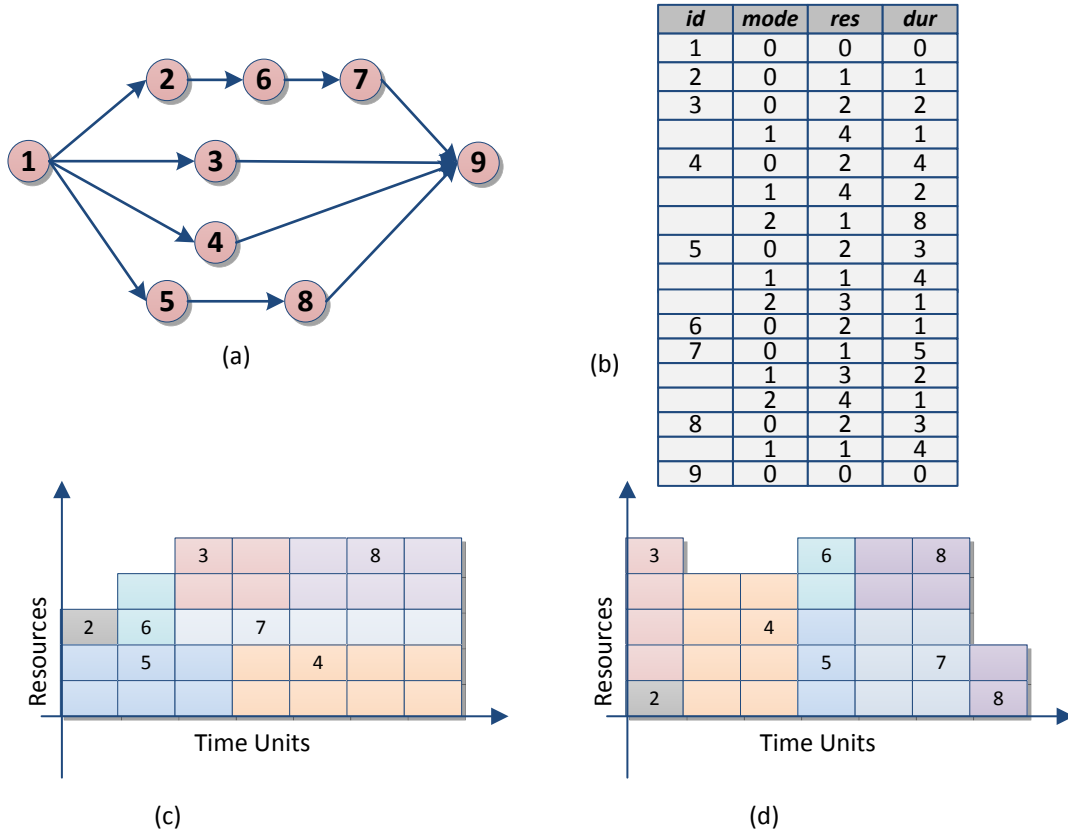
Based on the recent review of Hartmann 2010, following are presented some variations of preemptive scheduling, focusing in the diversity of problem settings currently available. Slowinski (1981) and Weglarz (1980) have studied the preemptive case when continuous processing times are assumed for the different activities and these activities can be restarted later on at no additional cost. Franck et al. (2001) propose a calendar concept for project scheduling where activities are allowed to be interrupted but only at specific points defined in the calendar and each activity  $j$  has a minimum processing time  $p_j$  during which it may not be interrupted. Debels and Vanhoucke (2005) extend the concept of preemption by a fast tracking option where the parts of a preempted activity can be carried out either in sequence or in parallel. Ballestin et al. (2007) consider a variant in which an activity may be interrupted at most  $m$  times. Damay et al. (2007) consider two types of activities, the first type contains non-preemptable activities and the second one preemptable at arbitrary points in time.

However, in practical situations not all activities are preemptable and often preemption is only possible in specific points of time and has a minimum, not unitary, duration per sub-activity. For example a task like a software module's development could be split at definite time instances corresponding to the sub-modules completion but it wouldn't be wise to split it in the middle of a complex function development.

#### 2.4.2.2 Multiple modes

The standard RCPSP assumes that an activity can only be executed in a single way which is determined by fixed duration and fixed resource requirements. The activity concept as given in the standard RCPSP has been extended by allowing several alternatives (modes) in which an activity can be performed. In the so called, multi-mode resource-constrained project scheduling problem (MRCPSPP), each activity can be performed in one out of several modes (Elmaghraby, 1964). Each mode reflects a feasible way to combine an alternative duration and different levels of resource requirements that allow accomplishing the underlying activity. The idea is based on the assumption that by using more resources of the same type or more efficient types of resources it is possible to get shorter execution time.

The corresponding optimisation problem can be stated as: "given a set of interrelated activities (precedence relations) where each activity can be performed in one of several ways (modes) and each mode is characterised by a known duration and given resource requirements, when should each activity begin and which resource – duration mode should be adopted so as to optimise some managerial objective?" (Boctor, 1990).



**Fig. 2.5** MRCPSP: (a) Activity on Node representation of the project's network, (b) duration and resource requirements of each activity, (c) resulting schedule when the resource's availability is five units and mode assignment  $\{0,0,0,0,0,0,0,0,0\}$  and (d) resulting schedule when the resource's availability is five units and mode assignment  $\{0,0,1,1,2,0,1,0,0\}$

In MRCPSP, as illustrated in Figure 2.5, each activity can be performed in one out of a set of prescribed ways, called modes, with mode specific duration and resource requirements. A mode represents a way of combining different resources and/or levels of resource requests.  $M_i$  denotes the number of distinct modes of activity  $i$ . The duration of activity  $i$  being performed in mode  $m_i$ ,  $1 < m_i < M_i$ , is given by  $d_{im_i}$ . Once an activity is started in one of its modes, it is not allowed to be interrupted or switch mode. Following (Patterson et al., 1989), renewable, non renewable and doubly constrained resources are distinguished. While renewable resources have a limited per-period availability, nonrenewable resources are limited for the entire project, and doubly constrained resources are limited both for each period and for the whole project. However, since the doubly constrained resources can be represented by a pair of, one renewable and one non renewable resource type, we do not consider them explicitly.

The set of renewable resources is referred to as  $R^p$ . For each renewable resource  $k \in R^p$  the per period availability is constant and given by  $\alpha_k^p$ . For nonrenewable resources, the availability within the entire project is limited. The set of nonrenewable resources is denoted as  $R^v$ . For each nonrenewable resource  $l \in R^v$  the overall consumption for the entire project is limited by  $\alpha_l^v$ . Each activity  $i$  in mode  $m_i$  requires the consumption of  $r_{im_i,k}^p$  renewable

resources of resource type  $k \in R^p$  and  $r_{im_i l}^v$  non renewable resources of resource type  $l \in R^v$ . The objective of the MRCPSP problem is to find a makespan minimal schedule that determines: a) timing of activities and b) assignment of modes, such that the schedule is feasible with respect to the precedence and resource constraints. The MRCPSP can be conceptually formulated as (Hartmann, 2001):

$$\min s_{n+1} \quad (2.16)$$

$$s.t. \quad s_i + d_{im_i} \leq s_j \quad \forall i \in Pred(j) \quad (2.17)$$

$$\sum_{i \in Act(t)} r_{im_i k}^p \leq \alpha_k^p \quad \forall k \in R^p, \forall m_i \in M_i \quad (2.18)$$

$$\sum_{\forall i} r_{im_i l}^v \leq \alpha_l^v \quad \forall l \in R^v, \forall m_i \in M_i \quad (2.19)$$

$$s_0 = 0 \quad (2.20)$$

$$s_i \in int^+ \forall i \quad (2.21)$$

Each activity  $i$  has to be performed in exactly one mode  $m_i$ . The objective function 2.16 minimises the total makespan of the project. Constraints set in Equation 2.17 are used to take the finish-start precedence relations with a minimal time lag of zero, into account. Constraints (2.18) and (2.19) concern the renewable and non renewable resource limitations, respectively. Equation 2.20 sets the project start at time instance zero and Equation 2.21 ensures that the activity start times get non negative integer values. A mathematical programming formulation for this model has been developed by Talbot (1982).

Before starting any solution process for the MRCPSP a procedure introduced by Sprecher et al. (1997) to reduce the volume of the data and speed up the execution of the solution algorithm is used to simplify the given inputs. More specifically, this procedure excludes modes which are inefficient or non-executable and resources which are redundant. A mode  $m_i$  is called inefficient if there is another mode  $m'_i$  of the same activity with the same or smaller duration,  $d_{im'_i} \leq d_{im_i}$  and no more requirements both for renewable,  $r_{im'_i k}^p \leq r_{im_i k}^p$  and non renewable,  $r_{im'_i l}^v \leq r_{im_i l}^v$  resources.

$$r_{im_i k}^p > \alpha_k^p \quad (2.22)$$

$$\sum_{j \neq i} \min(r_{jm_j l}^v) + r_{im_i l}^v > \alpha_l^v \quad (2.23)$$

A mode  $m_i$  is called non executable if its execution would violate either a renewable (2.23) or a non renewable (2.22) resource constraint.

$$\sum_j \max(r_{jm_j l}^v) \leq \alpha_l^v \quad (2.24)$$

A non renewable resource  $r_l^v$  is called redundant if the sum of the maximal requests for that nonrenewable resource can not exceed its availability (2.24). Excluding these modes and/or resources does not affect the set of feasible or optimal schedules.

Summarising, the MRCPSP includes time/resource and resource/resource trade-offs, multiple renewable, nonrenewable and doubly-constrained resources. In the basic problem setting activities have to be scheduled in one of their possible execution modes subject to renewable and nonrenewable resources. Under the minimum makespan objective the general problem can be denoted as  $m, 1T|cpm, disc, mu|C_{max}$  for projects with finish-start precedence con-

straints with zero time lag (Herroelen et al., 1997). It is a strong NP-hard problem and in the case of at least two non renewable resources, the problem of finding a feasible solution is already NP-complete, as was demonstrated by Kolisch and Drexel (1997).

### 2.4.2.3 Generalized temporal constraints

In the classical RCPSP, to start executing an activity all its immediate predecessors should have been finished. This precedence concept can be extended by considering generalised precedence relations (GPRs) with minimal and maximal time lags that can be used to define release dates and deadlines. There are four types of GPRs: start-start (*SS*), start-finish (*SF*), finish-start (*FS*) and finish-finish (*FF*) precedence constraints. Minimal time lags in a *FS* relation introduce a time period  $t$  between the finish time of activity  $i$  and the start time of activity  $j$ . Allowing negative minimal time lags implies that the corresponding activities may overlap. Similarly maximal time lags in a *FS* relation, introduce a maximum time period  $t$  between the finish time of activity  $i$  and the starting time of activity  $j$ . A release date is a minimal finish to start time lag between the dummy source and the under question activity  $j$  and a deadline is a maximal finish to finish time lag between the dummy source activity and activity  $j$ . GPRs are often useful in practice, for instance in cases where activities require fixed or simultaneous starting or completion times, non-delay execution, mandatory overlaps with other activities, time-varying resource requirements and deadlines ((De Reyck and Herroelen, 1999)).

The resource-constrained project scheduling problem with generalised precedence relations is often denoted as RCPSP-GPR or RCPSP/max and extends the standard RCPSP problem  $m, 1|cpm|C_{max}$  by allowing start-start, finish-start, start-finish and finish-finish precedence constraints with both minimal and maximal time lags. This extension can be denoted as  $m, 1|gpr|C_{max}$ . Furthermore, the use of minimal and maximal time lags allows modelling of activity deadlines as well as variable resource requirements and availabilities. Therefore, generalised precedence constraints can lead to a very general resource constraint scheduling problem setting denoted as  $m, 1, v\alpha|gpr, \rho_j, \delta_j, vr|C_{max}$  (Herroelen et al., 1999).

Generalised precedence relations with minimal and maximal time-lags between two activities  $i$  and  $j$  have the form:

$$s_i + SS_{ij}^{min} \leq s_j \leq s_i + SS_{ij}^{max} \quad (2.25)$$

$$f_i + FS_{ij}^{min} \leq s_j \leq f_i + FS_{ij}^{max} \quad (2.26)$$

$$s_i + SF_{ij}^{min} \leq f_j \leq s_i + SF_{ij}^{max} \quad (2.27)$$

$$f_i + FF_{ij}^{min} \leq f_j \leq f_i + FF_{ij}^{max} \quad (2.28)$$

where in the Equations [U+FFFC] 2.25 to 2.28,  $SS_{ij}^{min}$  represents a minimum time-lag between the start time  $s_i$  of activity  $i$  and the start time  $s_j$  of activity  $j$ , likewise  $SF_{ij}^{min}$  denotes a minimum time-lag between the start time  $s_i$  of activity  $i$  and the finish time  $f_j$  [U+FFFC] of activity  $j$ , similar definitions apply for  $SF_{ij}^{min}$ ,  $FF_{ij}^{min}$ ,  $SS_{ij}^{max}$ , etc. A common graphical representation of a network with time lags, is shown in Figure 2.6. A small rectangle is used to represent each activity. The left (right) side denotes the activity's start (completion). The activity's id and duration are written in the rectangle. Time lags are represented by arrows between the associated sides of the rectangles and the values are set in parenthesis ( $lag^{min}, lag^{max}$ ).

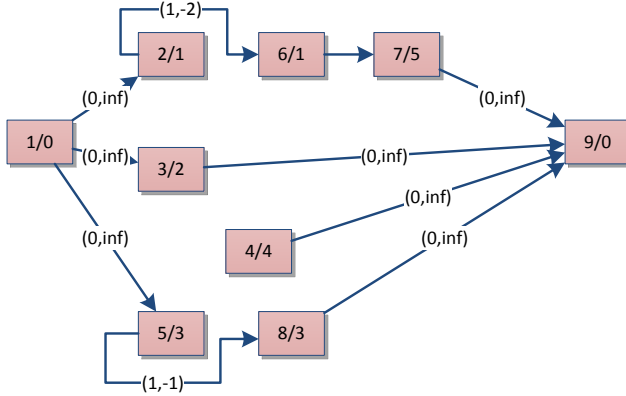


Fig. 2.6 Example digraph with time lags

The GPRs can be represented in standardised form by transforming all of them to the same, arbitrary selected, form:

$$s_j - s_i \geq \delta_{ij} \quad \forall (i, j) \in E \quad (2.29)$$

In Equation 2.29, that represents the so-called temporal constraints where all the generalised precedence constraints have been transformed to SS, we define  $\delta_{ij}$  to be the time-lag required between activities  $i$  and  $j$  and  $E$  the set of pairs of activities  $(i, j)$  with precedence relationships. A schedule that satisfies the temporal constraints of type 2.29 is termed time-feasible. The rules introduced by Bartusch et al. (1988) are used to represent all the different kinds of time lags in the standardised form:

$$s_i + SS_{ij}^{min} \leq s_j \rightarrow s_i + \delta_{ij} \leq s_j \text{ with } \delta_{ij} = SS_{ij}^{min} \quad (2.30)$$

$$s_i + SS_{ij}^{max} \geq s_j \rightarrow s_j + \delta_{ji} \leq s_i \text{ with } \delta_{ji} = -SS_{ij}^{max} \quad (2.31)$$

$$s_i + SF_{ij}^{min} \leq f_j \rightarrow s_i + \delta_{ij} \leq s_j \text{ with } \delta_{ij} = SF_{ij}^{min} - d_j \quad (2.32)$$

$$s_i + SF_{ij}^{max} \geq f_j \rightarrow s_j + \delta_{ji} \leq s_i \text{ with } \delta_{ji} = -(SF_{ij}^{max} - d_j) \quad (2.33)$$

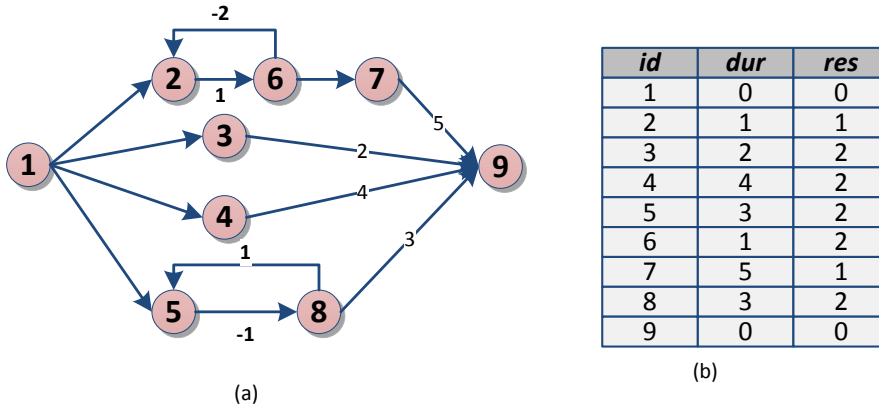
$$f_i + FS_{ij}^{min} \leq s_j \rightarrow s_i + \delta_{ij} \leq s_j \text{ with } \delta_{ij} = FS_{ij}^{min} + d_j \quad (2.34)$$

$$f_i + FS_{ij}^{max} \geq s_j \rightarrow s_j + \delta_{ji} \leq s_i \text{ with } \delta_{ji} = -(FS_{ij}^{max} + d_j) \quad (2.35)$$

$$f_i + FF_{ij}^{min} \leq f_j \rightarrow s_i + \delta_{ij} \leq s_j \text{ with } \delta_{ij} = FF_{ij}^{min} + d_i - d_j \quad (2.36)$$

$$f_i + FF_{ij}^{max} \geq f_j \rightarrow s_j + \delta_{ji} \leq s_i \text{ with } \delta_{ji} = -(FF_{ij}^{max} + d_i - d_j) \quad (2.37)$$

The interval  $[U+FFFC] [s_i + SS_{ij}^{min}, s_i + SS_{ij}^{max}]$  is called the time window of  $s_j [U+FFFC]$  relative to  $s_i$ ,  $SF$ ,  $FF$ ,  $FS$  time windows are defined analogously (Bartusch et al., 1988). This reduction makes possible the representation of the temporal constraints using a digraph  $G = (V, E)$  where each task is a vertex of the graph. An edge from  $i$  to  $j$  is formed if there are one or more constraints of the form  $s_i + \delta_{ij} \leq s_j$ . The maximum value of  $\delta_{ij}$  of all the constraints, between each two activities, is assigned as weight to the edge  $(i, j)$ , as illustrated in Figure 2.7.



**Fig. 2.7** RCPSP/max: (a) Representation of the project's network extended for GPRs using the  $G(V,E)$  digraph, (b) duration and resource requirements of each activity

It should be noted that project networks with GPRs when depicted as digraphs may contain cycles, a cycle is a directed path  $\langle i_s, i_k, i_l, \dots, i_t \rangle$  with  $s = t$ . The length of a cycle in a project digraph equals to the sum of all the lags associated with the corresponding path. It holds that if the project network does not contain any cycle of positive length then there are time-feasible schedules.

To ensure that the dummy start and finish activities correspond to the beginning and the completion of the project, it is required to ensure through the constraints that the dummy start activity will always be executed before every other activity and the dummy end activity will never terminate before any other activity.

The RCPSP/max problem can be conceptually formalised using the standard RCPSP equations 2.1 - 2.4 by replacing the precedence constraints of Equations 2.4 with Equation 2.29 and adding the following constraints:

$$s_i \geq 0 \quad (2.38)$$

$$s_0 = 0 \quad (2.39)$$

Equation 2.38 ensures that start times of activities are non negative numbers and Equation 2.39 sets the starting time of the dummy start activity to zero.

The initial idea about GPRs was introduced by Kerbosh and Schell (1975). Other studies include Elmaghraby (1977), Wiest (1967), Bartusch et al. (1988). In the last decade problems with minimal and maximal time lags have been discussed by a large number of authors like De Reyck et al. (1998), Dorndorf and Pesch (2000), Chassiakos and Sakellariopoulos (2005) etc.

#### 2.4.2.4 Resource requests varying with time

The activities in standard RCPSP require constant amounts of renewable resources, that is, the per-period request for a resource remains unchanged until the activity has been completed. This can be generalised by resource requests varying with time. This is formalised by denoting with  $r_{ikt}$  the request of activity  $i$  for renewable resource  $k$  in the  $t$  period of its processing time. This problem can be conceptually formalised using the standard RCPSP

equations 2.1 - 2.4 or 2.5-2.10 by slightly changing the way that the resource constraints are formalised, Equations 2.4 and 2.9 respectively, as follows:

$$\sum_{j \in Act(t)} r_{jkt} \leq R_k \quad Act(t) = \{j \mid j = 1, \dots, n \quad S_j + 1 \leq t \leq S_j\} \quad (2.40)$$

$$\sum_{i \in Act(t)} r_{ikt} \sum_{\tau = \sigma(t,i)}^t \xi_{i\tau} \leq R_k \quad t = 0, \dots, T_{max} \quad (2.41)$$

Note, that the only change is the replacement of the constant resource requirement  $r_k$  with the variable resource requirement  $r_{ikt}$  in Equations 2.40 and 2.41.

This extension has not yet received great attention in literature and there is a very restrict number of methods and applications for the case of varying resource requests. More specifically, (Hartmann, 2013) proposes an extension of RCPSP to include resource requests and availability varying with time. Cavalcante et al. (2001) had a similar problem setting handling activities with time-dependent resource requests for one renewable resource. Similarly, Drezet and Billaut (2008) deal resource requests having a minimum and a maximum value per period. Here should be noted that resource requests varying with time can be transformed into constant requests if maximal time lags are available by splitting the activities into parts with constant requests and adding a precedence constraint to order the sequence of execution of the parts (Bartusch et al., 1988).

#### 2.4.2.5 Generalised resource types and variable availability

The basic RCPSP features only one type of resources the renewable type that is available in each period with its full capacity. In project scheduling with multiple modes three different kinds of resources are considered (Sprecher et al., 1997): renewable, non-renewable and doubly constrained. Renewable are limited on a per period basis, Machines and manpower are examples of this resource category. Non-renewable have a limited capacity for the entire project. An example of this resource category is money if the budget of the project is limited. Doubly constrained are limited both for each period and for the whole project, an example of this resource category is money in the case that not only the budget of the project but also the per-period cash-flow is limited.

Less commonly used are the partially renewable resources, which generalise both renewable and non-renewable resources by defining different per period availabilities over a total availability for the whole project. Furthermore, continuous resource availability instead of discrete is needed in cases that the resources correspond to e.g. energy, raw materials like liquids, etc. Finally, dedicated resources refer to resources that can be assigned to one activity at a time, can be represented by renewable resources with 1 unit of per period availability.

In the RCPSP problem, resource availabilities have been assumed to be constant over time. This assumption is not very close to what actually happens in practical cases where changing availability of workers due to vacations, maternity leaves, sickness or varying equipment capacities due to maintenance or damage are on the everyday schedule. Bartusch et al. (1988) proves that a way to deal with resource capacities varying with time is to transform them into constant capacities by using minimal and maximal time lags. The constant capacity would be defined as the maximum of the time-dependent capacity over time, and for each time interval with a smaller capacity, an artificial activity is defined to reduce the capacity appropriately. Each artificial activity is then fixed to the desired time interval using a minimal and a maximal time lag.

### 2.4.2.6 Alternative Objectives

In addition to the parameters previously analysed there are also various alternative optimisation objectives, although the minimisation of the makespan is the most popular one. These objectives can be classified as: time based, robustness based, objectives for rescheduling and objectives based on renewable resources and non-renewable resources consumption and smoothness of profiles.

#### Time-based objectives

Besides the objective of minimising the makespan  $C_{max} := \max_{i=1}^n C_i$ , where  $C_i = S_i + d_i = f_i$  one may consider other objective functions depending on the completion times of the activities, like the total flow time,  $\sum_{i=1}^n C_i$  or more generally the weighted (total) flow time  $\sum_{i=1}^n w_i C_i$ . Other objective functions depend on due dates  $dd_i$ , which are associated with the activities, as follows:

$$L_{max} = \max_{i=1}^n L_i, \quad L_i = C_i - dd_i \quad (2.42)$$

$$\sum_{i=1}^n T_i, \quad T_i = \max\{0, C_i - dd_i\} \quad (2.43)$$

$$\sum_{i=1}^n U_i, \quad U_i = 0 \text{ if } C_i \leq dd_i \text{ otherwise } U_i = 1 \quad (2.44)$$

where Equation 2.42 describes the maximum lateness. Equation 2.43 describes the total tardiness, Ballestín et al. (2008), Kolisch (2000), and Viana and de Sousa (2000) consider the minimisation of the weighted version of this objective. Equation 2.44 refers to the number of late activities.

All the above objective functions are regular, thus monotone non-decreasing in the completion times. On the other hand, objectives like the maximum earliness (2.45) are an example for a non regular objective function (Vanhoucke et al., 2003; Lorenzoni et al., 2006).

$$\max_{i=1}^n E_i \text{ with } E_i := \max\{0, dd_i - C_i\} \quad (2.45)$$

Another non regular objective function that is quite commonly used (Kimms, 2001; Mika et al., 2005; Vanhoucke et al., 2008) deals with the net present value, where a so-called cash-flow  $c_i^F \in R$  is associated with each activity  $i$  and it is supposed to occur at the completion time  $C_i$  of  $i$ . The objective is to maximise the net present value (NPV) (2.46) given a discount rate  $\alpha \geq 0$ .

$$\sum_{i=1}^n c_i^F e^{-\alpha C_i} \quad (2.46)$$

#### Objectives based on resources

Resource based objectives occur in the area of resource investment (RIP) and resource levelling problems (RLP). In the RIP (Neumann and Zimmermann, 2000; Drexl and Kimms, 2001) the resource capacities  $R_k$  are not given but have to be determined as additional decision variables, given the per unit cost of each resource type  $c_k$  and a target value of resources



to be used  $Y_k$ , the objective is to find a schedule with makespan less than the given project deadline  $T$  and minimal resource cost.

In the RLP, the variation or the deviation of the resource usage over time is measured. In the deviation problems given a resource profile, where  $r_k^s(t)$  the resource usage of resource  $k$  the time period  $t \in 1, \dots, T$ , the goal can be (Davis, 1973; Viana and Pinho de Sousa, 2000; Neumann and Zimmermann, 2000) to minimise the deviation from a given resource usage level (2.47), the overload (2.48) or the squared deviation (2.49)).

$$\sum_{k=1}^r c_k \sum_{t=1}^T |r_k^s(t) - Y_k| \quad (2.47)$$

$$\sum_{k=1}^r c_k \sum_{t=1}^T \max(0, r_k^s(t) - Y_k) \quad (2.48)$$

$$\sum_{k=1}^r c_k \sum_{t=1}^T (r_k^s(t) - Y_k)^2 \quad (2.49)$$

On the other hand, in variation problems the resource usage should not substantially vary over time. This can be achieved by minimising the per period variation (2.50), the max variation (2.51), or the squared per period variation (2.52).

$$\sum_{k=1}^r c_k \sum_{t=1}^T |r_k^s(t) - r_k^s(t-1)| \quad (2.50)$$

$$\sum_{k=1}^r c_k \sum_{t=1}^T \max(0, r_k^s(t) - r_k^s(t-1)) \quad (2.51)$$

$$\sum_{k=1}^r c_k \sum_{t=1}^T (r_k^s(t) - r_k^s(t-1))^2 \quad (2.52)$$

#### Robustness-based objectives

During the execution of a project, delays may occur that could not have been foreseen when the schedule was determined. Therefore, a project manager might be interested in a robust schedule, that is a schedule in which a delay has only a limited effect. This approach is often referred to as proactive scheduling (Abbasi et al., 2006; Kobylanski and Kuchta, 2007).

#### Objectives for rescheduling

Rescheduling is necessary if the project is already in progress, but due to unexpected events (e.g., delays) the schedule that has been calculated before the start of the project is no longer valid. In such a situation, the problem's characteristics may have changed. For example, some activities may already be finished and can be ignored, other activities may be in progress and must be considered unchangeable and the resource availability may have changed and might even have switched from time-independent to time-dependent. In contrast to proactive scheduling which anticipates disruptions by building robust schedules, here the case is that some disruption has already occurred and a new schedule has to be determined with minimal

differentiation from the original/baseline schedule. This case is often referred to as reactive scheduling (Vanhoucke et al., 2002; Calhoun et al., 2002).

### 2.4.3 Complexity

When scheduling problems or more generally combinatorial optimisation problems are considered, an important issue is the complexity of the under question problem. Complexity theory provides a mathematical framework in which computational problems can be studied so that they can be classified as "easy" or "hard" (Karp, 1975; Graham et al., 1979; Garey and Johnson, 1979; Shmoys and Tardos, 1993). A computational problem can be viewed as a function  $f$  that maps each input  $x$  in some given domain to an output  $f(x)$  in some given range. Complexity theory is about the time required by an algorithm to compute  $f(x)$  as a function of the length of the encoding of the input  $x$ , denoted as  $|x|$ . The efficiency of an algorithm that computes  $f(x)$  on input  $x$  is measured by an upper bound  $T(n)$  on the number of steps that the algorithm takes on any input  $x$  with  $|x| = n$ . In most cases it is difficult to calculate the precise form of the  $T$  function, therefore its asymptotic order is used.  $T(n) = O(p(n))$  if there exist constants  $c > 0$  and a non negative integer  $n_0$  such that  $T(n) \leq c \cdot p(n) \quad \forall n \geq n_0$ . A problem [U+FFFC] is considered to be "easy" if there exists an algorithm  $A$  for its solution which has execution time,  $T(n) = O(n^k)$  for some constant  $k$ . Therefore,  $T(n)$  is bounded by a polynomial function of  $n$ . A polynomial-time (polynomial) algorithm is one whose time complexity function is  $O(p(n))$ , where  $p$  is some polynomial and  $n$  is the input length of an instance. Each algorithm whose time complexity function cannot be bounded this way is called an exponential-time algorithm (Garey and Johnson, 1979).

Any scheduling problem can be formulated as a decision problem, for example "Is there a feasible schedule with the given resource and precedence constraints?". Note that the 'yes' answer can be certified by a small amount of information and can typically be verified in polynomial time. A decision problem can not be computationally harder than the corresponding optimisation problem e.g. "Find the feasible schedule which has the smallest schedule length". That means that if one is able to solve an optimisation problem in an efficient way, then it will also be possible to solve a corresponding decision problem efficiently. On the other hand, if the decision problem is computationally hard, then the corresponding optimisation problem will also be hard (Demeulemeester and Herroelen, 1997).

**P** class consists of all decision problems that may be solved by the Turing machine (an abstract computer), in time bounded from above by a polynomial time algorithm in the input length. The **NP** class [U+FFFC] of decision problems consists of all decision problems for which no polynomial time algorithms are known but for which the 'yes' answer can be verified in polynomial time. It follows that  $\mathbf{P} \subseteq \mathbf{NP}$ . If a **NP** – **complete** problem would be solvable in polynomial time, then each problem in **NP** would be also solvable in polynomial time. The principal notion in defining [U+FFFC] **NP** – **completeness** is that of a reduction. For two decision problems  $P$  and  $Q$ , we say that  $P$  reduces to  $Q$ , if there exists a polynomial-time computable function  $g$  that transforms inputs for  $P$  into inputs for  $Q$  such that  $x$  will be a yes input for  $P$  if and only if [U+FFFC]  $g(x)$  is a yes input for  $Q$ . A decision problem is **NP** – **complete** if it is **NP** and every other problem in **NP** can be reduced to it. An optimisation problem will be called **NP** – **hard** if the associated decision problem is **NP** – **complete**. To prove that an optimisation problem is computationally hard, one has to prove that the corresponding decision problem is **NP** – **complete**. To prove that an optimisation problem is easy, it is sufficient to find an optimisation polynomial-time algorithm (Brucker, 2007).

Blazewicz et al. (1983) have shown that the RCPSP belongs to the class of the strongly  $[U+FFFC]$  **NP – hard** problems. More specifically, the decision problem corresponding to the RCPSP was proven to be **NP – complete** using reduction from the 3-partition problem, that concerns the decision whether a given set of integers can be partitioned into triples that all have the same sum. The 3-partition problem has been proven to be  $[U+FFFC]$  **NP – complete** by Garey and Johnson (1979). Therefore, the corresponding optimization problem, the RCPSP, is **NP – hard**.

#### 2.4.4 Solution Methods

For hard optimisation problems, like the RCPSP, exact algorithms, which always determine an optimal solution and approximation algorithms, which only provide approximate solutions, are distinguished.

Exact algorithms for project scheduling problems usually are either linear programming or branch and bound approaches. Branch-and-bound is the most widely used solution technique for solving RCPSPs when optimal solutions are needed, as very often it is the only available technique for the generation of optimal solutions within an acceptable computational effort. A heuristic may be defined as a logical sequence of steps giving a not necessarily optimal solution but good enough to be used in practice. The heuristic procedures for RCPSP fall into two categories, constructive heuristics and improvement heuristics. Constructive heuristics start from an empty schedule and add activities one by one until one feasible schedule is obtained. To that purpose, the activities are typically ranked by using priority rules which determine the order in which the activities are added to the schedule. Improvement heuristics, start from a feasible schedule that was obtained by some constructive heuristic. Operations are performed on a schedule which transforms a solution into an improved one. These operations are repeated until a locally optimal solution is obtained. In this category fall meta-heuristics like tabu search, simulated annealing and genetic algorithms.

One of the basic drawbacks with heuristics is their validation that is usually based on the comparison of average and worst case behaviour of the under examination heuristic on large, often randomly generated problem sets compared to known optimal results. Another drawback of heuristics is the impossibility to absolutely guarantee in advance which particular heuristic, or combination of heuristics, will produce the best results for a given problem. In spite of these drawbacks, heuristics are widely used in practice in order to cope with complex, highly combinatorial sequencing and scheduling problems (Herroelen et al., 1998).

The literature about resolution approaches for this problem is quite extensive, both in terms of heuristic and optimal procedures, as shown in numerous surveys on the field (Boctor, 1990; Sampson and Weiss, 1993; Icmeli and Erenguc, 1994; Ulusoy and Ozdamar, 1994; Ozdamar and Ulusoy, 1995; Kolisch, 1996; Herroelen et al., 1998; Kolisch and Padman, 2001; Hartmann and Briskorn, 2010).

##### 2.4.4.1 Branch and Bound

The method was first proposed by Land and Doig in 1960 for discrete programming. Branch and Bound (B&B) is a divide and conquer method, where a large problem is repeatedly divided into smaller ones, the "branch" part and for each sub-problem are estimated the possible solutions and if not promising the branch is ignored, the "bound" part. B&B searches

the complete space of solutions for a given problem without explicitly enumerating all of them by utilising bounds on the optimisation function in combination with the current best solution. This way parts of the search space are searched only implicitly. To describe branch and bound in detail, some terminology is introduced:

- Node: any partial or complete solution.
- Leaf node: a complete solution in which all of the variable values are known. Leaf nodes have objective function values, which are actual values and not estimates.
- Bud node: a partial solution, either feasible or infeasible. Bud nodes have associated bounding function values.
- Bounding function: the method of estimating the best value of the objective function obtainable by growing a bud node further. It should be an optimistic estimator to avoid omitting good solutions.
- Branching: the process of creating the child nodes for a bud node. One child node is created for each possible value of the next variable.
- Incumbent: the best complete feasible solution found so far.

Each specific branch-and-bound algorithm is defined as a set of rules for:

1. branching: given a bud node, how the child nodes will be created,
2. lower bound calculation: how to calculate the lower bound of the node, that is the number that bounds from below the solution set that can be generated by this node,
3. next node selection: how to choose the bud node from which to branch next,
  - best-first node: choose the bud node that has the best value of the bounding function anywhere on the B&B tree
  - depth-first node: choose from the children nodes of the current node, this way each iteration leads to one step deeper into the tree and early incumbent solutions are achieved,
  - breadth-first node: expand bud nodes in the same order in which they were created,
4. pruning/fathoming: how to recognise if a node will lead only to infeasible or nonoptimal solutions, or that for every solution that can be created from this node a similar or better solution would be constructed by branching from a different node
5. incumbent: how to recognise that a leaf node's feasible solution is the optimal one

Numerous branch-and-bound procedures for solving certain variants of the RCPSP optimally were developed (Pritsker et al., 1969; Davis, 1973; Patterson and Huber, 1974; Stinson et al., 1978; Talbot and Patterson, 1978; Christofides et al., 1987; Bell and Park, 1990). Following the most efficient approaches (based on % deviation from optimality) are briefly presented: Talbot and Patterson (1978) approach consists of a systematic enumeration of all possible activity finish times with the order of nodes selection defined beforehand. A network cut to eliminate from explicit consideration inferior activity completion times is used in the enumeration phase of the algorithm.

Stinson (1978) developed a best-first branch-and-bound procedure in which nodes in the solution tree correspond to precedence and resource feasible assignments for a subset of the activities of a project. Node branching is based upon a four-element decision vector. Left-shift dominance and lower bound pruning are used to bound the search space. This procedure was reported (Patterson et al., 1989) to be the most effective and efficient.

Christofides et al. (1987) proposed the use of disjunctive arcs for resolving conflicts that are created whenever sets of activities have to be scheduled whose total resource requirements exceed the resource availabilities in some periods.

Demeulemeester and Heroellen (1992) presented an efficient depth-first B&B procedure, called DH-procedure that computational experiments have proven almost twelve times faster than the best-first procedure developed by Stinson et al. (1978).

Following, the DH-procedure is analysed, as a representative branch and bound algorithm for the RCPSP. It generates a search tree having as nodes partial schedules  $PS$  in which finish times temporarily have been assigned to a subset of the activities of the project. The partial schedules are considered at time instants  $m$  corresponding to the completion time of one or more project activities. In  $PS$  scheduling decisions are temporary in the sense that in child nodes activities priorly scheduled may be delayed as a result of decisions made at later stages. Partial schedules are built up starting at time 0 and proceeding by adding at each decision point subsets of activities until a complete feasible schedule is obtained.

At time  $m$  the corresponding partial schedule  $PS_m$  will contain some activities which have been finished and others which are still in progress. The former activities have finish times smaller than or equal to  $m$  and are placed in the set  $F_m$ , the latter activities belong to the set  $S_m$ , of activities in progress. At every time instant  $m$  the eligible set  $E_m$  as the set of activities which are not in the partial schedule and whose predecessor activities have finished, therefore these activities can start at time  $m$  if the resource constraints are not violated (Algorithm 3). If it is impossible to schedule all eligible activities at time  $m$ , a resource conflict occurs and leads to new branching in the solution tree. Each branch describes a way to resolve the resource conflict through decisions about which combinations of activities are to be delayed. A delaying set  $D(p)$  consists of all subsets of activities  $D_q$ , either in process or eligible, the delay of which would resolve the current resource conflict at level  $p$  of the search tree.  $D_q$  is minimal if it does not contain other delaying alternatives as a subset (Algorithm 4). For every delaying alternative a set of extra precedence relations  $G_q$  is constructed by setting as predecessor the earliest finishing, in progress or eligible to start at time  $m$ , activity. Each delaying alternative is evaluated by computing the critical sequence lower bound  $LB$  as defined by Stinson et al. (1978).

Two dominance rules are used: the left-shift dominance rule and cutsets. The left-shift dominance rule is invoked on a non empty delay set, and consists on checking the hypothesis if the precedence relationships which were added at previous levels of the search tree forced an activity to become eligible at time  $m$  and the current decision was to start that activity at time  $m$  and if delaying activity set  $DS$  would allow activity  $i$  to be left-shifted without causing a resource conflict, then the corresponding partial schedule is dominated.

The second dominance rule is based on the concept of a cutset. At every time instant  $m$  a cutset  $C_m$  is defined as the set of all unscheduled activities for which all predecessor activities belong to the partial schedule. If a cutset  $C_k$  stored previously and belonging to a different tree path is equal to the current cutset  $C_m$  and its activities finish no later of those in  $C_m$  and  $k \leq m$  then the current partial schedule can be dominated.

Backtracking occurs when a schedule is completed or a branch is to be fathomed by the lower bound calculation and/or dominance rules. If there is no delaying alternative left unexplored at this level, backtracking to the previous one occurs. When level zero (root node) is reached, the search process is completed and the optimal solution has been found and has been verified.

#### 2.4.4.2 Schedule Generation Scheme

Constructive heuristics consist of two major components, the scheduling scheme and the priority rule. The scheduling scheme determines the way in which a feasible schedule is constructed by assigning starting times to the different activities. The two basic scheduling schemes are the serial and the parallel. The priority rule, on the other hand, determines the activity that is selected next during the heuristic search process. The usage of some priority rule

---

**Algorithm 2.1: Branch and Bound - DH-procedure (Demeulemeester and Herroelen, 1992)**


---

```

Step 1: Initialization;
 $T = 9999; p = 0; m = 0; f_i = 9999;$ 
 $f_1 = 0; PS = 1; S = \{1\};$ 
Calculate  $LB(0) = RCPL_i;$ 

Step 2: Calculate next decision point  $m;$ 
 $m = \min\{f_i, i \in S\};$ 
 $S = S - \{j \mid f_j = m\};$ 
if ( $n \in S$ ) then
     $T = f_n;$ 
    if ( $T = LB(0)$ ) then
         $STOP;$ 
    else
         $goto STEP 7$  (backtracking);
    endif
endif
if  $C$  dominated then  $goto STEP 7;$ 
else save  $S, f_i, m;$ 
;
 $E = \emptyset;$ 
 $E = E \cup \{i \mid i \in C, s_i = m\};$ 
if  $E = \emptyset$  then  $goto STEP 2;$ 
;
if  $S = \emptyset$  then  $goto STEP 3;;$ 
    else  $goto STEP 4;$ 
;

Step 3: Parallelization;
//  $\forall i \in E$  and  $i \notin PS_m$  count  $j$  that can be fs executed with  $i \rightarrow parCount_i$ 
if  $parCount_i = 0$  then
     $PS = PS \cup \{i\}, S = \{i\}, f_i = m + d_i;$ 
     $C = C - \{i\} + \{x \mid x \in Pred_i, Pred_x \in PS\};$ 
     $\forall x \in C: s_x = f_i;$ 
else if  $parCount_i = 1$  and  $d_j \leq d_i$  then
     $PS = PS \cup \{i, j\}, S = \{i, j\}, f_i = m + d_i, f_j = m + d_j;$ 
     $C = C - \{i, j\} + \{x \mid x \in (Pred_i || Pred_j), Pred_x \in PS\};$ 
     $\forall x \in C: s_x = f_i;$ 
if an activity was scheduled in STEP 3 then  $goto STEP 2;$ 
    else  $goto STEP 4;$ 
;

Step 4: Temporary Partial Schedule;
 $PS = PS \cup E, S = S \cup E, f_i = m_i + d_i;$ 
 $s_x = \max\{f_a \mid (a, x) \in H\};$ 
 $C = C - E + \{x \mid x \in Pred_i, Pred_x \in PS\};$ 
foreach  $k \in K$  do
    if  $\sum_{i \in S} r_{ik} > b_k$  then
         $goto STEP 5$ 
    else
         $goto STEP 2;$ 

```

---

results in a priority list, in which the activities are set in precedence feasible non-increasing order of priority. The generated feasible schedules fall into one of the following categories, as shown in Figure 2.8:

**Algorithm 2.2:** Branch and Bound - DH-procedure (Demeulemeester and Herroelen, 1992)

---

```

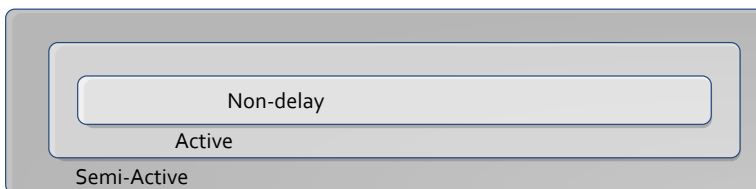
Step 5: Minimal Delay Sets;
p = p + 1;
foreach k ∈ K do
  | ck = ∑i∈S rik - bk
end
D(p) = {Dq ⊂ S | ∑i∈S rik ≥ ck } ∀k, Dq ⊈ Dr ∈ D(p);
foreach Dq ∈ D(p) do
  | foreach j ∈ (S - Dq) do
  | | Gq = {(j, i) | i ∈ Dq}
  | end
end
LB(p) = max{LB(p - 1), Lq*};
Dq* = {Dq* ∈ D(p) | min Lq*};
if LB(p) ≥ T then {goto STEP 7;};
else {STORE (f, PS, S, C, s, m)};
;

Step 6: Branching;
DS = {i ∈ Dq* | fi < m + di};
H = H ∪ Gq*; PS = PS - Dq*; S = S - Dq*;
foreach i ∈ Dq* do
  | fi = 9999;
  | si = {fi | (j, i) ∈ Gq*};
end
C = C + Dq* - {r | x ∈ Dq*, (x, r) ∈ H};
if DS ≠ ∅ then left shift dominance rule;
;
if PS dominated then goto STEP 7;
else goto STEP 2;
;

Step 7: Backtracking;
if p = 0 then STOP;;
else H = H - Gq*;
;
if D(p) = ∅ then p = p - 1; goto STEP 7;;
foreach Dq ∈ D(p) do
  | foreach j ∈ (S - Dq) do
  | | Gq = {(j, i) | i ∈ Dq}
  | end
end
LB(p) = max{LB(p - 1), Lq*};
Dq* = {Dq* ∈ D(p) | min Lq*};
if LB(p) ≥ T then
  | p = p - 1;
  | goto STEP 7;
else
  | RESTORE (f, PS, S, C, s, m);
  | goto STEP 6;
endif

```

---

**Fig. 2.8** Classification of Schedules

- Semi-active schedules: Feasible schedules obtained by sequencing activities as early as possible. In a semi-active schedule no activity can be started earlier without altering the precedences.
- Active schedules: Feasible schedules in which no activity could be started earlier without delaying some other activity or breaking a precedence constraint.
- Non-delay schedules: Feasible schedules in which no resource is kept idle when it could start processing some activity.

Initially, this type of methods consisted of a single scheduling scheme that was combined with a single priority rule generating a single solution schedule which constitutes in a single-pass method. Consequently, the methods become a little more elaborate, by requiring the repetition of the process more than one times using different priority rules and/or scheduling schemes, this approach is known as multi-pass method.

The serial schedule generation scheme (serial SGS) dates back to a paper by Kelley (1963). It sequentially adds activities to the schedule until a feasible complete schedule is obtained. In each iteration, the next activity in the priority list is chosen and for that activity the first possible starting time is assigned such that no precedence or resource constraint is violated. Let  $g = 1, \dots, n$  be the stages of the serial SGS algorithm. Let  $S_g$  be the set of activities which have been already scheduled and  $D_g$  the eligible set of activities that is comprised of those activities whose predecessors have already been scheduled and completed, therefore  $D_g = \{j \mid j \notin S_g, \text{Pred}(j) \in S_g\}$ . Let  $F_g = \{f_j \mid j \in S_g\}$  denote the set of finish times of activities at step  $g$  and  $\tilde{R}_k(t) = R_k - \sum_{j \in \text{Act}(t)} r_{jk}$ ,  $k \in K$  the remaining capacity of resource type  $k$  at time instant  $t$ . The serial SGS is shown in Algorithm 5.

---

**Algorithm 2.3:** serial Schedule Generation Scheme

---

```

 $F_0 = 0, S_0 = \{0\};$ 
for  $g = 1$  to  $n$  do
    Calculate  $D_g, F_g, \tilde{R}_k(t)$  ( $k \in K, t \in F_g$ );
    Select  $j \in D_g$ ;
     $EF_j = \max_{h \in \text{Pred}(j)} (f_h + d_j)$ ;
     $f_j = \min \{t \in [EF_j - d_j, LF_j - d_j] \cap F_g \mid r_{jk} \leq \tilde{R}_k(\tau), k \in K, \tau \in [t, t + d_j] \cap F_g\} + d_j$ ;
     $S_g = S_{g-1} \cup \{j\}$ ;
end
 $f_{n+1} = \max_{h \in \text{Pred}(n+1)} \{f_h\}$ 

```

---

For a given priority list, the application of the serial scheduling scheme requires [U+FFFC] time  $\mathcal{O}(n^2k)$  (Pinson et al., 1994). It has been proven by Kolisch (1995) that any schedule that is generated by the serial scheduling scheme belongs to the set of active schedules, that have the property that none of the activities can be started earlier without delaying some other activity. For scheduling problems with a regular performance measure the optimal solution will always be in the set of active schedules.

Contrary to the serial scheduling scheme, the parallel scheduling scheme Brooks (1963) iterates over the different decision points at which activities can be added to the schedule, thus it does time incrementation. These decision points correspond with the completion times of already scheduled activities and thus at most  $n$  decision points need to be considered in the parallel scheduling scheme. At each decision point, the unscheduled activities whose predecessors have completed are considered in the order of the priority list and are scheduled on the condition that no resource conflict originates at that time instant.



More specifically, for each stage  $g$  there is a schedule time  $t_g$ . Activities which have been scheduled up to  $g$  either belong to the complete set  $C_g = \{j | f_j \leq t_g\}$  or to the active set  $A_g = Act(t_g) = \{j | f_j - d_j \leq t < f_j\}$  of stage  $g$ . The eligible set  $D_g = \{j \notin (C_g \cup A_g) | (Pred(j) \subset C_g) \cap (r_{jk} \leq \tilde{R}_k(t_g))\}$  is composed of all the activities which can be precedence and resource feasibly started at  $t_g$  and  $\tilde{R}_k(t_g) = \tilde{R}_k - \sum_{j \in A_g} r_{jk}$  is the remaining capacity of resource type  $k$  at time instant  $t_g$ . The parallel SGS is illustrated in Algorithm 6.

---

**Algorithm 2.4:** *parallel* Schedule Generation Scheme
 

---

```

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

```

---

For a given priority list, the application of the parallel scheduling scheme also requires [U+FFFC] time  $\mathcal{O}(n^2k)$  (Kolisch and Hartmann, 1999). It has been proven by Kolisch (1996) that any schedule that is generated by the parallel scheduling scheme belongs to the set of non-delay schedules which are schedules where, even if activity preemption is allowed, none of the activities can be started earlier without delaying some other activity. The set of non-delay schedules is a subset of the set of active schedules but it has the drawback that it might not contain an optimal schedule for a regular performance measure.

### Priority rules

”A priority rule is a mapping which assigns each activity  $j$  in the decision set  $D_g$  a value  $v(j)$  and an objective stating whether the activity with the minimum or the maximum value is selected” (Kolisch and Hartmann, 1999). In case of tie the simplest way to resolve it is to choose the activity with the smallest activity label, however there are several tie breaking rules.

Research on priority rules for the RCPSP has been quite extended from the very early days of the field (Cooper, 1976; Doersch and Patterson, 1977; Alvarez-Valdes and Tamarit, 1989; Boctor, 1990; Ozdamar and Ulusoy, 1995; Thomas and Salhi, 1998).

Table 2.2 gives an overview of the most frequently used priority rules and their mathematical formulations: greatest rank positional weight (GRPW), latest finish time(LFT), latest start time (LST), minimum slack (MSLK), most direct and indirect successors  $Suc_j$  of activity  $j$  (MTS), resource scheduling method (RSM) that is applied to the  $AP$  set of eligible activity pairs, shortest processing time (SPT) and worst case slack (WCS) that also employs the  $AP$  set and sets  $E(i, j)$  as the earliest precedence and resource feasible start time of activity  $j$  assuming that activity  $i$  was started at the schedule time  $t_g$ .

**Table 2.2** Priority Rules based on Kolisch and Hartmann (1999)

Rule	Priority of $j$	Reference
GRPW	$d_j + \sum_{j \in \text{Suc}_j} d_j$	Alvarez-Valdes, Tamarit (1989)
LFT	$LF_j$	Davis, Patterson (1975)
LST	$LF_j - d_j$	Kolisch (1995)
MSLK	$LF_j - EF_j$	Davis, Patterson (1975)
MTS	$ \text{Suc}_j $	Alvarez-Valdes, Tamarit (1989)
RSM	$\max_{(i,j) \in AP} \{0, t_g + d_j - (LF_i - d_i)\}$	Shaffer et al. (1965)
SPT	$d_j$	Alvarez-Valdes, Tamarit (1989)
WCS	$LF_j - d_j - \max_{(i,j) \in AP} \{E(i, j)\}$	Kolisch (1996)

### 2.4.4.3 Solution Representations

Generally, meta-heuristic approaches for the RCPSP do not operate on actual schedules but on representations of schedules where the representation is transformed into a schedule through a decoding procedure. Consequently, operators used to produce new solutions should take into consideration the selected representation. Operators fall into two categories: a) unary operators that produce a new solution from an existing one, as in the case of neighbourhood move in SA and TS and mutation in GA., b) binary operator where a new solution is generated from two existing ones, as in GA's crossover. Following are briefly presented the most commonly used solution representations in RCPSP:

- *Activity List*: In the activity list representation, a precedence feasible activity list  $\lambda = \{j_1, j_2, \dots, j_n\}$  is given, in which each activity  $j_g$  must have a higher index  $g$  than each of its predecessors in  $\text{Preds}(j_g)$ .
- *Random Key of Priority Representation*: In this representation an array  $\rho = \{r_1, r_2, \dots, r_n\}$  is used to assign real-valued number  $r_j$  to each activity  $j$ . These  $r_j$  values are used as priorities, meaning that activities are ordered and scheduled in descending order of  $\rho$ . This encoding is called random key representation after Bean (1994).
- *Priority Rules*: The priority rule representation, is based on a list of priority rules  $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ , where each  $\pi_i$  is a priority rule (e.g. *LFT*, *LST*, *MSLK*, etc.) and each activity  $i$  should be scheduled according to the corresponding priority rule  $\pi_i$ . This representation was adapted by Hartmann (1998) to the RCPSP, from the job shop problem (Dorndorf and Pesch, 1995).
- *Shift Vector Representation*: In the shift vector representation a solution is represented by a shift vector  $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ , where  $\sigma_j$  is a non negative integer. The decoding procedure consists in calculating the starting time  $S_j$  of each activity  $j$  as the maximum of the finish times of all its predecessors plus the shift  $\sigma_j$  of activity  $j$  (Sampson and Weiss, 1993).

### 2.4.4.4 Genetic Algorithms

Genetic Algorithms (GA) firstly introduced by Holland (1975) were inspired by the process of biological evolution. A Genetic Algorithm is a problem solving technique based on the concepts of evolution and hereditary that is well fitted in cases of complex problems with large solution spaces due to its intrinsic parallelism, which allows efficient exploration of these spaces (Sevaux and Dauxere-Peres, 2003). The idea is to generate a group of initial solutions and iteratively work toward their improvement. This group of solutions is called

population and its initial form, initial population. A genetic algorithm starts with the generation of a number of random solutions for the problem, properly encoded, using one of the solution representations, as chromosomes, to form the initial population. Then new populations (off-springs) are repeatedly generated by combining the chromosomes of the current population using rules for combining chromosomes (crossover operators), randomly changing parts of chromosomes (mutation operators) and choosing which chromosomes will pass to the next generation using a selection policy (selection method). The fitness value measures the quality of a solution, usually based on the objective function value of the optimisation problem to be solved. In Algorithm 7 is outlined a standard genetic algorithm.

---

**Algorithm 2.5:** Pseudo-code of a standard genetic algorithm

---

```

set populationSize=POP;
set crossoverType; set probMutation ;
set generation counter g=0 ;
Generate initial population  $P_g$ ;
while stopping criteria not met do
    Evaluate  $P_g$ ;
    Crossover  $P_g$  and get  $P_{children}$  ;
    Mutate  $P_{children}$ ;
    Evaluate  $P_{children}$ ;
    Select from  $P_g$  and  $P_{children}$  and form  $P_{g+1}$ ;
     $g = g + 1$ ;
end

```

---

Studying the applications of GAs to the RCPSP (Hartmann, 1998, 2002; Kim et al., 2003; Cervantes et al., 2008; Mendes et al., 2009; Montoya-Torres et al., 2010; Peteghem and Vanhoucke, 2010; Wang et al., 2010; Xie et al., 2010; Proon and Jin, 2011) can be deduced that although GAs generally are very efficient procedures for finding global optima or near optimal solutions, the definition of activity representation, fitness function, crossover and mutation operators and selection process are the decisive factors for the efficiency and effectiveness of the algorithm. At the same time, these factors differentiate each proposed solution from those already existing. In depth analysis and experimental comparison of the initial population generation (Kim and Ellis, 2010), activity representation, crossover and mutation operators (Hartmann and Kolisch, 2000) can be found in the literature.

#### 2.4.4.5 Simulated Annealing

Simulated Annealing (SA) was introduced by Kirkpatrick et al. (1983), it is based on a concept from the physical annealing process in which a metal is heated to above the critical temperature, maintained in a suitable temperature and then cooled. The process begins with a single initial solution that is used as basis to generate a so-called neighbourhood by slightly perturbing the initial solution. The new solution will be accepted and used to proceed the search when it is better than the current one. However, it can also be accepted with a probability when it is worse. This acceptance probability depends on the cooling temperature that is a parameter initially set at such value to allow the acceptance of a large proportion of the generated solutions and it is gradually decreased to reduce the acceptance rate of less promising solutions. This prevents the algorithm from getting trapped in a local optimum at

early stages (Boctor, 1996). The algorithm is stopped as soon as a stopping criterion reaches a predetermined value. This solution method can be classified as a First Fit Strategy .

Following the simulated annealing algorithm of Bouleiman and Lecocq (2003) is briefly presented in Algorithm 8. The solution representation used in this approach is the activity list and the decoding procedure is the serial schedule generation scheme.

Neighbourhood generation begins with the current solution and a randomly selected activity. The positions of this activity's latest predecessor  $lp$  and earliest successor  $es$  are calculated. Then the new position of the activity is randomly chosen within  $[lp, es]$ . The neighbour is obtained by a cyclical (left/right) shift of all the activities placed between the old and the new positions.

---

**Algorithm 2.6:** Simulated annealing algorithm by Bouleiman and Lecocq (2003)

---

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

---

The cooling scheme consists of a multiple cooling chain  $C$  that is restarted each time a different initial solution is tested. The number of neighbourhoods tests in every step  $s$  of the chain is progressively increased as  $N_s = N_{s-1}(1 + h \cdot s)$ , where  $h$  defines the step length.

The temperature  $T$  is decreased in  $S$  steps, starting from an initial value  $T_0$ , which is supposed to be high enough to allow acceptance of any new neighbour in the first steps and using an attenuation factor  $\alpha, 0 < \alpha < 1$ . In each step  $s$ , the procedure generates a fixed number of neighbour solutions  $N_{s,ol}$  and evaluates them using the current temperature value  $T_s = \alpha^s T_0$ .

#### 2.4.4.6 Tabu Search

Tabu Search (TS) was developed by Glover (1989) and can be defined as a steepest descent / mildest ascent method. It starts with a single solution used to create a neighbourhood and then all the generated solutions are evaluated and the best one is chosen and used in the next iteration. This process can very easily lead to cyclic moves around a local optimum. In order to avoid this problem a number of previous moves are stored in a memory like data-structure, the so-called tabu list, which is used to reject repeating moves that could lead back to a recently visited solution. Usually, a tabu status can be ignored only in the case that the proposed move would lead to a new overall best solution, based on the so called aspiration rule (Nonobe and Ibaraki, 2002b). In Algorithm 9 is outlined a generic tabu search algorithm for RCPSP.

---

#### Algorithm 2.7: Pseudo-code for tabu search algorithm

---

```

Generate initial solution  $x_0$  and calculate  $f(x_0)$ ;
Initialise TabuList;
 $x_{best} = x_0, f_{best} = f(x_0)$ ;
 $x_{current} = x_0, f_{current} = f(x_0)$ ;
while stopping criteria not met do
    Generate  $Moves(x_0)$  list of candidate moves ;
    while move not effectuated do
        Select best move  $M(x')$ ;
        if  $M(x') \notin TabuList$  OR  $M(x')$  meets AspirationCriteria then
            Execute move  $M(x')$ ;
             $x_{current} = x', f_{current} = f(x')$ ;
            Update TabuList;
            Update AspirationCriteria;
        end
    end
end

```

---

The usage of tabu search in the RCPSP is not as extended as the genetic algorithm and simulated annealing but has often given very good results as in the case of the approached proposed by Klein (2000), Nonobe and Ibaraki (2002a) and Thomas and Salhi (1998).

#### 2.4.4.7 Particle Swarm Optimization

Particle Swarm Optimisation (PSO) simulates a social behaviour such as bird flocking to a promising position for certain objectives in a multidimensional space (I. C. Trelea, 2003). In PSO a population, called swarm, of individuals, called particles, is updated using information from both the local and the global search. Each particle represents a solution, that for PSO is a candidate position and it is treated as a point in an  $\mathcal{M}$  – *dimension* space. The particle is characterised by its position and velocity. PSO, as GA, is initialised using random particles to form the swarm and in each iteration improvement is obtained by adjusting the particle's position and velocity based on it's overall best position (local best) and the best position ever found by all particles (global best).

Let an  $N$  dimension space that has  $M$  particles. Let  $i$  be a particle,  $i = 1, \dots, M$  of  $N$ . Let the positional vector of  $i$  to be defined as  $X_i = \{X_{i1}, \dots, X_{iN}\}$  and the velocity vector as  $V_i = \{V_{i1}, \dots, V_{iN}\}$ . For each particle  $i$  the individual experience is  $L_i = \{L_{i1}, \dots, L_{iN}\}$  and

the global best experience is defined as  $G = \{G_1, \dots, G_N\}$ . The updating mechanism for each component is described by Equations 2.53 and 2.54 (Chen et al., 2010).

$$V_{ij}^{new} = wV_{ij}^{old} + c_1r_1(L_{ij} - X_{ij}^{old}) + c_2r_2(G_{ij} - X_{ij}^{old}) \quad (2.53)$$

$$X_{ij}^{new} = X_{ij}^{old} + V_{ij}^{new} \quad (2.54)$$

where  $w$  is a weighting parameter used to adjust the influence of the previous velocity to the new velocity,  $c_1, c_2$  are learning factors used to define the effect of individual and global experience to the velocity and  $r_1, r_2 \in [0, 1]$ , are random variables also influencing the balance between local and global search. In Algorithm 10 is shown the PSO algorithm's generic formulation.

---

**Algorithm 2.8:** Pseudo-code for PSO algorithm

---

```

Generate Swarm;
Initialise Local Best  $L$  and Global Best  $G$ ;
set  $w, c_1, c_2, r_1, r_2$ ;
while stopping criteria not met do
    foreach particle  $i$  in Swarm do
        Update Velocity;
        Update Position;
        Calculate Fitness of new particle;
        Update  $L, G$ ;
    end
end
end

```

---

Although PSO, the last few years has been applied in scheduling problems (Zhang et al., 2005, 2006; Deng et al., 2008; Jarboui et al., 2008; Li et al., 2009; Liu et al., 2009) only a few efforts have been done to use it for the RCPSP problem Zhang et al. (2006), Chen (2006).

## 2.5 Multi-Criteria Decision Making

"Decision aiding is the activity of the person who, through the use of explicit but not necessarily completely formalised models, helps obtain elements of responses to the questions posed by a stakeholder in a decision process" (Roy and Vanderpooten, 1997).

Multi-Criteria Decision Making is the most well known tool of decision aiding. It is a branch of a general class of Operations Research models which deal with decision problems under the presence of a number of decision criteria. This super class of models is divided into Multi-Objective Decision Making (MODM) that studies decision problems in which the decision space is continuous and Multi-Attribute Decision Making (MADM), that deals with problems with discrete decision spaces, where the set of decision alternatives has been pre-determined (Neumann and Zimmermann, 2000).

In broad terms, decision problems involving multiple axes of evaluation of the merits of potential alternative solutions can be classified as:

- Multi-objective optimization problems: alternatives are implicitly defined by a set of constraints defining a feasible region (search space) and the objective functions are optimised in this region.

- Multi-attribute decision making problems: alternatives and their performance according to the evaluation criteria are explicitly known before initiating the solution process. In this case the decision maker's partial preferences regarding each one of the multiple evaluation criteria need to be aggregated, which implies some loss of information. There are two main methodological approaches: Outranking and Multi Attribute Utility/Value Theory (MAUT/MAVT). Multi-Attribute Value Theory (MAVT) is a rigorous framework for computing an overall score for each alternative. The main difference of MAUT from MAVT is that it works with utility functions that take into account the clients' attitudes towards risk.

### 2.5.1 Multi-Objective Decision Making

In the single objective case we have a single objective function while all the other properties are defined through the constraints. In real-world projects it is very common to need to pursue at the same degree more than one objectives. The aim is to find a vector of decision variables which satisfies constraints and optimises a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which may be in conflict with each other. Hence, the term optimise means finding a solution which would give values to all the objective functions that are acceptable to the decision maker.

There is a number of differences between a single and a multi-objective optimisation problem. The latter usually has a) an optimal set with cardinality greater than one, b) two distinct goals of optimisation, instead of one, convergence to the Pareto-optimal solutions and maintenance of a set of maximally-spread Pareto-optimal solutions and c) two different search spaces, the multi-dimensional space that is formed by the objective functions, in addition to the usual decision variable space, common to all optimisation problems.

An ideal multi-objective optimisation procedure consists of two discrete phases, first a multiple trade-off optimal solutions with a wide range of values for objectives, should be found and in the second step one of the obtained solutions is chosen using higher level information (Deb et al., 2002), as shown in Figure 2.9.

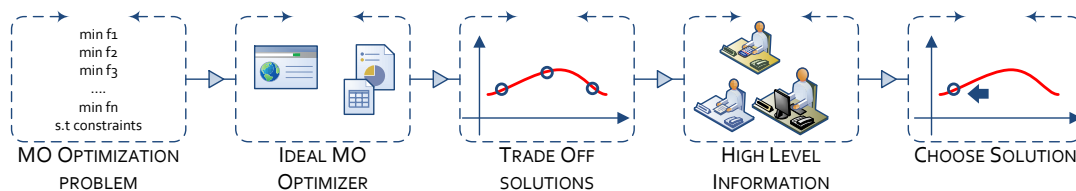


Fig. 2.9 Multi-objective optimisation procedure

Each trade-off solution corresponds to a specific order of importance of the objectives. Therefore, if a relative preference factor among the objectives is known for the problem being solved, there is no need to follow the above process and solve the multi-objective problem. Instead, a composite objective function as the weighted sum of the objectives, where a weight for an objective is proportional to the preference factor assigned to that particular objective can be formed. This method of scalarising an objective vector into a single composite

objective function converts the multi-objective optimisation problem into a single-objective optimisation problem. This path can be followed as long as the objectives may be expressed in the same metric.

The RCPS problems discussed above have a single objective function, usually the makespan minimisation, while all other properties of the schedule are controlled by means of constraints. However, several authors have employed multiple performance measures into their project scheduling problems. A widely used approach for such problems is to artificially convert them into a single-objective problem by defining one overall objective as the weighted sum of all the performance measures and solve it (Al-Fawzan and Haouari, 2005; Bomsdorf and Derigs, 2008). Another equally popular way is the generation of Pareto-optimal schedules (Hapke et al., 1998; Nabrzyski and Weglarz, 1999; Viana and Pinho de Sousa, 2000) and letting to the decision maker the final decision about which is the most fitted solution.

### 2.5.1.1 Single Objective Function Aggregation

This category consists of approaches that combine all objective functions to a single one. Scalarization is the traditional approach to solving multi-objective problems. It involves formulating a multi objective problem as single objective by means of a real-valued scalarizing function typically being a function of the objective functions of the initial problem, auxiliary scalar or vector variables, and/or scalar or vector parameters. The feasible set of the multi-objective problem can be additionally restricted by new constraint functions related either to the objective functions of the initial problem and/or the newly introduced variables. Therefore, all objective functions  $f_i(x)$  corresponding to the objectives  $i = 1, \dots, k$ , are aggregated into a single one (2.55) where the non negative weights used sum to 1 (2.56).

$$F(x) = \sum_{i=1}^k w_i f_i(x) \quad (2.55)$$

$$\sum_{i=1}^k w_i = 1, \quad w_i > 0 \quad (2.56)$$

If the weights are constant throughout the optimisation process, the method is called Conventional Weighted Aggregation, otherwise when the weights are dynamically adjusted during the optimisation, we have the Dynamic Weighted Aggregation. In this case the weights are not user defined, but the way that the weights should change from one iteration to the next, is defined.

The trade-off solution obtained by using the preference-based strategy is largely sensitive to the relative preference vector used in forming the composite function. A change in this preference vector will result in a different trade-off solution. The process of finding the relative preference vector is highly subjective and not straightforward as it requires an analysis of the non-technical, qualitative and experience-driven information to find a quantitative relative preference vector.



### 2.5.1.2 Pareto Optimality

Multi-objective optimisation problems consist of several objectives that are necessary to be handled simultaneously. Often the different objectives can be competing and/or incommensurable and need to be optimised concurrently.

Let  $S \subseteq R^k$  be a  $k$ -dimensional search space, and  $f_i(x)$ ,  $i = 1, \dots, k$ , be  $k$  objective functions defined over  $S$ . Let  $\mathbf{f}$  (2.57) be a vector consisting of all the objective functions and  $m$  inequality constraints (2.58).

$$\mathbf{f} = [f_1(x), f_2(x), \dots, f_k(x)] \quad (2.57)$$

$$g_i(x) \leq 0 \quad i = 1, \dots, m \quad (2.58)$$

The goal is to calculate a solution,  $x^* = (x_1^*, x_2^*, \dots, x_k^*)$ , that optimises (minimise /maximise)  $\mathbf{f}(x)$ . However, the objective functions  $f_i(x)$  may be conflicting with each other, therefore a unique global minimum cannot be found in the search space. Therefore, the definition of optimal solution should be adapted, giving rise to the concept of Pareto optimality, that was originally proposed by Francis Ysidro Edgeworth in 1881 and it was later generalized by Vilfredo Pareto (1896).

To overcome this issue, optimality of a solution for multi-objective problems is redefined as follows. Let  $\mathbf{u} = (u_1, \dots, u_k)$  and  $\mathbf{v} = (v_1, \dots, v_k)$  be two vectors of the search space  $S$ . Then,  $\mathbf{u}$  dominates  $\mathbf{v}$ , if and only if,  $\mathbf{u}$  no worse than  $\mathbf{v}$  (2.59) in all objectives, and it is better in at least one objective (2.60). This property is known as Pareto dominance.

$$u_i \leq v_i \quad \forall i = 1, 2, \dots, k \quad (2.59)$$

$$u_i < v_i \quad \text{for at least one } i = 1, 2, \dots, k \quad (2.60)$$

When Equations 2.59 and 2.60 stand, we equivalently say that  $\mathbf{v}$  is dominated by  $\mathbf{u}$  or  $\mathbf{u}$  is non dominated by  $\mathbf{v}$  or  $\mathbf{u}$  is non inferior of  $\mathbf{v}$ . A solution,  $x$ , of the multi-objective problem is said to be Pareto optimal, if and only if there is no other solution,  $y$ , in  $S$  such that  $f(y)$  dominates  $f(x)$ . The set of all Pareto optimal solutions of a problem is called the Pareto optimal set, and it is denoted as  $P^*$ . The set  $PF^* = \{f(c) : c \in P^*\}$  is called the Pareto front.

Primary goal when solving a multi-objective optimisation problem is to find the Pareto front. However, the Pareto optimal set can be infinite. Consequently, the goal is rendered as the detection of as many Pareto optimal solutions as to form an adequately spread Pareto front that is not distant from the actual Pareto front. The need for multiple trade-off optimal solutions (Pareto-optimal solutions), is based on the fact that any two Pareto solutions constitute a trade-off among the objectives and decision makers would be in a better position to make a choice when many such trade-off solutions are unveiled.

### 2.5.1.3 Evolutionary algorithms

Evolutionary algorithms such as evolution strategies and genetic algorithms have become the method of choice for optimization problems that are too complex to be solved using deterministic techniques (exact methods) such as linear programming. The large number of applications (Beasley, 1997) are due to several advantages of this kind of algorithms when compared to the deterministic techniques especially in cases of complex and/or large sized problems. These advantages can be summarised as: a) need of very restricted amount of information about the problem being solved, it is enough to have a way to represent the solutions and a measure to compare the different solutions and select the most fitted, b) ease

of implementation, no complex encoding is needed, c) robustness, and d) fitness for parallel optimisation (Sbalzarini, 2001).

Evolutionary algorithms seem to be particularly suited to multi-objective problems due to their ability to synchronously search for multiple Pareto optimal solutions and perform better global exploration of the search space ( Lamont, 2002, Deb, 1999 and Schaffer, 1984). More specifically, the population approach of evolutionary algorithms allows an efficient way to find multiple Pareto-optimal solutions simultaneously in a single simulation run.

This aspect has made the research and application in evolutionary multi-objective optimisation (EMO) popular in the past decade (Deb, 2001, Coello, 2002; Bagchi, 1999, Fonseca et al., 2003). Of special interest are approaches, implementing concepts such as fitness sharing and niching (Fonseca and Fleming, 1993, Goldberg, 1994 and Deb, 1994), and elitism (Deb, 2002, Erickson 2001; Zitzler 1999).

Following, the Strength Pareto Evolutionary Algorithm (SPEA) is presented, as it is a representative example of this class of algorithms (Zitzler and Thiele (1999), Zitzler and Thiele (2000)). The algorithm consists of the following steps:

Step 1: Generate random initial population  $P$  and create an initially empty set of non dominated individuals  $P'$ .

Step 2: Evaluate objective function for each individual in  $P$ .

Step 3: Select the non dominated members of  $P$  and copy them to  $P'$ .

Step 4: Examine solutions within  $P'$  and remove all those that are covered by an other member of the same set.

Step 5: If the number of non dominated solutions exceeds a given maximum size, remove the exceeding by means of clustering in order to have equally distributed non dominated members of  $P'$  along the Pareto-front.

Step 6: Calculate the fitness of each individual, both in  $P$  and in  $P'$ .

- Each solution  $i \in P'$  is assigned a real value  $s_i \in [0, 1)$ , called strength that is proportional to the number of population members  $j \in P$  for which  $i$  dominates  $j$ . The strength gives the fitness of the corresponding individual, as shown in Equation 2.61, where  $n$  is the number of individuals in  $P$  that are covered by  $i$  and  $N$  the size of  $P$ .
- The fitness of each individual  $j \in P$  is calculated by summing the strengths of all external non dominated solutions  $i \in P'$  that cover  $j$ . This sum is augmented by 1 to guarantee that members of  $P'$  always have better fitness than members of  $P$  (2.62).

$$f_{P'_i} = s_i = \frac{n}{N+1} \in [0, 1) \quad (2.61)$$

$$f_{P_i} = 1 + \sum_{i,i \succeq j} s_i, \quad f_i \in [1, N) \quad (2.62)$$

Step 7: Select individuals from  $P \cup P'$  to create the predefined number of pairs for off spring generation.

Step 8: Apply crossover and mutation operators in order to create a new population  $P_{new}$ .

Step 9: If maximum number of generations is reached, then stop, else set  $P = P_{new}$  and go to Step 2.

### 2.5.2 Multi-Attribute Decision Making

Multi-Attribute Decision Making (MADM) deals with the problem of choosing an option from a set of alternatives which are characterized in terms of their attributes. It is a mainly

qualitative approach that requires inputs from the decision maker about the preferences among the alternative solutions based on their performance on the selected criteria. Final goal is to obtain the optimum alternative that has the highest degree of satisfaction for all of the relevant attributes (Ribeiro, 1996)

In MADM, three concepts play a fundamental role for analysing and structuring the decision aiding process: a) alternatives, b) criteria and c) problem types.

Potential action, is the object of the decision, or the goal of the decision making process. An action is qualified as potential when it is feasible or seems possible to implement it and thus, deserves to be taken into consideration during the decision making process. The subset of actions that are mutually exclusive and therefore cannot be implemented conjointly, defines the alternatives of the decision making problem under question. (Roy and Mousseau, 1996).

A criterion is a measure of effectiveness, which provides the basis for the analysis. It is constructed for evaluating and comparing potential actions according to a point of view which must be well-defined and consistent throughout the decision making process. It is necessary to define explicitly the set  $[U+FFFC]$  of all the possible values that can be given when evaluating each criterion. To enable comparability, it should be possible to define a complete order of the evaluations, which is called the scale of the criterion. The impact matrix displays the performance of each of the alternatives according to the criteria in an appropriate scale (Roy, 1994). According to Baker et al. (2001), criteria should be able to discriminate among the alternatives and to support the comparison of the performance of the alternatives, complete to include all goals, operational and meaningful, non-redundant and as few in number as possible. An attribute is a performance parameter. Attributes may be quantitative and/or qualitative. In some approaches, qualitative attributes must be transformed into quantitative ones by means of some operation. If that is strictly required by the method, transforming qualitative into quantitative measurements can be done using a bipolar scale (e.g 0-100) although it is an arbitrary process. Also, even quantitative attributes can be expressed in different measurement units. However, if needed the different measurement scales can be transformed into a common (artificial) scale by e.g. rescaling or Euclidean normalisation (C. Bana e Costa, 1990).

In order to compare two actions according to a criterion the two degrees of preference used for evaluating their respective performances need to be compared. This leads to distinguishing various types of scales: (Roy, 1990)

- Ordinal: alternatives can be ordered but no information exists about the distances between the different levels. Consequently, the gap between two degrees does not have a clear meaning in terms of difference preferences, as in the case of verbal scales where having pairs of consecutive degrees doesn't lead to invariant preference difference all along the scale. This type of scale is often called a qualitative scale.
- Numerical: a scale whose degrees are defined by referring to a clear, concrete defined quantity. Furthermore, the total absence is clearly defined and each degree of the scale can be calculated as the sum of some kind of units. This scale substantially gives information about the difference of each alternative to a non-arbitrary origin. In this case, as opposed to the ordinal scale, the ratio between two degrees can receive a meaning which does not depend on the two particular degrees considered. This type of scale is also called quantitative, cardinal or ratio scale.

The problem's formulation, is used to define the way that the problem should be posed, the expected results and often the most appropriate solution process. Usually the MADM problems are classified in three groups (Roy, 1996):

- Choice: select the best alternative or a reduced set of good alternatives. The idea is to select a small number of good actions in such a way that a single alternative can be chosen.

However, it is not implied that the selection will lead to the optimum solution. In this category, also fall problems that concern the elimination of as many actions as possible so as to have as result a very restricted set of alternative solutions.

- Ranking: order the alternatives from the best one to the worst one, even if there are incomparable alternatives.
- Sorting: assign the alternatives to predefined ordered classes (categories) of merit. Each action should be assigned to only one category that is judged the most appropriate among those that were initially defined as categories.

The decision making process can be summarised in the following steps:

Step 1: Identification of the stakeholders, decision makers and main actors in the decision.

Step 2: Problem definition. At this step system and organizational boundaries and interfaces, assumptions and overall goal are identified. The expected output consists in the problem's statement, including initial conditions and expected results.

Step 3: Requirements specification. The requirements are expressed as the problem's constraints. They are used to describe the set of the feasible/acceptable solutions of the decision problem.

Step 4: Goals definition. Goals are the objectives set by the decision makers for the specific problem under consideration. There can be one or more, often conflicting, objectives.

Step 5: Alternatives identification. Alternatives are the actions that can be taken in order to achieve the goals. Alternatives should be mutually exclusive, the actions that can be executed conjointly, usually are grouped together to form a single alternative. In the mathematical formulation of the decision problem, correspond to the solutions.

Step 6: Criteria definition. Criteria related to the problem and its context, and are used to evaluate the alternatives. Each criterion shows how well each alternative achieves the goals from the point of view defined by the criterion. It is expected to have at least one criterion per goal/objective.

Step 7: Alternatives evaluation. At this step the evaluation of the alternatives against the criteria takes place and depending on the criterion, the assessment may be objective using some kind of measurement scale or can be judgmental, reflecting the subjective assessment of the evaluator.

Step 8: Solutions validation. The final solutions should be validated against the initial problem statement and their feasibility should be confirmed.

### 2.5.2.1 Outranking

Outranking Methods (OMs) were first developed in France in the late sixties following difficulties experienced with the value function approach in dealing with practical problems. The outranking decision aid methods compare all couples of actions. Instead of building complex utility functions, they determine which actions are being preferred to the others by systematically comparing them on each criterion. An outranking relation is developed to model the "non-controversial" component of the decision maker's preferences. This relation should allow incomparability of alternatives and intransitivity. The outranking relation is exploited according to the problem type (choice, ranking, sorting) to be solved.

The outranking methods are based on the idea of building a preference relation that is called outranking relation, among alternatives evaluated on several attributes. An outranking relation is defined as a binary relation  $S$  on the set  $X$  of alternatives such that  $xSy$  stands if there are enough arguments to declare that  $x$  is at least as good as  $y$ , and there is no essential reason to refute the statement. This definition assumes that the existing data about the preferences of

the decision-maker are satisfactory, the quality of the evaluations of the alternatives is good and the nature of the problem complies to the selected decision making method. In most cases the outranking relation is built through a series of pairwise comparisons of the alternatives. Pairwise comparisons usually are made using the concordance-discordance principle: "an alternative  $x$  is at least as good as an alternative  $y$  ( $xSy$ ) if: a majority of the attributes supports this assertion (concordance condition) and if the opposition of the other attributes—the minority—is not "too strong" (non-discordance condition)" (Roy and Vanderpooten, 1997). The concordance-discordance principle is based on a "voting" analogy and can be applied without having to do a detailed analysis of trade-offs between attributes (Bouyssou, 1986). The application of this principle gives rise to binary relations which are neither complete ( $NOT(xSy)$  different from  $NOT(ySx)$ ) nor transitive, therefore, it is not a simple process to go from the outranking relation to the final alternative recommendation, thus the application of specific techniques is needed (Roy and Vanderpooten, 1997). The most popular families of methods under this category are ELECTRE (Roy and Vanderpooten, 1997) and PROMETHEE (Brans, 1992) that represent "the European school" of multi-criteria decision making.

### 2.5.2.2 Multi Attribute Utility Theory (MAUT)

Multiple attribute utility theory (MAUT) is an Multi Attribute Decision Making approach that tries to assign a utility value to each action. This utility is a real number representing the preferability of the considered action. Very often the utility is the sum of the marginal utilities that each criterion assigns to the considered action.

More specifically, in MAUT the preferences according to each criterion are aggregated into a function, which measures the global preference of each alternative. This preference relation should be complete and transitive. There is an underlying compensation effect, therefore bad performances in some criteria may be compensated by good performances in other criteria. MAUT methods are based on the use of utility functions which quantify the preferences of a decision-maker by assigning a numerical index to each level of satisfaction of a particular criterion. For a single criterion, the utility of satisfaction of a consequence  $x$  is denoted by  $(u(x))$ . Utility functions are constructed such that  $(u(x))$  is less preferred to  $(u(x'))$  if and only if  $x$  is less preferred to  $x'$ . Therefore, utility functions are used to transform the raw performance values of the alternatives against diverse criteria, both quantitative and qualitative, to a common scale in such a way that a more preferred performance obtains a higher utility value. The methods of this class are differentiated based on the technique used to derive the function and its mathematical properties.

$$U_i = \sum_j W_j U_{ij}, \quad \forall i \quad (2.63)$$

$$U_{ij} = u(X_{ij}), \quad n \leq i \leq 1, \quad m \leq j \leq 1 \quad (2.64)$$

The most common formulation of a utility function is the additive model (2.63), where  $U_i$  is the overall utility value of alternative  $i$ ,  $U_{ij}$  is the utility value of  $j$  criterion for  $i$  alternative (2.64),  $n$  is the number of criteria,  $m$  is the number of alternatives and  $W_j$  is the relative weight of  $j$  criterion. Depending on the decision maker's attitudes toward risk, utility functions can be concave, which describe risk-averse situations, convex for risk-seeking situations or linear, in cases of risk-neutral situations.

### Analytic Network Process (ANP)

Thomas Saaty (Saaty, 1980) has presented a methodology to build utility functions, the AHP (Analytic Hierarchy Process) and its more recent generalisation, the ANP (Analytic Network Process). ANP is a theory of measurement that uses pairwise comparisons along with expert judgments to deal with the measurement of qualitative or intangible criteria. ANP is based on four axioms: (1) reciprocal judgments, (2) homogeneous elements, (3) hierarchic or feedback dependent structure, and (4) rank order expectations (Wiecek et al., 2008).

It is a multi-criteria decision making method where a graph structure is created using the problem's components and the decision maker is asked to pairwise compare the components, in order to determine their priorities. The method is based on relative measurements used to derive composite priority ratio scales from individual ratio scales that represent relative measurements of the influence of elements that interact with respect to control criteria (Saaty, 1996). Paired comparisons are made with judgments using numerical values taken from the AHP absolute fundamental scale of 1-9 (Saaty, 1996) to capture the outcome of dependence and feedback within and between clusters of elements. A scale of relative values is derived from all these paired comparisons and it also belongs to an absolute scale that is invariant under the identity transformation.

The elicitation of pairwise comparison judgements and the possibility of expressing them verbally are cornerstones of the popularity of AHP/ANP. However, a key problem in the applicability of the method is the fact that the priority vector derived from the principal eigenvalue method used in AHP/ANP can violate the order of the respective preference intensities, causing semantical inconsistencies between the decision maker's preferences and their representation in the model (Bana e Costa et al., 1999).

Summarising, the idea is to analyse the problem and extract the critical factors that affect the decision along with the most viable alternative solutions. These factors, called criteria in ANP, are grouped, based on some common property, in clusters, to make easier the decision process. Each model should have a cluster containing all the alternative solutions of the problem and one or more clusters containing the elements/decision criteria. Then the relationships among all the objects of the model, both clusters and elements, should be defined. These relationships can be either internal among elements of the same cluster or external from an element of a cluster to an element of another cluster. The decision maker is asked to compare couplets of elements with respect to some common property that they share. From that point begins the computational part of the method, which should be automated through software tools (Saaty and Sagir, 2009; Onut et al., 2011; Rokou and Kirytopoulos, 2012).

Following, the conceptual model of the ANP method is analysed and a brief description of the process of applying the ANP is provided (Saaty and Sagir, 2009):

*Step 1: Based on the decision goal, a network structure including clusters, criteria and alternatives should be configured.* The decision problem should be described in detail including its objectives, criteria, actors and their objectives and the possible outcomes of that decision. At this point the details of influences that determine how that decision may come out, are defined. Therefore, the decision maker selects which components influence the decision and how they are grouped together. Hence, the network of clusters and their elements is determined.

Let  $C$  be the set of clusters composing the ANP model, and let  $N_i$  be the set of nodes (criteria) belonging to cluster  $C_i$ . These sets fully describe the ANP network's clusters and criteria. The alternatives are not handled in any special way; they are just another cluster throughout the process.

*Step 2: The dependences among all components of the previous structure should be identified and listed, in order to define the impacts among all the elements.* For each control criterion or sub-criterion, the clusters of the general feedback system with their elements are determined and connected according to their outer and inner dependence influences. A relationship is defined from the source cluster to any cluster whose elements influence it. The approach to be followed in the analysis of each cluster or element, influencing other clusters and elements with respect to a criterion, or being influenced by other clusters and elements, is selected and should be kept consistent throughout the entire project.

Let  $R$  be the criteria relationship matrix.  $R$  is a  $k \times k$  matrix, having  $k = \sum_{i=1}^N n_i$  where  $r_{ij} = 1$  if and only if  $n_i$  node influences  $n_j$  node, otherwise  $r_{ij} = 0$ . Let  $Q$  be a  $n \times n$  matrix, holding the cluster relationships. Matrix  $Q$  is calculated from  $R$  based on the assumption that if one or more criteria of a cluster are connected to one or more criteria of another cluster, then the first cluster influences the second. Consequently inner and outer dependences among the components of the ANP model are described using  $R$  and  $Q$  matrices. In other words, the decision maker decides which criterion or group of criteria is influenced by other criteria or even whole groups of criteria and this way creates the graph describing the decision space of the problem currently being solved.

*Step 3: Pairwise comparison matrices of the components with interval judgments have to be constructed.*

For each control criterion, the Supermatrix is constructed by laying out the clusters in a pre-defined order and all the elements in each cluster both vertically on the left and horizontally at the top. The priorities derived from the paired comparisons are entered as sub-columns of the corresponding column of the Supermatrix. Then paired comparisons on the elements within the clusters themselves according to their influence on each element in another cluster they are connected to (outer dependence) or on elements in their own cluster (inner dependence), are performed. Comparisons of elements according to which element influences a given element more and how strongly more than another element it is compared with are made with a control criterion or sub-criterion in mind.

Let  $A_{C_i}$  be the cluster's pairwise comparison matrix containing all judgments done by the decision maker and having as control element cluster  $C_i$ . Let  $B_{C_i N_j}$  be the node's pairwise comparison matrix, containing all judgments done by the decision maker and having as control elements: cluster  $C_i$  and node  $N_j$ . For each  $a_{kl}$  (or  $b_{kl}$ ), if  $k = l$  then  $a_{kl} = 1$  that represents the comparison of an item with itself and if the matrix is reciprocal then  $a_{kl} = \frac{1}{a_{lk}}$ . These comparison matrices are filled in by asking the decision maker to compare pairs of elements in relation to a control element (i.e. is criterion "Price" more important than criterion "Cost of Repairs" in relation to criterion "Budget" or the opposite? How much?).

*Step 4: For each comparison matrix consistency should be checked and judgments should be adjusted till the maximum inconsistency is less than 10% of the order of magnitude of the actual measurement.* Having an  $A_{C_i}$  (clusters) or  $B_{C_i N_j}$  (criteria - nodes) matrix, that is positive due to the scale used to represent decision makers judgments and it is reciprocal due to the way it was created, we need to determine if the contained judgments are consistent or in the opposite case, if the inconsistency is within acceptable levels. As measure of deviation from consistency, we use the introduced by Saaty (Saaty, 1996) consistency index (C.I.):

$$C.I. = \frac{\lambda_{max} - n}{n - 1} \quad (2.65)$$

where  $\lambda_{max}$  is the Perron eigenvalue of the positive reciprocal matrix being examined. The consistency ratio (C.R.), of the pairwise comparison matrix is the ratio of its inconsistency

index  $C.I.$  to the corresponding random index value,  $C.R. = \frac{C.I.}{R.I.}$ . Random index ( $R.I.$ ) values are computed using multiple simulations of randomly created comparison matrices and calculating the average of the consistency index. If the  $C.R.$  of a pairwise comparison matrix is larger than 10% then it is necessary to find which are the most inconsistent judgments in that matrix and ask the decision maker to consider changing his judgment to a value that will lead to an acceptable value of  $C.R.$  The most inconsistent judgment can be computed using the formula:

$$\text{Max}(a_{ij} * \frac{w_j}{w_i} \forall i, j \in 0, 1, \dots, n) \quad (2.66)$$

*Step 5: The relative importance weights (local priorities) from each matrix can be calculated.* Some optimization methods, used for priority elicitation are the eigenvalue method, the least squares, the weighted least squares and the logarithmic least squares. Another way to compute a priority vector is to use the sum of the rows of each power of the matrix in combination with Cesaro summability and Perron's theorem (Tarazaga, 2001). The last one is used in the herein discussed algorithm. *Step 6: Supermatrix and Cluster Matrix have to be filled with the weights elicited during Step 3 and then the Supermatrix should be transformed to column stochastic.* This way all the results from questioning the decision maker are summarised and used as a basis for the following calculation steps.

*Step 7: Weighed Supermatrix calculation.* This step uses the results from the paired comparisons that were performed on the clusters to weight the element's priorities, as they influence each cluster to which they are connected with respect to the given control criterion. The derived weights are used to weight the elements of the corresponding column blocks of the Supermatrix. When there is no influence, a zero value should be assigned.

To compute the Weighted Supermatrix, the Supermatrix is transformed to column stochastic, and then the Hadamard product (Cheng, 2007; Li, 2007) of the updated Supermatrix with Cluster Matrix, is calculated. If needed the columns are again normalized to keep summing to 1. Attention should be given to columns that are from the beginning stochastic and thus should not take part in the transformations. Furthermore, in the case where an entire vector but not all vectors in that component are zero then the weighted column must be renormalized. Last issue are sink components that need not to be included in Supermatrix, instead its priorities will be used in the process of synthesis after the calculation of the limited priorities. *Step 8: The Weighted Supermatrix should be limited by raising it to a sufficiently large power until it converges into a stable limit matrix.* In the end, the weights of criteria and alternatives are aggregated into final priorities. The process starts having a stochastic matrix  $W$ . First of all, being  $W$  stochastic, we know that  $\lambda_{max} = 1$ , because the principal eigenvalue of a matrix lies between its largest and smallest column sums, and all columns of a column stochastic matrix sum to 1.

Primary goal is to compute the limit matrix by calculating powers of this matrix till the limit is reached, that is when  $W^{n+1} = W^n$ . In this case all the columns of the matrix are identical and priorities can be easily computed for the elements of each cluster. However, this is not always a straightforward calculation of matrix powers. In order to select the computational method for getting the limit matrix from the Weighted Supermatrix we need to know if it is reducible or not. A matrix is reducible if it can be placed into block upper-triangular form by simultaneous row/column permutations. Thus, a matrix is reducible (Muoneke, 1987; Mesnard and Dietzenbacher, 1995) when its associated digraph is not strongly connected. An easy way to control if a square matrix is irreducible is based on the Perron–Frobenius theory of nonnegative matrices where is proved that a square matrix is irreducible, if and only if for each  $i$  and  $j$ , there exists some  $k$  such that  $(I + W)^{n-1} > 0$ . The corresponding model cannot have source or sink nodes.



Knowing that the matrix in question is irreducible, the next step is to define if it is primitive or cyclic. A sufficient condition for a matrix to be primitive is to be a nonnegative, irreducible matrix with a positive element on the main diagonal. In that case the limit matrix calculation is given by raising the Weighted Supermatrix to large powers. On the other hand, a square matrix  $A$  such that the matrix power  $A^{k+n} = A^n$  for a positive integer  $k$  is called a cyclic matrix (Tam, 1999). If  $k$  is the least such integer, then the matrix is said to have period  $k$ . If the matrix is reducible and cyclic then the result is calculated by averaging all matrices belonging to a cycle and normalising the results by blocks. In the other case, when the matrix in question is reducible we have to determine if  $\lambda_{max} = 1$ , is simple or multiple root and if there are other roots of unity or not. If there are other unitary roots then it is a cyclic matrix and limit can be computed in the same way used for irreducible cyclic matrix. If  $\lambda_{max} = 1$ , is a simple root and the matrix is reducible the same computational steps with those used for irreducible primitive matrix will give the desired result. If  $\lambda_{max} = 1$ , is a multiple root and the matrix is reducible then we are talking about hierarchies and the limit matrix can be computed as the average of all powers of the matrix till the point that  $W^k = 0, k \leq n$ .

Two kinds of outcomes are possible. In the first all the columns of the matrix are identical and each gives the relative priorities of the elements from which the priorities of the elements in each cluster are normalised to one. In the second the limit cycles in blocks and the different limits are summed and averaged and again normalised to one for each cluster.

*Step 9: Perform sensitivity analysis on the final outcome.* Sensitivity analysis is concerned with "what if" kind of question to investigate whether the final results are stable to changes in the inputs whether judgments or priorities. Of great importance is control if these changes affect the order of the alternatives or not. The Compatibility Index of the original outcome and each new outcome can be used to measure how significant the change is.



## **Chapter 3**

### **Research Method**

#### **3.1 General Research Methods**

Undertaking a research study aims at finding answers to a problem or more generally a question and it is implied that the process is being undertaken within a framework of a set of approaches, uses procedures, methods and techniques that have been tested for their validity and reliability and it is designed to be unbiased and objective. Therefore a process to qualify as research, it must, as far as possible, be controlled, rigorous, systematic, valid and verifiable, empirical and critical (Ball et al., 1995).

There are various ways of classifying the research methods based on the application of the research study (pure or applied), the objectives in undertaking the research (descriptive, correlative, explanatory, exploratory), the inquiry mode employed (quantitative or qualitative). The current research followed the quantitative approach. Following are presented the main differences between the quantitative and qualitative research methods to show the rationale behind this choice.

Qualitative research methods are more appropriate for exploring the nature of a problem, issue or phenomenon without quantifying it. Focal point of this type of research is to describe the variation in a phenomenon, situation or attitude, the reasons that generate the phenomenon, the factors that affect it and how it is affected, leading this way to new hypothesis or explanations of the why and when something happens. On the other hand quantitative research methods focus on the generalisation of experimental results, quantification of data and are more appropriate when a very structured approach that will give specific results is needed.

Furthermore, there are several technical differences between qualitative and quantitative research methods, for example the former usually needs small samples for the research as opposed to the latter that usually needs large and random samples to be valid. Additionally, the data collection methods are usually unstructured or loosely structured methods as opposed to quantitative data collection methods that give less freedom and focus on reproducibility.

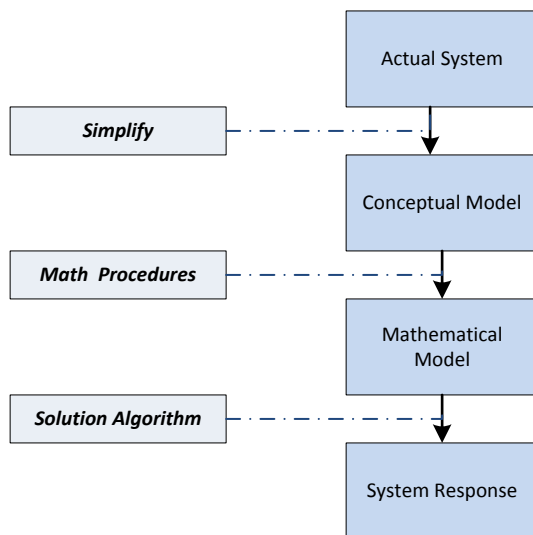
#### **3.2 Mathematical Modelling**

Having in mind that one of the major aspect of this Thesis is the generation of a holistic conceptual and mathematical modelling of the project scheduling problem, following the basic notions of mathematical modelling and the relevant process are described.

A mathematical model is a description of a system using mathematical concepts and language. Mathematical models are often classified according to how much a priori information is available for the system or from having all the needed information at hand to not having any. However, in practice all systems are somewhere between these two states.

Mathematical modelling is defined as the implementation of mathematics in solving unstructured problems in real-life situations. In other words, mathematical approaches are used in finding solutions related to real-life problems. The goal is to transform a real-life problem into a mathematical problem and solve it using mathematical techniques (Jin et al., 2001).

The process of developing a mathematical model usually begins with the statement of a problem which has come to light in some practical situation. The first task is to create a mathematical representation of the physical process, that is a first "draft" of the model being created which defines the basic variables and incorporates all the needed assumptions and constraints. From this model, information is extracted which must then be compared with physical evidence typically gained through experimentation. This comparison is designed to determine the worth of the mathematical model and potentially lead to adjustments based on the results. If the model appears to be inadequate it must be altered and new quantitative information gathered. Fidelity to the original problem and inclusion of all the needed details is of great importance if the modelling process is to be effective (Spanier, 1980). The basic steps of this procedure are depicted in Figure 3.1.



**Fig. 3.1** Mathematical modelling process and its validation

Noble (1982) summarises the major activities in mathematical modelling:

- understanding the actual practical problem to be modelled and defining the aspects to be modelled or the specific instance types that are of interest and then try to simplify the representation,
- manipulating the problem and developing a mathematical model by identifying the variables and the relationships among them and constructing hypotheses, evaluating contextual information and in the end developing models,
- interpreting the provided solution and evaluating its validity and completeness

Therefore, mathematical modelling could be described as a loop between the real life problem to be modelled and the mathematics used to describe it until a satisfactory model is generated. The modelling process begins with a complicated real-life situation. A problem representation is obtained from that situation. Then, a mathematical model is obtained via a mathematical study performed on the actual model. The proposed initial model is applied to the real problem and if it conforms with the reality then the goal has been reached, otherwise certain stages or the entire modelling process is repeated.

A crucial step of the modelling process is the evaluation of whether or not a given mathematical model describes a system accurately (Noble, 1982). The validation of a mathematical model can be a difficult task. The easiest part of model evaluation is checking whether the generated model fits experimental measurements or other empirical data. Assessing the scope of the generated model by determining what situations the model is applicable to, can be less straightforward especially if no initial instantiation of the problem at hand has ever been done.

Summarising, mathematical modelling aims at describing in a precise and quantifiable mode a practical problem in such a way that it will be close to the practical cases, enough generic to cover multiple if not all the instances of the targeted problem and as simple as possible to be easily applicable.

### 3.3 Algorithm Design

An algorithm is a procedure to accomplish a specific task, composed of a finite number of steps handling a well-specified set of inputs characterising the problem at hand and resulting in a predefined output. When defining an algorithmic problem, the complete set of instances it must work on, should be specified.

There are several general approaches to the construction of efficient solution algorithms to problems, usually called algorithm design paradigms, providing templates suited to solving a broad range of diverse problems. Each algorithm design paradigm is expressed in such a way to be easily translated into clear execution steps using one or more common data structures (provided by most high level programming languages). The resulting algorithms have specific temporal and spatial requirements that characterise the algorithm (e.g. number of fundamental instructions to be executed on worst case scenario). Each problem can be solved using more than one algorithmic design techniques and lead to correct results. However, often a design pattern (paradigm) can lead to clearly superior algorithms than the other alternatives (e.g. to more efficient algorithms). Following some of the most popular design paradigms are presented:

- Brute force is a straightforward approach to solve a problem based on the problem's statement and definitions of the concepts involved. It is considered as one of the easiest approaches to apply and is useful for solving small – size instances of a problem. In optimisation problems, brute force methods exhaustively explore the solution space until finding the best solution.
- In greedy algorithms the solution is constructed through a sequence of steps, each expanding a partially constructed solution obtained during the previous steps. The core of this method is that at each step the choice must be locally optimal.
- Divide-and-conquer is a top-down method where the given instance of the problem is split into several smaller sub-instances (of the same problem) and the process is repeated until having a number of smaller and usually simpler problems that are independently solved

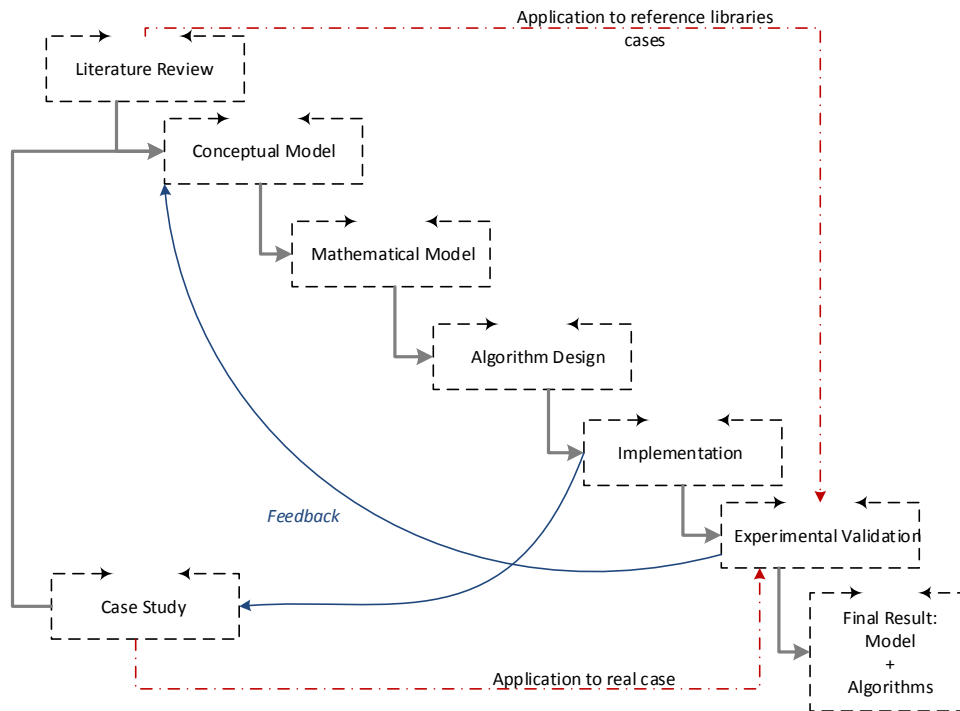
and then the sub-instance solutions are combined so as to yield a solution for the original instance, usually by means of recursion.

- Dynamic Programming improves the divide-and-conquer logic by maintaining a table of the sub-instances results to avoid recalculations of the same sub-problems. It is a bottom-up method as the smallest sub-instances are solved first and are used to solve progressively larger sub-instances.
- Backtracking and branch-and-bound methods are used for state-space search problems, where the problem representation consists of an initial state, one or more goal states and a set of operators to pass from one state to another. Optionally a cost function for evaluating the cost of each operation and a utility function for evaluating the closeness of the current state to the goal state, can be used. The solving process is based on the construction of a state-space tree, whose nodes represent states, the root represents the initial state, and one or more leaves are goal states. Each edge is labeled with some operator. The solution is obtained by searching the tree until a goal state is found. Rules for branching, bounding and backtracking are defined based on the specific problem.
- Meta-heuristics are computational methods that optimise a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. These algorithms have very little or no information about the problem being solved. Their great advantage is that they can search very large spaces of candidate solutions. Even so, these methods do not guarantee that an optimal solution is ever found.
- Evolutionary algorithms are population based meta-heuristics which are used mainly for optimisation problems for which the exact algorithms are of very low efficiency. These methods search for good solutions to a problem from among a (usually very large) number of possible solutions. The current set of possible solutions is used to generate a new set of possible solution. These algorithms are inspired by biological evolution and use mechanisms like reproduction, mutation, fitness and selection.

The proposed solution algorithm is an adaptive evolutionary algorithm that based on the instance being solved selects the best fitted solution algorithm among different available meta-heuristics.

### 3.4 Research Process

The process followed during this research is shown in Figure 3.2. Starting from the project scheduling literature review and preliminary field study, a first conceptual model of the problem has been defined and then mathematically formulated. Following a set of algorithms of proven efficiency for this kind of problems where implemented and based on the experimental data an innovative adaptive algorithm was designed and added to the implementation in order to moderate the single and multi-objective optimisation process.



**Fig. 3.2** Research Process

This evolutionary algorithm, based on the instance being solved selects through evolution the most proper algorithm and uses it to find a near-optimal solution. After the implementation of the model and the solution algorithm a large set of data available in the literature were used to initially fine tune the algorithms used and then prove its validity and efficiency. The set of model plus algorithm was used for the actual scheduling of a real life medium sized project and feedback gathered from the modelling stage was used to update the model. Finally, the numerical results were compared to those given by commercial scheduling tools.





## Chapter 4

# Problem Definition

### 4.1 Systems approach to project scheduling

The traditional analysis of projects and project scheduling is based solely on the project characteristics (duration of activities, resource availability, costs, etc.) and takes as granted that all the information is available at the start of the project and will remain unchanged during its execution, allowing the design of an optimal schedule and leaving to the project manager the task to keep the project on track.

In practice, management needs to be dynamic, responding to new information, easily adaptable to disruptions rather than sticking to the original and able to make decisions based on the actual project state instead of the perceived one which often is quite different to the reality (Rodrigues and Bowers, 1996). This does not mean that the traditional operational research methods for project scheduling should be rejected but that they should be properly adapted so as to endorse the dynamic environment and the effect of contextual parameters. Traditional methods are still the best way to set the initial schedule and budget-related goals, though they can only be achieved if all goes strictly according to plan. The effort is to enhance these methods so as to endorse environmental parameters related to the organisation and the specific situation being faced, without losing their generality and the ability to give specific results.

Systems approach, or systems thinking, which has been connected with the development of operational research and management science from its beginning especially through the work of Churchman (1963) and Ackoff (1962), may well advance project scheduling. The systems approach is being steadily adopted in management thinking especially for the last two decades (McMaster, 1996; Battram, 1998). Systems theory is about studying the properties of entities in relation both to their components and the ways that they interact with each other and with their environment. The fundamental systems thinking ideas can be summarised as follows:

- studying each situation holistically as a set of interacting elements within an environment,
- recognising that the relationships and interactions between elements are more important than the elements themselves in determining the behaviour of the system,
- recognising the hierarchy of system's levels and the consequent ideas of properties emerging at different levels, and mutual causality both within and between levels,
- accepting that different people act differently, in accordance with their specific purposes and syllogisms.

Therefore, in project scheduling, when having in mind a systems approach, it is important to make a detailed analysis not only of the specific project data and components, such as the

activities and the resources needed, but also of the ways these elements interact with each other and how the overall organisation's strategies, priorities and other projects or external environmental parameters affect the initial optimisation problem formulation and the choices that should be done during the scheduling and the monitoring phase.

#### ***4.1.1 Problem Structuring***

Problems often are classified as "hard" and "soft" problems. A problem is defined as "hard" when all its components are fully observable, closed to the environment and have sub-systems fully aligned to the main system. On the other hand, "soft" problems usually arise when not all parts of the system in question are observable, contain epistemic uncertainty, evolve over time, interact with the environment and/or involve political, cultural and ethical aspects (Daniel, 1990).

Project management's definition as a "hard" problem is based on the assumption that the decision to initiate a project is based on a well thought-out strategy, against which the outcome of the project can be objectively evaluated, having all the important components described in detail and closed to the environment. Similarly, project scheduling is thought as a "hard" problem assuming that all the needed information are initially available, remain unchanged during the process and the only relations are the internal ones that are clearly stated through the activities relationships. At the start of the twenty first century, the view of project scheduling, as a "hard" problem has come under criticism (Costello et al., 2002; Checkland and Winter, 2005; Winter, 2006). In practice, projects can be sometimes initiated as an outcome of non-rational decision, undertaken with the process in mind rather than the outcomes, and pursued despite environmental changes which eventually leave the project objectives obsolete or even undesirable (Checkland and Winter, 2005). Project scheduling is not immune to changes and interactions caused by its components as well as external factors and contains probabilistic elements like the activities durations, the resource requirement and availabilities. Therefore it can be defined as a "soft" problem.

When dealing with project scheduling as a "soft" problem it is very important to use a well fitted problem structuring method, that is a modelling approach very useful for framing and defining the issues constituting the problem. One very popular method for problem structuring is the Soft Systems Methodology (SSM) that is a methodology and a learning system (Rosenhead and Mingers, 2001) which can be used both for general problem solving and management of change.

SSM uses the notion of "system" as a dialectical tool, through developing rich pictures and root definitions to enable debate amongst concerned parties. In its initial form the methodology consists of seven steps, starting with an initial appreciation of the problem situation leading to the modelling of several human activity systems that are relevant to the problem situation. By discussions and exploration of these, the decision makers will arrive at accommodations on what is the problem being solved and what are its parameters. The seven stages of SSM can be summarised as:

1. set initial description of the problem to be tackled with, giving focus to the area of interest and not the exact problem,
2. express the problem, in all its "richness" (Checkland, 2000) usually using a rich picture,
3. extract root definitions of relevant systems. In this step the passage from the "real" world to the systems, is effectuated and relevant perspectives are defined (Customer - Actor - Transformations - World view - Owner - Environment)

4. develop the conceptual model using systems conventions,
5. compare the models with the real world,
6. define which changes are desirable and feasible under the specific circumstances,
7. implement the selected changes.

SSM uses a schematic model in the sense that it uses rich pictures to identify and represent the problem. Rich picture diagram is an early draft of the structure of the processes with the individual actors and their relation to each other. It can be used as a means of communication between the analyst and the users of the system and therefore uses the terminology of the environment in order to be self explanatory and easy to understand.

The drawing of the rich picture is subjective and the process of drawing it, is in itself useful, by forcing decisions, illustrating and discussing the roles in the organisation and identifying possible conflicts. Based on the guidelines provided by Checkland a rich picture should include: structures, processes, climate, people, issues expressed by people and conflicts. A rich picture describing the project scheduling problem is presented in Figure 4.1. All people involved in the process of project scheduling and partly in its execution are depicted. We focus on the project manager that is responsible for the generation of an optimal project schedule given the task to be completed, the organisation policies and priorities, the identified risks and the customer's needs and the financial manager, that could be also a role covered by the project manager, who is concerned with the top level financial issues that are related to the project. Project and financial managers are usually in conflict with each other, as the former tries to make a robust schedule with minimal risks and the other to force lower usage of resources, minimal cost and balanced cash flows. These conflicting objectives when are handled by the same person, as in cases that the project manager also covers the position of the financial manager, are even more difficult to balance.

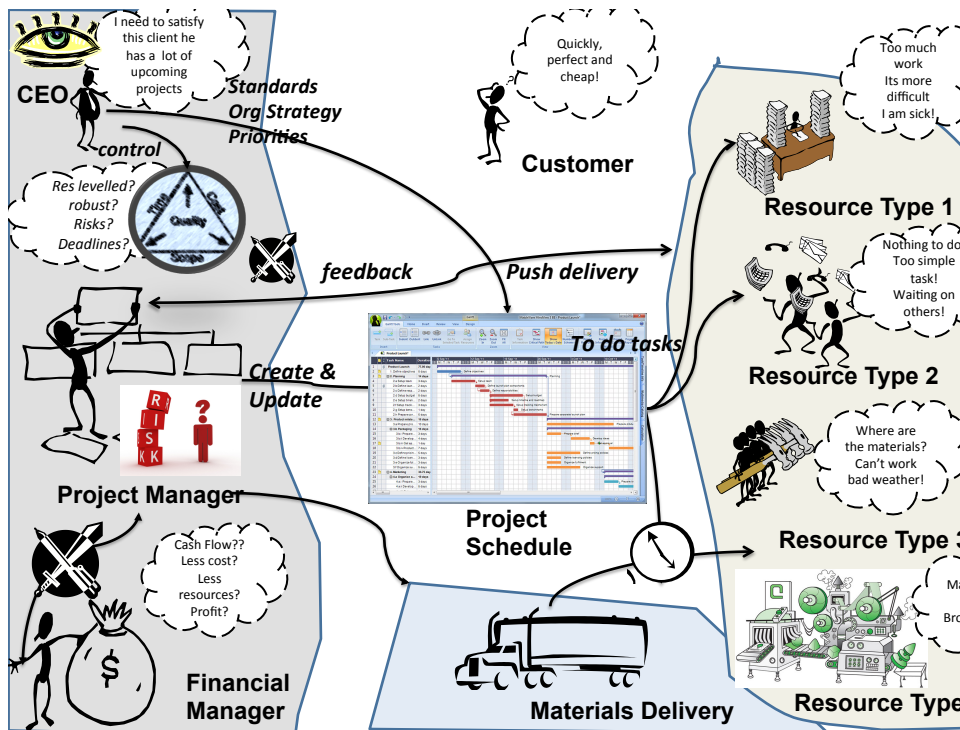


Fig. 4.1 Project Scheduling Rich Picture

Furthermore, there is the customer aiming at getting the best possible result at the lowest possible price and with high quality. The executive management of the organisation where along with relevant legal and international standardisation authorities monitor from a distance the project and ensure that both legal and related standards restrictions are obeyed and a satisfactory profit is earned fulfilling at the same time the expectations of the customer. The resources used to implement the project, the human resources - internal and external work teams- have different views and expectations from the project schedule. Workers need a balanced work load, possibility to leave work when there is the need to do so (e.g. time off, sickness), an unobstructed flow of work with no delays caused by erroneous synchronisation either caused by external contractors or belated deliveries of materials and of course they need motivation and clear guidelines to know what should be done, when and how and avoiding errors that cause frustration and extra work. Resources are also the machinery needed for the specific project that might break down and need replacement or maintenance during the project execution.

#### 4.1.2 System Dynamics view of project scheduling

System dynamics was introduced by (Forrester, 1961) as a method for modelling and analysing the behaviour of complex social systems, particularly in an industrial context and it has been used to examine various social, economic and environmental systems, where a holistic view is important and feedback loops are critical to catch the interrelationships. The focus is on the results of the interaction of positive loops that lead to continual growth or

decline and negative loops that lead to stability (Forrester, 1961, 1968, 1980). There are two stages in the process: a) identifying and mapping the causal loops and b) quantifying the causal loops and building a computer model. Often only the first stage is implemented and the process is stopped with the production of a causal-loop (influence) diagram, especially in cases that the aim is to get a good understanding of the situation at hand (Wolstenholme, 1999).

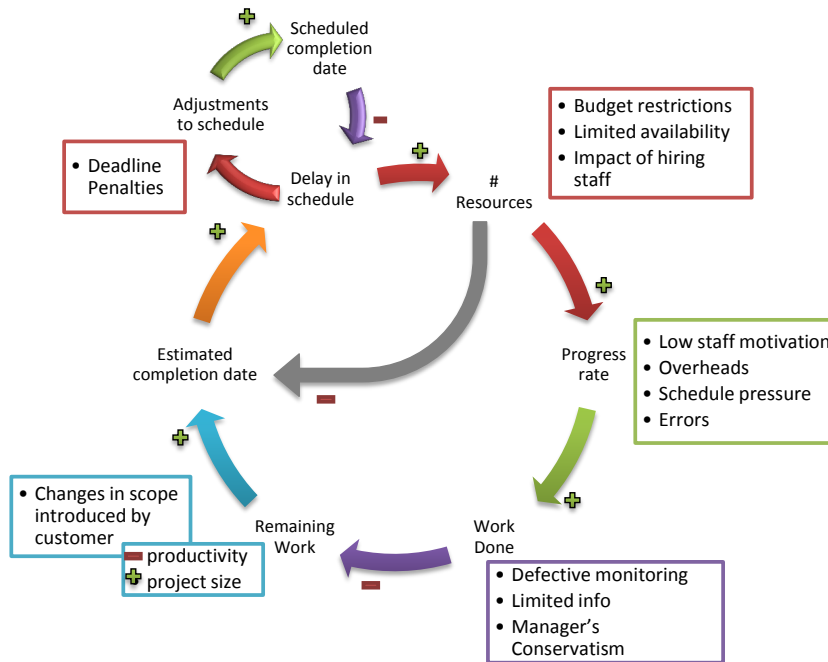
The idea of using system dynamics to project scheduling rises from the need to consider the project as a whole and not solely as the sum of its individual elements, activities and resources, recognising the difference among these two aspects (Daniel, 1990). An easy way for experimenting with management's options and possible effects to the whole organisation and not only the undergoing project, is offered to the project manager before deciding which route to follow by comparing the different possible routes. In this case, the emphasis is given to the influences of the different elements, on the identification of what might go wrong and the generation of realistic estimates that can result in an overall more realistic project model. In other words, a system dynamics analysis offers a distinctly different view of a project with the main output being a better understanding of the important underlying influences and the ways that they can affect the project under different circumstances.

The proposed approach includes the use of both traditional methods and system dynamics. For each discrete organisation, an initial influence diagram should be generated and used for highlighting the situations that cause the greatest actual-planned difference and the alternative paths for accommodating activities disruptions or wrong estimation of their duration. The results obtained from system dynamics studies can also be used to generate guidelines for use in estimating activity durations, resource requirements, costs and risks to reflect the underlying influences and identify typical behaviours. Then traditional OR methods are used to give results representing a desirable optimal schedule. Always keeping in mind that this initial project schedule is unlikely to be achieved unless there is no difference between actual and planned performance, but it is useful to have it as goal.

Toward this direction is the following project monitoring cycle inspired by (Rodrigues and Bowers, 1996), that is shown in Figure 4.2, where the alternative paths that can be followed in case of a perceived delay in the project schedule are presented. Moreover, Figure 4.2 shows how the environmental parameters influence each possible action and the associated chain of possible events. The management responds to a perceived delay in the project schedule by either deploying more resources or usage of schedule pressure, in order to increase the progress rate and bring forward the completion date or simply adjusting the schedule in order to accommodate the delay. However, there are disruptive factors that could cause problems in the implementation of any of the above responses.

Analytically, adding resources, equipment and/or staff, can be hindered by low availability of the needed resource type, restricted budget and low willingness of the management to change workforce due to organisation's policies or even legislation (hiring/firing). Increasing the number of resources should lead to increased progress rate, however, this not always the case, due to changes in the balance of the work team, augmented overheads for training the newly hired staff (if not equipment) and communication between the members of the enlarged team and possible higher rate of errors causing rework of activities and additional delays. On the other hand, using the schedule pressure as a mean to increase the existing staff's productivity instead of adding new personnel, can increase the amount of work done but it is very likely to lead to low motivation and increase of the errors rate and thus the activities needing rework. Rework is a very crucial point, as the number of errors and the time taken over their detection is very significant for having high, actual and not only perceived, progress rate, as the later the errors are discovered the more activities are likely to be affected.

Increased progress rate is expected to lead to more work done in less time, however, the perceived work done can not accurately reflect the actual work done, due to defective monitoring process for example not accounting for quality errors of the completed activities and limited information on the progress or inability to objectively account the progress (e.g. in some cases of code development).



**Fig. 4.2** Project monitoring cycle

Note, that in practical situations it is not improbable that during efforts to recover from a delay in schedule, new requests by the customer arise, which if accepted will lead to changes in scope, possibly increasing the project size and additionally setting back the schedule.

Finally, the second option of adjusting the schedule to overcome the delay, although is quite straightforward, it can require great effort in order to re-calculate an optimal schedule taking into consideration the work done, the rework needed and possible deadlines on the project and/or on specific project activities. It is also very probable to be hindered by existing arrangements made with sub-contractors and other external collaborators, material deliveries and arising need for storage or synchronisation points involving environmental parameters (e.g. weather related activities).

The above analysis, shows that both traditional approaches and system dynamics examine the same basic issues but from different perspectives. In system dynamics the focus is on feedback loops and the whole project instead of looking into details in restricted areas and ignoring others, giving great importance to those factors that are subjective and to the human's behaviour, aiming at the simulation of reality including human and system weaknesses in order to indicate likely outcome.

Consequently, the two approaches provide valuable complementary information, the traditional techniques supply the detailed output necessary for project monitoring, whereas sys-

tem dynamics offer a holistic view of the problem itself and useful general strategic lessons which should be considered when planning projects and producing the estimates for the traditional analysis. There are undeniable benefits in formally incorporating the two models (Rodrigues and Bowers, 1996) both for the success of the undergoing project and the ongoing learning process of the organisation itself.

## 4.2 Problem Description

Project scheduling is a quite complex problem that every project manager faces in the beginning of each project. The consequences of an ill designed schedule can seriously endanger the successful project execution and completion.

This problem, as analysed in section 4.1, although initially was faced as a "hard" problem assuming that it is fully observable, governed by well-defined laws of behaviour and closed to the environment, in practice has been proven that this is not the case. Still, when trying to take into consideration all the parameters defining and affecting a good project schedule, a very complex system emerges. Therefore, it is essential to balance the hard and soft aspect the project and try to define the project scheduling problem taking into consideration both but in a level of abstraction that will keep it general and permit its modelling and solution.

The idea is to provide a way to define the desired characteristics and provide a solution process that will generate project schedules adaptable to different project settings, organisational sizes and strategies and be scalable according to the size and criticality of the undergoing project. Furthermore, the process should be simple and quick enough to permit immediate re-runs for the generation of alternative scenarios giving the opportunity to the project manager and/or the group of decision makers responsible for the definition and final selection of the baseline schedule to have a satisfactory number of alternatives to discuss on and choose from.

First of all, the desired schedule characteristics should be defined and related to the environmental parameters that affect them either directly or indirectly. Prevailing organisation's strategic goals include customer satisfaction, profit and risk minimisation. Different contribution, of each factor for each organisation and time period, is assumed. Therefore a "good" project schedule should enable the organisation to:

- (a) satisfy the customer, leading to the expected product, as expressed in terms of quality and project's scope, in the predefined time frame, deadline, and with the agreed cost,
- (b) optimally use the available budget and minimise the cost without overruling (a) ,
- (c) manage human resources so as to conform with work hours, contracts and related legislation and make a balanced use of materials and equipment,
- (d) be as much insensitive to changes as possible and be robust. That is, meaning that small changes in scope, perturbations in activity's duration, due to internal or external parameters, or resource availability must not escalate but have limited effect to the total schedule.

### 4.2.1 Goals

The generated project schedule should lead to a product of the agreed quality, on time and on budget and this should be achieved with the lowest cost, balanced usage of resources and minimal sensitivity to unpredictable factors. The objectives to be pursued during project

scheduling are defined by translating these schedule characteristics, to actual objectives to be optimised during project scheduling.

In all projects there is a finish date, that is a deadline, either imposed by the customer or internally set. Hence, the project's duration is a natural measure of performance and as expected it is the objective which is most often found in the literature (Boctor, 1990; Icmeli and Erenguc, 1996b; Hartmann and Drexl, 1998) and used in practice. In some situations due dates for individual activities play an important role and the explicit consideration of the lateness or tardiness of the different activities is needed. Often the project's deadline and or interim deadlines are associated with costs as in penalties, bonuses for quick execution or planned cash inflows, giving to the time objective a connection to the cost dimension.

The project scope, defined as "the work that needs to be accomplished to deliver a product, service, or result with the specified features and functions" (PMI, 2012) does not lead to a specific objective for the schedule as it marks out the boundaries of a completed project as opposed to a failed (not resulting in the desired outcome) or incomplete one. Therefore it is mandatory to have in the project schedule all the activities, as they arise from the scope declaration and are outlined in the work breakdown structure. In other words the project's schedule activities are a specification of the project's scope.

Quality is defined as "the total of features and characteristics of a product or service that bears on its ability to satisfy stated or implied needs" (Standardization, 1994). Although the quality of the project is monitored through the quality management process, it can affect the project's schedule as analysed by (Icmeli and Erenguc, 1996a). Activities may be performed at a poor quality, below the desired level. These activities will require additional rework time, resulting in delaying the project completion time or if left as they are, lead to poor quality of the project's outcome. It should be noted here that rework is considered by several authors as the primary cause of project schedule disruptions (Lewis, 1998; Cooper, 1993). However the quality of the activities cannot be appraised before the actual execution of the activity so cannot be used as a direct requirement of the schedule. A project schedule that can handle the distress caused by the need to rework activities as to reach the desired level of quality is a realistic way of accommodating the need for specific quality levels of the end product with respect to the duration aspect of the project. Therefore to have the ability to address the needed quality leads to the objective of robustness maximisation and possible constraints on the quality of a subset of activities.

Cost and budget are facets of a complex requirement referring to the project in general and strongly related with the project schedule itself. The cost refers to resource usage, resource availability cost, earliness/tardiness penalties. When the maximum cost is predefined there is a clear budget constraint. Otherwise the need of minimal cost can be expressed as an objective. The majority of activities encountered in practice can be performed in shorter or longer durations by increasing or decreasing the amount of resources available to them or the quality of the used resources (Fulkerson, 1961). However, usually, this acceleration in the execution of activities comes at a cost and at the same time accomplishing activities in longer durations often gives reduced cost but can lead to increase in project duration which can result in additional cost in form of time related penalties (Kelley Jr and Fort Washington, 1963).

Furthermore, there are cases where penalties or other cost raising factors are attached to specific activities deadlines and not only to the project's makespan (Vanhoucke et al., 2001). For example, costs of earliness can reflect extra storage requirements and idle time like tardiness leads to customer complaints, loss of reputation and profits, monetary penalties or goodwill damages. The resource related costs can origin not only from the resource usage itself but also from the need to keep resources available for the project duration even when they are



not being used (Mahring, 1984; Demeulemeester and Herroelen, 1996) as in the case of time based contracts. This leads to the need of minimisation of the maximum number of needed resources per type, usually weighted by a factor related to their individual cost.

Therefore, the need of minimal cost is reflected in a variety of project scheduling objectives, like the project's makespan, earliness/tardiness of activities, minimal maximal usage of resource units made available throughout the project besides the cost itself. In most cases when scheduling a project aiming at the cost minimisation the only costs that are taken into consideration are the resource related costs and the time related penalties, as those are the directly connected costs. The precise determination of these gains and penalties often poses a tough problem to management that needs to prioritise these conflicting objectives.

The scheduling problem turns even more complicated by the need to manage human resources and materials. Human resources either staff or external contractors, based on the legislation and their individual contracts should work specific hours per day augmented by specific overtimes constraining the amount of daily work per person, have vacation and sickness leaves causing usually small variations in resource demands that not always are known in advance and variable performance affected by a variety of parameters, like motivation, work environment, etc. Furthermore, it is not easy and it is not very efficient to hire staff for small periods to handle peak of work and fire them after that or even move team members from one project to another too often as it can lower the project team's productivity by requiring time both from the existing members and the newcomers to teach/learn the way that the work should be done, learning curve, and also become acquainted to each other. Therefore, a smooth resource profile, for human resources is essential and strongly related to the complexity and specialisation needed to execute the task, as the more complex is the task the less changes in resources used is desired.

Materials as resources are less complex than human resources to handle but there are also some points that attention is needed, especially when deliveries are connected to tasks initiation as usually it is the case and storing the material is needed till its usage, as storing comes at a cost, leading to dilemmas on how earlier should the delivery be scheduled as to lower the risk of delaying the task in case of belated delivery but without creating a large additional cost for storage.

Uncertainty lies at the very heart of any project leading to very low probability to have a precomputed baseline schedule being executed exactly as planned. Activities may take more or less time than originally estimated, resources may become unavailable, material may arrive behind schedule, new activities may have to be incorporated or activities may have to be dropped due to changes in the project scope, etc. An apparently optimal baseline schedule may well be based on an unreasonable set of expectations about the real world and therefore may be significantly less optimal when executed. Similarly, a baseline schedule which appears less optimal before execution but which contains some built-in flexibility for dealing with unexpected events, may turn out to be a good schedule upon execution (Davenport et al., 2004). A safe route, when working in a deterministic environment, to absorb uncertainties is the development of robust schedules that are schedules in which a delay has only a limited effect due to the usage of techniques for absorbing the delays so as to cause minimal effect to the rest of the project schedule.

Summarising, the project schedule generation problem seeks a baseline schedule for an upcoming project where time, cost, smooth resource profile, minimal maximum usage per resource type and robustness are the core objectives. Project managers always reason in terms of a mix of the above objectives, therefore, a multidimensional approach is implicitly or explicitly taken in practice (Viana and Pinho de Sousa, 2000). These different aspects, are often conflicting and all of them need to be taken into consideration and will play different role in

the schedule generation process based on the specific organisation and its priorities, the size and the budget of the project, the customer and other environmental parameters.

#### ***4.2.2 Available Inputs and Constraints***

A project schedule is a plan that defines which activity should be executed, when should its execution start and what amount of resources per resource type will be used. The major components of a schedule are the activities and the resources.

An activity can have one or more execution modes, meaning that it can be executed using various resource types and amounts of resources, resulting in different durations. Each mode reflects a feasible way to combine a duration and resource requests that allow accomplishing the underlying activity. Multiple modes of execution assume that either more resources of the same type or more efficient types of resources are used to get a shorter execution time. In all cases, the values of both duration and resource requirements, are estimates based on the past project's experience and the current situation characteristics.

The activities can be splittable or not based on the specifics of the task. Splits can happen in predefined points of time or in any time period, for example, a task like a software module's development could be split at definite time instances corresponding to the sub-modules completion but it wouldn't be wise to split it in the middle of a complex function. An obvious way to avoid using splittable activities is to split the activity itself in sub-activities and add constraints to keep the chain of events intact. This actually, is the way that splittable activities are handled when their number is limited. However, by using splittable activities the size of the project remains the same, helping the project manager during the monitoring and in case of activities that can be splittable at virtually any time, the manual process would be rather copious while it is straightforward its automation and gives to the scheduling mechanism a much needed grade of freedom letting the essence of the schedule intact.

The activities can either require constant amounts of renewable resources, that is, the per-period request for a resource remains unchanged until the activity has been completed or the resource requests can vary with time. However, it is not very probable that all activities will have variable demands so a subset of activities with variable demands should be defined. Again splitting the activities in sub-activities would do the trick but would cost in complexity and readability of the schedule along with increase of the probability of errors during the setting up of the schedule, as a number of new activities and constraints should be manually added and never removed or relaxed by mistake.

To start executing an activity all its immediate predecessors should have been finished. This precedence concept in practical situations is extended by allowing start-start, finish-start, start-finish and finish-finish precedence constraints with both minimal and maximal time lags. Obviously, finish-start constraints are more often used but the scheduler should not be limited by the model. It is the other way round, the model should try to accommodate as many facets of the problem as possible and let the scheduler decide what level of abstraction is needed in each case.

Three different kinds of resources are considered: renewable, non-renewable and doubly constrained. Renewable are limited on a per period basis, non-renewable have a limited capacity for the entire project and doubly constrained are limited both for each period and for the whole project. In special cases we can have dedicated resources that refer to resources that can be assigned to one activity at a time and can be represented by renewable resources with one unit of per period availability. Each resource type comes with a cost either per time pe-

riod of work (e.g. daily cost) or in form of salary. There are two basic types of contractual agreements that can lead to great difference on the way that the resources should be handled. In case of resource types having a contract for the whole period of the project or more generally get a monthly salary then there is the need to make the most of their availability and the issue is the smoothness of the resource profile, as they already have a contract and will be available for the project duration so its no point in limiting their usage. On the contrary, the number of resources on time contracts, should be minimised as the contracts can fit the project needs so that the resource profile is not as important as the number of the needed contractors.

Resource availabilities usually are assumed to be constant over time. This assumption is not very close to what actually happens in practical cases where changing availability of workers due to vacations, maternity leaves, sickness or varying equipment capacities due to maintenance or damage are on the everyday schedule.

The constraints posed by the problem can be roughly grouped in two categories, those induced by the problem's logic and those induced by requirements that constitute the problem's definition. In the former are the precedence constraints either explicit like those set to define the sequence of the activities or generated when handling splits and variable resource requests. In the latter are the project cost/budget, resource availabilities and deadlines of activities that although should be obeyed under extreme circumstances the project manager can decide to allow their relaxation in order to be able to solve the problem or even simply get a better solution under some aspects.

Concluding, the organisation's stakeholders, represented by the group of decision makers (i.e. project manager or project team members), are responsible for the initialisation of the process by setting the criteria and the priorities of the objectives to be pursued during the scheduling process. This way the conflicting opinions can be expressed during the initial decision making process, including both qualitative and quantitative factors related to the specific project and thus giving a way of custom tailoring the project scheduling process to the specific needs and circumstances. It should be noted here that it is crucial to be able to have a general decision model that can be easily modified to reflect the changes in the organisation's strategic priorities but also can be repeatedly used in similar cases without the need of continuous redesign.

Following, based on estimations of the needed input data a calculation process is followed to generate one or more schedules. The input data, as defined in section 4.2.2, like activities duration, resource requirements and resource availabilities in real cases it is highly probable that will not be deterministic but we assume them deterministic. However, multiple executions using upper and lower bounds of estimations or even using multiple scenarios and fast execution of the schedule generation process, are suggested to overcome the issue and get more realistic results. Finally, the generated schedules are presented to the decision makers/project manager and the most proper baseline schedule will be selected and used as a tool for monitoring the project's execution.



## Chapter 5

# Proposed Holistic Mathematical Model

### 5.1 Proposed Problem Formulation

The proposed process is initialised by a group of decision makers in charge of the project. This group based on estimations of the needed input data selects the type or mix of types of solution scenarios among single objective, weighted and simple Pareto optimal schedules that they would like to get as a result. Then a calculation process is followed to generate the requested number of schedules. Although in real cases the input data, like activities duration, resource requirements and resource availabilities, will not, most of the times, be deterministic it is assumed that multiple executions using upper and lower bounds of estimations can cover satisfactorily the issue. Finally, the generated schedules are presented to the decision makers and the most proper baseline schedule is selected and used as a tool for monitoring the project's execution.

#### 5.1.1 Definitions

The proposed variation of the resource constrained project scheduling problem may be conceptually formulated as follows.

- All data is assumed to be deterministic and known in advance.
- There is a single project consisting of  $n$  activities plus a dummy source activity 0 representing the “project start” and a dummy sink activity  $n + 1$  representing the “project end” both with zero duration and resource requirements. We will denote by  $V = \{0, 1, \dots, n, n + 1\}$  the set of all activities.
- $T$  is the planning horizon, calculated as the sum of maximal durations of all activities.
- The set of renewable resources will be denoted by  $R^p$ . For each renewable resource  $k \in R^p$  the per period availability is variable and is denoted by  $\alpha_{kt}^p, t = 0, 1, \dots, T - 1$ .
- The set of non-renewable resources will be denoted by  $R^v$ . With each non-renewable resource  $l \in R^v$  we associate a subset  $\{t_{lx} | x = 0, \dots, X_l\}$  of  $\{0, 1, \dots, T\}$  with

$$0 = t_{l0} < \dots < t_{lx} < t_{l(x+1)} < \dots < t_{lX_l} = T.$$

Obviously this subset defines a partition of the interval  $[0, T)$  consisting of the subintervals  $I_{lx} = [t_{lx}, t_{l(x+1)}), x = 0, \dots, X_l - 1$ . The overall consumption of the non-renewable resource  $l$  for the period  $I_{lx}$  of the project is limited by  $\alpha_{lI_{lx}}^v$ .

- Each activity  $i$  is associated with a set  $M_i$  of modes which are alternatives ways of executing the activity.
  - Each activity  $i$  has to be performed in exactly one mode  $m \in M_i$  in each discrete schedule.
  - Each mode  $m$  has a duration of  $d_{im}$  time units.
  - Activity  $i$  in mode  $m$  requires  $r_{imk}^p \tau_{im}$  renewable resources of type  $k \in R^p$  in the  $\tau_{im}$ -th period of its execution,  $\tau_{im} = 0, \dots, d_{im} - 1$ . The required resources are not consumed but used for the time period of the activities duration and then returned to the resource pool.
  - Activity  $i$  in mode  $m$  requires the consumption of  $r_{iml}^v \tau_{im}$  non-renewable resources of type  $l \in R^v$  in the  $\tau_{im}$ -th period of its execution,  $\tau_{im} = 0, \dots, d_{im} - 1$ .
- Each mode  $m$  of an activity  $i$  defines the activity either as preemptive, i.e. its execution can be stopped once it has been started, or not.
  - The set of non-preemptive modes of an activity  $i$  is denoted by  $M_i^p$  and the set of its preemptive modes by  $M_i^p$ .
  - An activity in a preemptive mode can be splitted either at any point, which gives splits of unitary duration, or at specific, user defined, points. The definition of the split points can be done either directly or using duration windows.
  - The duration  $d_{im}$  of an activity  $i$  may be split in  $z_{im} + 1$  duration units of unitary or greater integer size. Each segment is denoted by  $p_{imq}$ ,  $q = 0, \dots, z_{im}$ , and has duration  $d_{imq}$ . Each segment is assigned a start time  $s_{imq}$  and a finish time  $f_{imq}$ .
  - To simplify the formulation all modes of all activities are considered to be preemptive. Activity modes that are not preemptive, e.g. the dummy start activity, will not have any split points, i.e.  $z_{im} = 0$ . If  $s_{imq}$ ,  $q = 0, \dots, z_{im}$ , are known, then we can transform the periods of the execution of activity  $i$  in mode  $m$ ,  $\tau_{im}$ , to periods  $t$  of the project as follows

$$t = \begin{cases} \tau_{im} + s_{im0}, & \tau_{im} = 0, \dots, d_{im0} - 1 \\ \tau_{im} + s_{im1}, & \tau_{im} = d_{im0}, \dots, d_{im1} - 1 \\ \vdots \\ \tau_{im} + s_{imz_{im}}, & \tau_{im} = d_{im(z_{im}-1)}, \dots, d_{imz_{im}} - 1 \end{cases} \quad (5.1)$$

- Based on the above:
  - $s_{im0}$  is the start time of activity  $i \in V$  in mode  $m$  and its first segment  $p_{im0}$ .  $f_{imz_{im}}$  is the finish/completion time of activity  $i \in V$  in mode  $m$  and its last segment  $p_{imz_{im}}$ .
  - The dummy source activity has one mode  $m = 0$ , duration of 0 time units and is not preemptive. Therefore  $z_0 = 0$  and  $s_{00z_0} = s_{000}$ . Consequently, setting the project to begin at time zero gives  $s_{000} = 0$ .
  - The dummy sink activity has one mode  $m = 0$ , duration of 0 time units and is not preemptive. Therefore  $z_{n+1} = 0$  and  $f_{(n+1)0z_{n+1}} = f_{(n+1)00}$  represents the project's duration or makespan.
- Four different types of precedence relations are defined: the start-to-start  $SS_{imjn}$ , the finish-to-finish  $FF_{imjn}$ , the finish-to-start  $FS_{imjn}$  and the start-to-finish  $SF_{imjn}$ , with minimal and maximal time lags between the activities  $i$  and  $j$  executed in modes  $m$  and  $n$  respectively. More specifically:
  - $SS_{imjn}^{min}$  denotes that activity  $j$  in mode  $n$  cannot begin earlier than  $SS_{imjn}^{min}$  time units after the start of activity  $i$  in mode  $m$ ,
  - $SS_{imjn}^{max}$  denotes that activity  $j$  in mode  $n$  cannot start later than  $SS_{imjn}^{max}$  time units after the start of activity  $i$  in mode  $m$ ,

- $SF_{imjn}^{min}$  denotes that activity  $j$  in mode  $n$  cannot finish earlier than  $SF_{imjn}^{min}$  time units after the start of activity  $i$  in mode  $m$ ,
  - $SF_{imjn}^{max}$  denotes that activity  $j$  in mode  $n$  cannot finish later than  $SF_{imjn}^{max}$  time units after the start of activity  $i$  in mode  $m$ ,
  - $FS_{imjn}^{min}$  denotes that activity  $j$  in mode  $n$  cannot begin earlier than  $FS_{imjn}^{min}$  time units after the finish of activity  $i$  in mode  $m$ ,
  - $FS_{imjn}^{max}$  denotes that activity  $j$  in mode  $n$  cannot start later than  $FS_{imjn}^{max}$  time units after the finish of activity  $i$  in mode  $m$ ,
  - $FF_{imjn}^{min}$  denotes that activity  $j$  in mode  $n$  cannot finish earlier than  $FF_{imjn}^{min}$  time units after the finish of activity  $i$  in mode  $m$ ,
  - $FF_{imjn}^{max}$  denotes that activity  $j$  in mode  $n$  cannot finish later than  $FF_{imjn}^{max}$  time units after the finish of activity  $i$  in mode  $m$ .
- After fixing the activities' durations and time lags, all the relations are represented using just one type, the SS which is arbitrarily selected, using the following transformation rules:

Start to Start:

$$s_{im0} + SS_{imjn}^{min} \leq s_{jn0} \rightarrow s_{im0} + \delta_{imjn} \leq s_{jn0},$$

with  $\delta_{imjn} = SS_{imjn}^{min}$ ,

$$s_{im0} + SS_{imjn}^{max} \geq s_{jn0} \rightarrow s_{jn0} + \delta_{jnim} \leq s_{im0},$$

with  $\delta_{jnim} = -SS_{imjn}^{max}$ .

Start to Finish:

$$s_{im0} + SF_{imjn}^{min} \leq f_{jnz_jn} \rightarrow s_{im0} + \delta_{imjn} \leq s_{jn0},$$

with  $\delta_{imjn} = SF_{imjn}^{min} - d_{jn}$ ,

$$s_{im0} + SF_{imjn}^{max} \geq f_{jnz_jn} \rightarrow s_{jn0} + \delta_{jnim} \leq s_{im0},$$

with  $\delta_{jnim} = -(SF_{imjn}^{max} - d_{jn})$ .

(5.2)

Finish to Start:

$$f_{imz_{im}} + FS_{imjn}^{min} \leq s_{jn0} \rightarrow s_{im0} + \delta_{imjn} \leq s_{jn0},$$

with  $\delta_{imjn} = FS_{imjn}^{min} + d_{im}$ ,

$$f_{imz_{im}} + FS_{imjn}^{max} \geq s_{jn0} \rightarrow s_{jn0} + \delta_{jnim} \leq s_{im0},$$

with  $\delta_{jnim} = -(FS_{imjn}^{max} + d_{im})$ .

Finish to Finish:

$$f_{imz_{im}} + FF_{imjn}^{min} \leq f_{jnz_jn} \rightarrow s_{im0} + \delta_{imjn} \leq s_{jn0},$$

with  $\delta_{imjn} = FF_{imjn}^{min} + d_{im} - d_{jn}$ ,

$$f_{imz_{im}} + FF_{imjn}^{max} \geq f_{jnz_jn} \rightarrow s_{jn0} + \delta_{jnim} \leq s_{im0},$$

with  $\delta_{jnim} = -(FF_{imjn}^{max} + d_{im} - d_{jn})$ .

- When the structure of the project is represented by an activity-on-node network  $G = (V, A)$ , then the vertex set  $V = \{0, 1, \dots, n, n+1\}$  contains all activities and the set of arcs  $A = \{(i, j) | i, j \in V; i \rightarrow j\}$  represents the generalised precedence constraints. More precisely, there will be an arc from node  $i$  to node  $j$  if and only if there are one or more prece-

dence relations between the two nodes. If  $(i, j) \in A$ , then, for each pair  $(m, n) \in M_i \times M_j$ , the maximum value of  $\delta_{imjn}$  in (5.2) is assigned as weight to the edge  $(i, j)$ .

- The vector  $S = (s_{imq})_{i=0,1,\dots,n,n+1}^{q=0,\dots,z_{im}}$  defines a schedule of the project. A schedule  $S$  is called feasible if all resource and generalised precedence constraints are fulfilled.
- $Act(t)$  will denote all the activities of which a time unit is in progress at  $t$ ,  $t = 0, 1, \dots, T$ . The goal is to determine execution modes  $m$  and starting times  $s_{imq}$  for all the activities  $i = 1, \dots, n$  and all  $q = 0, \dots, z_{im}$  in such a way that the objectives are optimised while all the given constraints are obeyed.

Table 5.1 summarises the notation introduced in this section.

**Table 5.1** Basic Notation

Symbol	Definition
$V = \{0, 1, \dots, n, n+1\}$	the set of activities $i$
$n$	number of real activities
$G(V, A)$	directed graph of precedence or temporal constraints
$T$	the planning horizon, sum of maximal durations of all activities
$t$	periods, index of $T$
$[t, t+1)$	time interval corresponding to period $t$
$Act(t)$	set of all the activities of which a time unit is in progress at $t$ , $t = 0, 1, \dots, T$
$R^P$	set of renewable resources
$\alpha_{kt}^P$	variable amount of available units of renewable resource $k$ , $t = 0, \dots, T-1$
$R^V$	set of non-renewable resources
$l_x$	each non renewable resource $l \in R^V$ is associated to a subset $\{t_{lx}   x = 0, \dots, X_l\}$ of $\{0, 1, \dots, T\}$ with $0 = t_{l0} < \dots < t_{lx} < t_{l(x+1)} < \dots < t_{lX_l} = T$
$I_{lx}$	subintervals $I_{lx} = [t_{lx}, t_{l(x+1)})$ , $x = 0, \dots, X_l - 1$ composing a partition of $[0, T)$
$\alpha_{I_{lx}}^V$	variable amount of available units of non-renewable resource $l$
$M_i$	set of modes (alternative ways of execution) of activity $i$
$M_i^P$	set of non-preemptive modes of activity $i$
$M_i^P$	set of preemptive modes of activity $i$
$d_{im}$	duration of activity $i$ in mode $m$
$r_{imk}^P$	per period usage of activity $i$ of renewable resource $k$ in mode $m$
$r_{iml}^V$	per period consumption of activity $i$ of non-renewable resource $l$ in mode $m$
$z_{im}$	number of splits on activity $i$ in mode $m$ , $z_{im} = 0, \dots, d_{im} - 1$
$p_{imq}$	segment of the preempted activity $i$ with $q = 0, 1, 2, \dots, z_{im}$
$d_{imq}$	duration of segment $q$ of activity $i$ in mode $m$
$s_{imq}$	start time of segment $q$ of activity $i$ in mode $m$
$f_{imq}$	finish time of segment $q$ of activity $i$ in mode $m$
$s_{im0}$	start time of activity $i$
$f_{imz_m}$	finish time of activity $i$
$s_{000}$	start time of project
$f_{(n+1)00}$	finish time of project
$S = (s_{imq})$	schedule, vector of start times of all segments of all activities
$SS_{imjn}^{min}/SS_{imjn}^{max}$	minimum/maximum time lag between start of activities $i$ and $j$ in modes $m$ and $n$
$SE_{imjn}^{min}/SE_{imjn}^{max}$	minimum/maximum time lag between start of activity $i$ in mode $m$ and finish of $j$ in mode $n$
$FS_{imjn}^{min}/FS_{imjn}^{max}$	minimum/maximum time lag between finish of activity $i$ in mode $m$ and start of $j$ in mode $n$
$FF_{imjn}^{min}/FF_{imjn}^{max}$	minimum/maximum time lag between finish of activities $i$ and $j$ in modes $m$ and $n$
$\delta_{imjn}$	minimum/maximum time lag between start of activities $i$ and $j$ in modes $m$ and $n$

We will now illustrate the above definitions using the project example displayed in Figures 5.1 - 5.3, where a project with 6 activities plus the dummy source, activity 0 and the dummy



id	mode	dur	R1			R2			NR1		Preemption
			$r^p_{im,k_1,\tau_{im}}$	$r^p_{im,k_2,\tau_{im}}$	$r^p_{im,k_3,\tau_{im}}$	$r^p_{im,k_1,\tau_{im}}$	$r^p_{im,k_2,\tau_{im}}$	$r^p_{im,k_3,\tau_{im}}$	$r^v_{im,l_1,\tau_{im}}$	$r^v_{im,l_2,\tau_{im}}$	
$l$	$m_i$	$d_{im}$	$t_{i1}$	$t_{i2}$	$t_{i3}$	$t_{21}$	$t_{22}$	$ta_1$	$ta_2$	Y/N	
0	0	0	0	0	0	0	0	0	0	0/N	
1	0	5	3	2	1	2	1	1	2	1/N	
1	1	9	2	1	1	1	2	2	2	1/N	
2	6	1	2	3	1	2	3	2	3	2/N	
2	0	5	1	1	1	1	2	1	1	3/N	
2	1	7	2	0	2	1	2	1	2	2/Y	
2	9	0	2	2	2	1	1	1	1	1/Y (30%)	
3	0	4	1	2	1	4	1	2	4	4/N	
1	5	2	1	2	2	4	2	2	1	1/N	
2	5	1	1	3	1	2	2	2	2	2/Y (20%)	
4	0	5	2	1	2	1	2	3	2	2/Y (50%)	
1	4	1	2	2	2	1	3	2	1	2/N	
2	4	1	1	2	1	2	4	2	1	2/N	
5	0	5	2	1	2	1	3	2	3	3/N	
1	8	2	1	2	1	3	1	2	1	2/Y (25%)	
2	4	2	1	2	3	1	1	1	1	3/N	
6	0	4	1	2	1	2	1	3	1	1/N	
1	5	1	2	1	1	2	3	2	1	2/Y (20%)	
2	8	1	2	1	2	1	2	2	1	2/Y (75%)	
7	0	0	0	0	0	0	0	0	0	0/N	

id	m	0	1	2	3	4	5	6	7								
m	0	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0
0	0	0		0													
1	0				3	2	1										
1	1				6	4	1										
2	2				1	2	3										
2	0						4	4	1								
1	1						1	1	4								
2	2						2	2	1								
3	0								1	1	2						
1	1								2	2	1						
2	2								4	1	2						
4	0			-4	-4	-6						0	4	1			
1	1			-2	-1	-2						2	1	7			
2	2			-6	-2	-2						1	1	4			
5	0																2
1	1																2
2	2																1
6	0																1
1	1																1
2	2																4
7	0																0

Fig. 5.1 (a) Project activities, modes, resource requirements and preemption status, (b) minimal and maximal lag of activities per mode

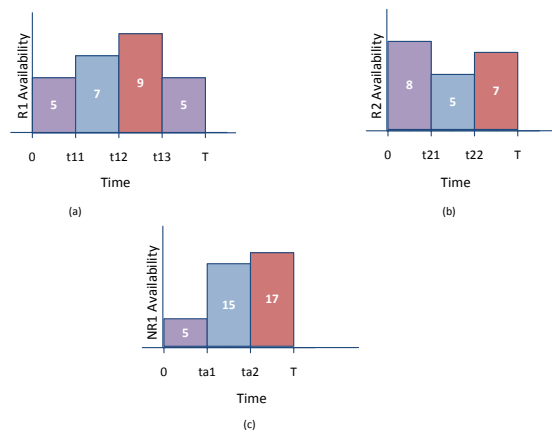


Fig. 5.2 (a) R1 renewable resource availability in relation to time, (b) R2 renewable resource availability in relation to time and (c) NR1 non-renewable resource availability in relation to periods  $I_{l0} = [0, t_{a1})$ ,  $I_{l1} = [t_{a1}, t_{a2})$  and  $I_{l2} = [t_{a2}, T)$

sink, activity 7, is defined. Each activity, in this example project, has a maximum of 3 alternative modes of execution. For each mode the duration, preemption status and resource requirements are defined.

The duration refers to the total of the activity even when preemption is allowed. The preemption status determines whether the activity in the specific mode of execution is splittable and, if yes, what kind of split should be effectuated, as it can be either auto split in segments with 1 time period of duration e.g. activity 6 in mode 1 or at specific points as in activity 2, mode 2. The resource requirements per activity mode can vary over time and therefore the resource requirements of both renewable and non-renewable resources are defined for each time period. However the time instances where there is a change in requirements can vary from activity to activity or even among the different modes of the same activity, but for reasons of simplicity in this example we use the same time instants.

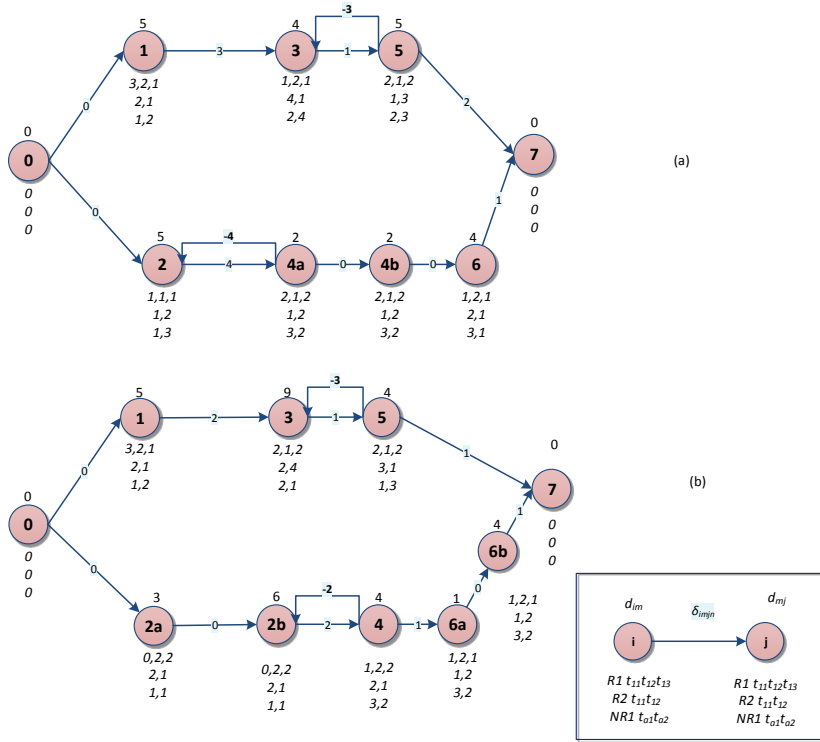


Fig. 5.3 (a)Project network for mode set  $M_1(0,0,0,0,0,0,0,0)$  and (b) Project network for mode set  $M_2(0,0,2,1,1,2,1,0)$

Furthermore, each resource type, either renewable or not, can also have variable availability, as shown in Figure 5.2. This feature can be used in practice to define the renewable resources calendar by denoting for each resource type, the time periods that availability changes due to scheduled vacations, weather related impossibility to use specific types of equipment/machines, etc. In the case of non-renewable resources attention should be given in defining the availability of each resource type. For example, at  $t_{a2}$  there would be 15 units of NR1 if no unit would have been used by that time.

Finally, in Figure 5.3, the project networks for two different, random, combinations of modes are presented:  $M_1(0,0,0,0,0,0,0,0)$  and  $M_2(0,0,2,1,1,2,1,0)$ . It can be deduced that the project network itself and not only the durations and resource requirements are affected by each mode selection. This way alternative project scenarios and not only activities execution modes are defined. Each of the resulting combinations of modes gives a more or less complex project scheduling problem that represents a different aspect of the project and how it should be implemented.

### 5.1.2 Objectives

In the project schedule generation problem we seek a baseline schedule for an upcoming project where time, cost, smooth resource profile, minimal maximum usage per resource type and robustness are the core objectives.

The project duration is a regular measure of the performance measure and can be expressed as the start of the dummy end activity of the project:

$$\min s_{(n+1)0}. \quad (5.3)$$

Furthermore, a penalty factor can be used in case that there are specific deadlines attached to activities (e.g. milestones) of the project:

$$\min s_{(n+1)0} + T_{over}, \quad (5.4)$$

where  $T_{over}$  is the sum of the time periods that the completion of each activity is overdue. The penalty factor can be weighted by the criticality of the corresponding activity (e.g. costs related to the delay of specific milestones):

$$T_{over} = \sum_{i=1}^n w_i (T_i - f_{imz_{im}}), \quad (5.5)$$

where  $T_i$  is the due date (deadline) of activity  $i$ .

The resource related objectives concern the reduction of extraordinary demands and excessive fluctuations in the usage of resources. When the goal is to use the required resources as even as possible over time, the deviations of the resource usages from a given resource profile are calculated. A measure of variability for this case is the resource levelling index (RLI), where the request for a smooth resource profile of one or more resource types is expressed as the total deviation of the consumption of that resource type from a target value, as it is the average resource utilisation:

$$\min \sum_{k \in R^p} \sum_{t=0}^{f_{(n+1)00}} \left| \left( \sum_{i=1}^N r_{imk}^p \tau_{im} \right) - \frac{\sum_{t=0}^{f_{(n+1)00}} \sum_{i=1}^N r_{imk}^p \tau_{im}}{f_{(n+1)00}} \right|. \quad (5.6)$$

Note that, both in the above equation and in what follows, by  $\tau_{im}$  we mean the corresponding value of  $t$  given by (5.1).

If  $c_k$  is the unitary cost related to renewable resource type  $k$  and  $c_0$  is the sum of the non resource related costs then the project cost can be expressed as the sum of these two cost types:

$$\min c_0 + \sum_{k \in R^p} c_k \left( \sum_{t=0}^T \sum_{i=1}^N r_{imk}^p \tau_{im} \right). \quad (5.7)$$

In case of specific budget, it can be set either as constraint or as a penalty in the objective function, as in the case of the deadlines.

The reduction of extraordinary demands of one or more resource types, the so-called resource investment problem, is used for those resources that are very expensive and even one unit of difference has a consistent impact on the project's cost. Therefore for these resource types the maximal usage should be minimised:

$$\min \max \left\{ \sum_{i=1}^N r_{imk}^p \tau_{im} \mid t = 0, 1, \dots, T-1 \right\}. \quad (5.8)$$

The slack-based method is usually used to generate robust schedules. Two types of slack are widely used in the scheduling literature: total slack (also referred to as float time) that is the difference between the earliest start time and latest start time of an activity and free slack that is the amount of time that an activity can be delayed without delaying the start of the very next activity. In the current approach the total slack is set as objective:

$$\max \sum_{i=1}^n (LS_{im} - ES_{im}), \quad (5.9)$$

where  $ES_{im}$  is the earliest start time and  $LS_{im}$  is the latest start time of activity  $i$  when executed in mode  $m$ —in the case of splitting the start time of the first segment is taken into consideration. It should be noted here that  $ES_{im}$ ,  $LS_{im}$  calculations are not trivial in the multi-mode preemptive with generalised precedence constraints case.

All the above different aspects are often conflicting. On the other hand all of them need to be taken into consideration and will play different roles in the schedule generation process based on the specific organisation and its priorities, the size and the budget of the project, the customer and other environmental parameters.

### 5.1.3 Constraints

The activities should be processed in a specific order given by the generalised precedence constraints, which can be represented in standardised form, by transforming all of them to the same, arbitrarily selected, form using the transformation rules in Equations (5.2), leading to the general constraint:

$$\begin{aligned} \delta_{imjn} &\leq s_{jn0} - s_{im0}, \\ \forall (i, j) \in A, \forall m \in M_i, \forall n \in M_j. \end{aligned} \quad (5.10)$$

All activities when executed in preemptive mode should have the start time for every segment  $q$  of the preempted activity  $i$ , at least  $d_{im(q-1)}$  time units later than the start time for the previous segment  $q - 1$ :

$$\begin{aligned} s_{im(q-1)} + d_{im(q-1)} &\leq s_{imq}, \\ \forall i = 1, \dots, n, \forall m \in M_i, \forall q = 1, \dots, z_{im}. \end{aligned} \quad (5.11)$$

This constraint also addresses the need to relate each segment to the previous one with a finish to start type of precedence constraint with zero minimal and no maximal time lag. Furthermore, the dummy source activity should always be the first to be scheduled at time  $t = 0$ :

$$s_{000} = 0, \quad (5.12)$$

and the dummy sink activity should always be the last one:

$$\begin{aligned} s_{(n+1)00} &\geq f_{imz_{im}}, \\ \forall i = 0, 1, \dots, n, \forall m \in M_i. \end{aligned} \quad (5.13)$$

The renewable and non-renewable resource usage at each time instant  $t$  should be less or equal than the available amount:

$$\sum_{i \in Act(t)} r_{imk}^p \tau_{im} \leq \alpha_{kt}^p, \quad (5.14)$$

$$\forall k \in R^p, \forall t = 0, 1, \dots, T-1, \forall m \in M_i,$$

$$\sum_{t=0}^{t_l(x+1)-1} \sum_{i \in Act(t)} r_{iml}^v \tau_{im} \leq \alpha_{l|x}^v, \quad (5.15)$$

$$\forall l \in R^v, \forall x = 0, \dots, X_l - 1, \forall m \in M_i.$$

In the case that milestones with specific deadlines are introduced, then each of the milestones should have its start time constrained by its due date or a penalty factor can be used in the objective function. In the proposed approach penalties are used to give more flexibility during the solution search.

## 5.2 Mathematical Formulation

The proposed variation of the resource constrained project scheduling problem can be mathematically formulated introducing the binary decision variables  $x_{imqt}$  which are defined as follows:

$$x_{imqt} = \begin{cases} 1, & \text{if the segment } p_{imq} \text{ of } i \text{ in mode } m \text{ starts at } t \\ 0, & \text{otherwise} \end{cases}. \quad (5.16)$$

The mathematical formulation, shown in Equations (5.17)-(5.25), is an extension of the model first presented by Pritsker et al. (1969) to include preemption, multimode activities and generalised constraints. It is also based on the formulations provided by De Reyck (1999) and Hartmann (2013) for the MRCPSp-GPR and the RCPSP/t respectively.

$$\min f(x_{imqt}), \quad (5.17)$$

subject to:

$$\sum_{m \in M_i} \sum_{t=0}^{T-1} \sum_{q=0}^{z_{im}} x_{imqt} = 1, \quad (5.18)$$

$$\forall i = 0, 1, \dots, n+1,$$

$$\left( \sum_{m \in M_i} \sum_{t=0}^{T-1} t x_{im0t} \right) + \delta_{imjn} \leq \sum_{n \in M_j} \sum_{t=0}^{T-1} t x_{jn0t}, \quad (5.19)$$

$$\forall (i, j) \in A,$$

$$\sum_{i=1}^n \sum_{m \in M_i} \sum_{q=0}^{z_{im}} r_{imk}^p \tau_{im} x_{imqt} \leq \alpha_{kt}^p, \quad (5.20)$$

$$\forall k \in R^p, \forall t = 0, 1, \dots, T-1,$$

$$\sum_{t=0}^{t_{l(x+1)}-1} \sum_{i=1}^n \sum_{m=1}^{m_{i\lambda}} \sum_{q=0}^{z_{im}} r_{iml}^v \tau_{im} x_{imqt} \leq \alpha_{l_{lx}}^v, \quad (5.21)$$

$$\forall l \in R^v, \forall x = 0, \dots, X_l - 1,$$

$$x_{0000} = 1, \quad (5.22)$$

$$\sum_{t=0}^{T-1} tx_{imz_{im}t} \leq \sum_{t=0}^{T-1} tx_{(n+1)00t}, \quad (5.23)$$

$$\forall i = 0, 1, \dots, n,$$

$$\sum_{t=0}^{T-1} tx_{im(q-1)t} + d_{imq} \leq \sum_{t=0}^{T-1} tx_{imqt}, \quad (5.24)$$

$$\forall i = 0, 1, \dots, n, \forall m \in M_i, \forall q = 1, \dots, z_{im},$$

$$x_{imqt} \in \{0, 1\}, \quad (5.25)$$

$$\forall i = 0, 1, \dots, n+1, \forall m \in M_i, \forall t = 0, 1, \dots, T-1.$$

The objective function (5.17) minimises the selected objective; for example the objective of minimising the makespan can be written as

$$\min \sum_{t=0}^{T-1} tx_{(n+1)00t}.$$

Constraints (5.18) ensure that each activity is assigned exactly one mode and exactly one start time.

Constraints (5.19) denote the generalised precedence relations with minimal and maximal time lags, where the actual values for each time lag, whether they originate from a minimal time lag or a maximal time lag are given by  $\delta_{imjn}$ .

The resource constraints are given in Equations (5.20) and (5.21) for renewable and non-renewable resources, respectively.

Equations (5.22) and (5.23) ensure that the first activity of the schedule is the dummy source and the last the dummy sink.

Constraints (5.24) ensure that the splitted activities will be executed in the correct order.

Equation (5.25) forces the decision variables to assume binary values.

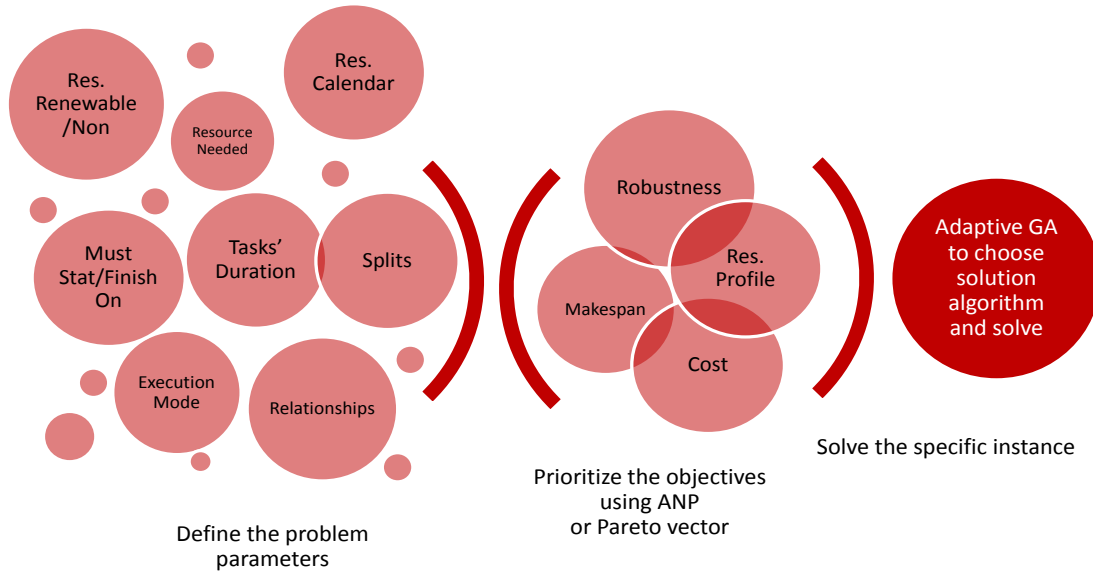
## **Chapter 6**

### **Solution Process**

#### **6.1 Overview**

The proposed solution approach consists of three stages, as shown in Figure 6.1. Initially, the problem is analysed to decide which are its activities and how they are related in terms of execution precedences. Then, for each activity the different execution modes and whether it can be split and at which points, are identified and the duration and resource requirements are estimated. The estimations usually are based on the project's manager experience on similar projects and the specific's project's characteristics. Finally, the resources calendars and availabilities are roughly outlined. The above information composes the solution's process input.

The second stage concerns the objectives. It is decided which objectives from the given set: makespan, resource level index, max resource usage and robustness, should be pursued and whether the objectives should be ranked or used to compose a vector and go for a Pareto solution, as shown in Figure 6.2. The first option leads to the generation or usage, if a well-fitted model already exists, of an ANP model and the calculation of the ranking through an iterative process where the project manager or a whole group of decision makers give judgments by answering simple pairwise comparison questions. The ANP model is formed by the alternative solutions, that are the selected objectives, and the criteria needed to rank them. Therefore the choice of ANP as a decision support tool requires additional inputs.



**Fig. 6.1** Phases of the proposed approach

The last stage includes all the transformations needed to convert the given input to the form required by the solution algorithm and the execution of the solution algorithm in order to get the final result set.

The flow of events, from the definition of the project to be scheduled till the actual generation of the solution set, is summarised in Figure 6.2. Initially, the project to be scheduled along with its context is analysed in order to generate not only the actual input data for the solution algorithm but also the data needed to make the related decisions, like whether an ANP or Pareto approach should be followed and which are the criteria that should be used to prioritise the objectives and how they are related.

Following, input data are transformed as needed and fed to the actual solution algorithm that is a genetic algorithm that adapts the solution method to the specific project that is being scheduled. Each chromosome is composed of the encoded solution (the schedule), the decoding procedure (serial or parallel extended SGS) and the corresponding solution algorithm (SA, TS, PSO, GA). The generation of the initial and children populations through crossover and mutation and the final selection of the chromosomes to be passed to the next generation are handled by this algorithm. However, the fitness calculation depends on how the project manager decided to solve the problem (ANP, Pareto, single objective).



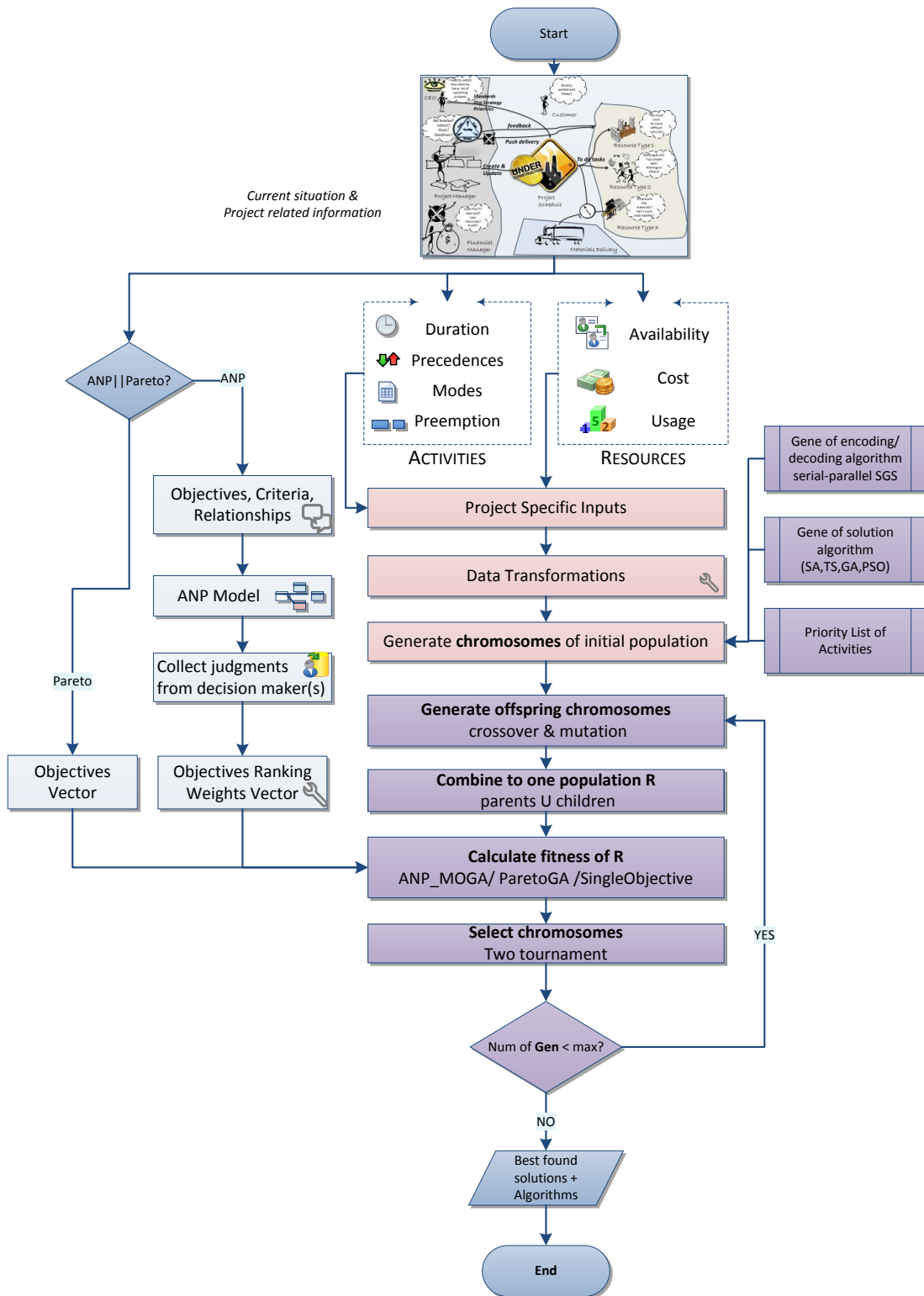


Fig. 6.2 Flow of events

In the case of ANP, the optimisation process is based on splitting the given population in sub-populations, using different objective functions for each one and then selecting the solutions by comparing subpopulations corresponding to the same objective using the ANP weights to define the percentage of the population that is represented by each objective. The second option, the Pareto vector, is based on the ranking of the population according to a predefined dominance rule and the fitness value assigned to each chromosome derives from its rank in the population, not its actual objective function value. Finally, the single objective option consists on using the given objective function to calculate the fitness of each chromosome. In all the cases, attention should be given on the fact that the fitness calculation of a chromosome is not a straightforward process, as it implies the execution of the solution algorithm on part of the chromosome, before calculating the value of the encoded solution, therefore it can cause update of the chromosome.

The iterative process continues until either the optimum is found or the given number of iterations (generations) is reached. The result set consists of one or more solutions that are the best found solutions in relation to the given objectives. The maximum number of different solutions in the result set is given by the project manager.

## 6.2 Decision Making using the Analytic Network Process

The second stage of the proposed approach consists in defining the desired optimisation objectives, namely time, resource profile, robustness and cost and either generate a weight vector reflecting the decision makers preferences on the objectives or handling all of them as a single vector. To weight the objectives, the Analytic Network Process (ANP) (Saaty, 1996), which is a generalisation of the Analytic Hierarchy Process (AHP), was used.

The reasons behind the selection of the specific method instead of other MCDA methods reside on its basic characteristics: simplicity, handling of mixed qualitative and quantitative inputs versus exclusively quantitative methods, no need of an analyst to support the process and focus on the subjective perspective of the decision maker to the problem.

In this section the ANP method is described and a generic model for weighting project scheduling objectives is provided. The result of this process is the definition of relative preferences among the optimisation objectives, in the form of a normalised weight vector that will be given as input to the next stage of the solution process. This a priori definition of the preferences has been proven more efficient than the popular process of solving the problem, computing the objective functions and then limiting the solution space based on the decision maker's preferences.

In this multi-criteria decision analysis method, the first step is the identification of the criteria and the alternative solutions of the problem to be solved. Then, a graph structure, the so-called network, is created and the decision maker is asked to pairwise compare the components, in order to determine their priorities.

The decision about whether AHP or ANP should be used is based on the problem being solved and the corresponding network structure. The network is a logical conceptualisation of the problem that reduces it, to its essentials. When the elements and their connections are easily located in levels of dominance with connections that transmit influence downwards, a hierarchical structure is best fitted for the decision problem. On the other hand, if the elements and their connections are complicated and can only be grouped in clusters that do not fit well in defined levels, a network structure is more appropriate.

The process that was followed for structuring the under consideration system, described in chapter 4, as a network consists of four phases:

- 1) Identification of the decision alternatives, goals and elements, where as elements are defined the criteria used to rank the different alternatives based on the preferences of one or more decision makers that constitute the decision making group. These criteria derive from the purpose of each functional unit of the system in question.

The project scheduling problem has as alternative solutions the optimisation objectives that have been chosen in the previous stage, therefore the elements:

- **Duration** that refers to the makespan of the schedule and respect of deadlines set for certain activities or phases of the project
- **Cost** as the sum of resource and non resource related costs generated by a specific schedule,
- **Resource Profile**, which refers to the smoothness of one or more renewable resource types profiles as they are utilised in the schedule,
- **Robustness** that is about the total free slack in the generated schedule and it is used to enhance the stability of the schedule in case of alterations: in the duration of one or more tasks, the resource availabilities and/or requirements.
- **Max Resource Usage**, which is used to limit the maximum usage of expensive or rare resource types.

Beside the alternative solutions of the problem, in the network are included all the criteria that will be used to rank the objectives which are also defined as elements. The identification of the criteria is an iterative process where all the decision makers related to this problem need to discuss which are the factors that affect the definition of an objective as more important than another taking into consideration the organisational strategy, the current situation and characteristics related to the project itself and the customer related to it. Therefore the criteria are subject to change from project to project and time to time. However, here a generic set of criteria that can be widely used are provided. The selection of the criteria is based on interviews with experts of the project management field. The proposed set of criteria can be summarised in the following elements:

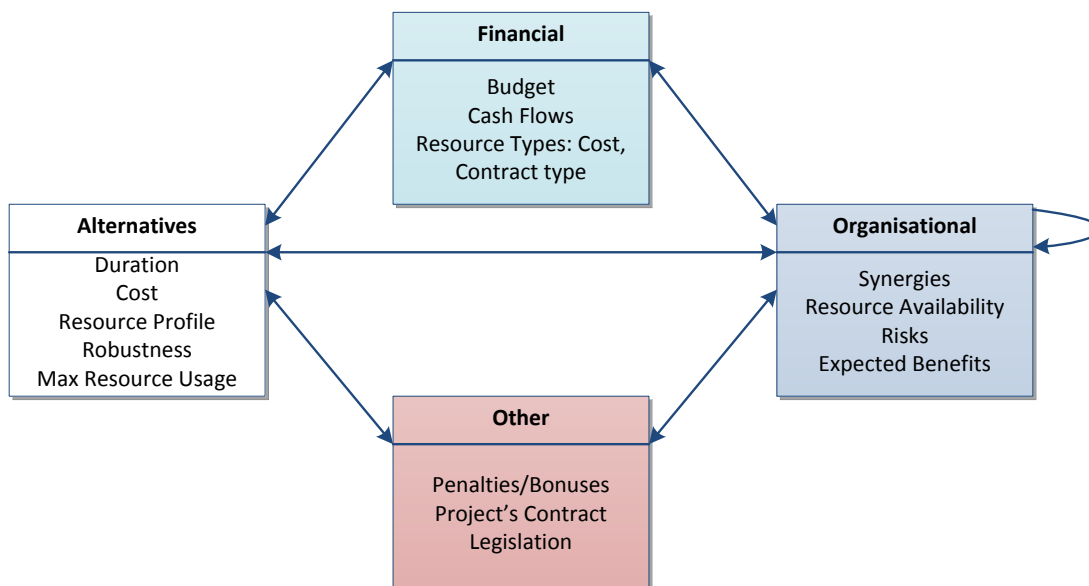
- **Budget:** the budget available for the project is a good indicator of how important will be the cost factor in the decisions to be taken, as a tight budget will let small margins for those makespan minimisations that require high level of resource availability and usage, or expensive resource types. On the other hand it complies with the minimisation of the maximum resource usage and the smoothness of resource profiles however, it is not strogly related to the objective of robustness.
- **Cash Flows:** timing of the inflows can affect the overall strategic decisions about the project, for example when the inflows are attached to specific milestones, then the deadlines, therefore the time objective, is critical for the whole project's viability. Similarly, the outflows affect the ranking of the objectives, as for example large payments connected to specific activities can lead to right shifting those activities and thus slowing down their execution.
- **Resource Types- Cost, Contact type:** the cost of the resource types to be used and the type of contractual relation that they have with the company that is executing the project affect a variety of decisions about the way that the project should be executed and how the objectives should be ranked. Permanent contractual relations, ask for very smooth resource profiles and enforce the need to be within the given resource requirements as the need for new resources will call for hiring of probably, temporary personnel, with unsure impact on the efficiency of the work team. Low daily rates and extended usage of external contractors lead to flexibility of the resource related

constraints and low ranking of the resource profile objective. The existence of different resource types and associated needs for their profile is handled using different weights for each resource type within the resource profile objective.

- **Penalties/Bonuses:** in case of penalties and/or bonuses related to the completion time of the project or specific milestones, the time objective is expected to be prioritised and the cost objective needs to be balanced in relation to the time related loss or gain of money.
  - **Project's Contract:** is related to specific clauses of the contract signed between the organization and the customer which might affect the ranking of the objectives. For example in case that there are clauses specifying quality levels of the generated project outcome/product then the robustness objective's value is elevated due to the need to have extra time available to anticipate the need to rework part of some activities to meet the required quality levels.
  - **Legislation:** related to work hours, hiring and firing of personell, subcontracting, defining wages, specialised personnel needed to execute specific activities, environmental issues, etc. affects the way that the project should be scheduled and thus the ranking of the objectives.
  - **Synergies:** existing projects, on going or already executed, can be combined to the under examination project in terms of budget, human resource pool, equipment, etc., affecting this way the priorities set for the specific project, especially, the aspects of cost and resource usage.
  - **Resource Availabilities:** refers to the effort needed to find extra resources if needed, the corresponding cost of using extra units of some resource type and when referring to work force, the expected efficiency of the new resources when combined to the existing ones. This data are always related to circumstantial factors as the specific time period, the social and political situation, etc. Having low resource availabilities makes the resource constraints less flexible and the possibility to use execution modes of activities requiring large numbers of resources very expensive, thus unlikely.
  - **Risk:** refer to scope risks, like ill defined scope, integration issues, scope creep, etc., schedule risks as delayed decisions, wrong estimations of duration and effort needs, ommision of dependencies, etc. and resource related risks as delays caused by outsourcing, lack of cash flow, low quality and/or attrition of resources, loss of work team balance by people joining the team late, scarcity of skills. The types and levels of risk are strongly related to all the objectives and based on the probability of appearance and the expected impact of each one, different objectives can be prioritised for the mitigation of one or more risks.
  - **Expected Benefits:** are related to factors like the project's expected outcomes and the customer's importance for the organization. The expected benefits can balance high costs and prioritize objectives like time and robustness.
- 2) Categorisation of the elements in suitable clusters, that is based on the similarity of the previously identified element's characteristics.
- The criteria and alternative solutions composing the model for ranking the project scheduling objectives, are grouped in three clusters: Organisational containing the criteria that are strongly related to the specific organisation, its strategy and profile, Financial, containing the cost and budget related factors and Other, containing the factors that could not fit in the previous two categories. Finally, there is also the Alternatives cluster containing the objectives to be ranked.

- 3) Definition of the influences: inner dependencies refer to influences (social, political, technical, economic, etc.) between two elements of the same cluster and outer dependencies between two different clusters.

In this model, the Alternatives cluster depends on all the other clusters and viceversa. The Financial cluster influences and is influenced by the Organisational, as the resource availability influences the rates, the budget is affected by the risk factors, the synergies and the expected benefits, and there are risks related to the flows and the budget. The Organisational cluster is influenced by the Others cluster and it has inner dependencies among its components, as the legislative rules and the contract affect the risk factors, for example having high penalties for exceeding the deadline or getting products below the expected quality level raise the impact of the corresponding risks. The penalties along with the specific project contract influence the expected benefits. As concerned to the inner influence for the Organisational cluster, the synergies and the risks influence the expected benefits, the more synergies that can be generated through the project and the less risk the higher are the expected benefits. The final model is shown in Figure 6.3.



**Fig. 6.3** ANP model

- 4) Forward and backward examination of the network by cluster, in order to make sure that the generated network structure is complete and consistent. The decision makers check that the network reflects the real problem to be solved, that the relationships among the elements have the correct direction and no important aspects of the problem have been omitted.

This process, for the generic model herein analysed, was effectuated by the same group of project management experts, as to ensure its completeness and consistency. The goal was to design a generic model that could fit in a variety of project scheduling cases and would give a way to express those qualitative factors that affect the way that the project should

be scheduled without omitting the quantitative data needed for the prioritisation of the objectives.

After the generation of the network describing the problem, the decision makers are asked to enter their judgments. The network is used as a starting point for the formulation of all the pairwise comparisons and their conversion to meaningful questions. These questions have the form: "Which element  $A$  or  $B$  is more important in the context of cluster  $C$ ?", where  $A$  and  $B$  are elements of the same cluster  $D$  and  $C$  is a cluster that influences the cluster  $D$  containing the elements  $A$  and  $B$ . This step leads to filling a matrix having as lines and columns the elements of cluster  $D$ , the so called comparison matrix. The same is done for the clusters themselves. A comparison matrix is generated for each relationship between two clusters. Then for each comparison matrix the consistency ratio is computed and in case that it exceeds the predefined limit, the judgments responsible for this excess are located and corresponding questions are redirected to the decision maker for reassessment until the desired level of consistency is reached. As measure of deviation from consistency, the introduced by Saaty 1996 consistency index ( $C.I.$ ) is used:

$$C.I. = \frac{\lambda_{max} - n}{n - 1} \quad (6.1)$$

where  $\lambda_{max}$  is the Perron eigenvalue of the positive reciprocal matrix being examined. The consistency ratio ( $C.R.$ ), of the pairwise comparison matrix is the ratio of its inconsistency index  $C.I.$  to the corresponding random index value,  $C.R. = \frac{C.I.}{R.I.}$ . Random index ( $R.I.$ ) values are computed using multiple simulations of randomly created comparison matrices and calculating the average of the consistency index (Saaty and Sagir, 2009). If the  $C.R.$  of a pairwise comparison matrix is larger than 10% then it is necessary to find which are the most inconsistent judgments in that matrix and ask the decision maker to consider changing his judgment to a value that will lead to an acceptable value of  $C.R.$  The most inconsistent judgment can be computed using the formula:

$$Max(a_{ij} * \frac{w_j}{w_i} \forall i, j \in 0, 1, \dots, n) \quad (6.2)$$

where  $a_{ij}$  is the  $(i, j)$  item of the comparison matrix and  $w_i, w_j$  are the corresponding weight values.

The next step consists in the computation of the weight vector that corresponds to each comparison matrix and the combination of the vectors to create the initial Cluster Matrix and Supermatrix. To generate the Cluster Matrix from the pairwise comparison of the clusters we calculate the limit priorities of the corresponding matrices. To do so, starting from the matrices that were formed from the pairwise cluster comparisons, we compute the weight vector, raise the matrix to  $n + 1$  where  $n$  gets values  $[1, 2, \dots]$  and then compare the new weight vector to the old one. When the old and new weight vectors are equal, we have reached the goal of computing the weights for the specific cluster. Beforehand, is known that in some point the weight vector will be stabilised because the initial weight vector was column stochastic. This operation is repeated for all the clusters contained in the model. The weight vectors are then put together to formulate the Cluster Matrix. The algorithm ( $O(n)$ ) for this process is shown in Algorithm 6.11.

**Algorithm 6.1:** Cluster Matrix Calculation

---

```

foreach cluster  $C$  do
  repeat
     $A[,] = \text{Get}$  (pairwise comparison results that have  $C$  as context cluster);
     $W[] = \text{RowSum}$  ( $A[,]$ );
     $\text{TotalSum} = \text{Sum}$  ( $A[,]$ );
     $W[] = W[]/\text{TotalSum}$ ;
     $W_{old} = W$ ;
     $A_{new} = A * A_{old}$ ;
     $\text{Calculate}$  ( $W_{new}$ );
    if  $W_{new} = W_{old}$  then
      |  $\text{Stop}()$ ;
    else
      |  $A_{old} = A_{new}$ ;
    end
  until  $W_{new} = W_{old}$ ;
end

```

---

When the weight vectors for all the clusters are computed, the Cluster Matrix can be formed by setting as Cluster Matrix column the weight vector that corresponds to the column's cluster. In case of Group Decision, on this step before computing the limit priorities all group members' pairwise comparisons per context cluster are combined, using the geometric mean and then the above process can be initiated.

The Supermatrix consists of the normalised pairwise comparisons on a node level. This matrix is used to represent the flow of influence from each element of the network on all other elements in the same network. It is composed of principal eigenvectors of all the model's elements. To compute the Supermatrix the same process used for computing Cluster Matrix is used. The only point that needs attention is the handling of blocks, a block, consists of the weight vectors of its child nodes. Elements that have zero value correspond to elements that have no influence on the element in question.

The Weighted Supermatrix is nothing more than a stochastic Supermatrix. Indeed, from the initial Supermatrix to get the Weighted Supermatrix, first the Supermatrix is transformed to column stochastic and then the Hadamard product of the updated Supermatrix with Cluster Matrix, is calculated. If needed the columns are again normalised to keep summing to 1. Attention should be given to columns that are from the beginning stochastic and thus should not take part in the transformations. Furthermore, in the case where an entire vector but not all vectors in that component are zero then the weighted column must be renormalized. Last issue are sink components that need not to be included in the above calculations and whose priorities will be used during the final synthesis of

the results. In Algorithm 6.12, a more formalised way of describing this process, is shown.

---

**Algorithm 6.2:** Weighted Supermatrix Calculation

---

```

foreach column of the Supermatrix S[,j] do
  | if SumOfCol (S[,j]) != 1 then
  | | foreach item a in Col(S[,j]) do
  | | | a = a* W[OwnerCluster];
  | | end
  | end
end
foreach column of the Supermatrix S[,j] do
  | if SumOfCol (S[,j]) != 1 then
  | | foreach item a in Col(S[,j]) do
  | | | a = a/SumOfCol (S[,j]);
  | | end
  | end
end

```

---

In the end, the Weighted Supermatrix is limited by raising it to a sufficiently large power until it converges into a stable limit matrix and the weights of criteria and alternatives are used to get the final priorities. Input is the stochastic matrix  $W$ , therefore,  $\lambda_{max} = 1$ , because the principal eigenvalue of a matrix lies between its largest and smallest column sums, and all columns of a column stochastic matrix sum to 1. The idea is again to compute the limit matrix by calculating powers of this matrix till the limit is reached, that is when  $W^{n+1} = W^n$ . At that point, all the columns of the matrix will be identical and priorities can be easily computed for the elements of each cluster.

This is not always a straightforward calculation of matrix powers. The computational steps differ based on the initial matrix's morphology. There are three different computational paths: the first one deals with irreducible and primitive matrices or reducible with no other unitary roots besides the simple  $\lambda_{max} = 1$ , root, the second deals with cyclic matrices and the third one with hierarchies.

A matrix is reducible if it can be placed into block upper-triangular form by simultaneous row/column permutations. Thus, a matrix is reducible (Muoneke, 1987; Mesnard and Dietzenbacher, 1995) when its associated digraph is not strongly connected. An easy way to control if a square matrix is irreducible is based on the Perron–Frobenius theory of nonnegative matrices where it is proved that a square matrix is irreducible, if and only if for each  $i$  and  $j$ , there exists some  $k$  such that  $(I + W)^{n-1} > 0$ . The corresponding model cannot have source or sink nodes. Knowing that the matrix in question is irreducible, the next step is to define if it is primitive or cyclic. A sufficient condition for a matrix to be primitive is to be a nonnegative, irreducible matrix with a positive element on the main diagonal. In that case the limit matrix calculation is given by raising the Weighted Supermatrix to large powers. On the other hand, a square matrix  $A$  such that the matrix power  $A^{k+n} = A^n$  for a positive integer  $k$  is called a cyclic matrix (Tan, 2013). If  $k$  is the least such integer, then the matrix is said to have period  $k$ . If the matrix is reducible and cyclic then the result is calculated by averaging all matrices belonging to a cycle and normalising the results by blocks. In the other case, when the matrix in question is reducible we have to determine if  $\lambda_{max} = 1$ , is simple or multiple root and if there are other roots of unity or not. If there are other unitary roots then it is a cyclic matrix and limit can be computed in the same way used for irreducible cyclic matrix. If  $\lambda_{max} = 1$ , is a simple root and the matrix is reducible the same computational steps with those used for irreducible primitive matrix will give the desired result. If  $\lambda_{max} = 1$ , is a multi-



ple root and the matrix is reducible then we are talking about hierarchies and the limit matrix can be computed as the average of all powers of the matrix till the point that  $W^k = 0$ ,  $k \leq n$ . The idea behind the above calculations is simple, we initially have a graph, the ANP network, which describes the one step transitions among the nodes and we need to calculate the overall influence of all the transitions from any node of the graph to any other connected node, no matter the path length. Each transition length is represented by the corresponding power of the Weighted Supermatrix and the goal is to find that power of the Weighted Supermatrix for which all columns are identical and next powers don't add detail to the result. The fact that the initial matrix is column stochastic guarantees the existence of such steady state. In the proposed algorithm instead of having the conditions and then the path selection a unified process ( $O(n^k)$ ) is proposed to avoid repetition of steps, as shown in Algorithm 6.13.

Summarising, the proposed multi-objective approach requires from the decision maker to select the desired optimisation objectives and the criteria and relationships among them, to be able to prioritise the objectives. The ANP is used to generate the weight vector that reflects the preferences of the decision makers. This weight vector, gives an a priori knowledge of the decision makers preferences about the optimisation objectives. However, it is not a mandatory step, as the multi-objective optimisation process can be executed either using weights generated through any other external method or the unweighted vector of the objectives.

**Algorithm 6.3:** Limit Matrix Calculation

---

```

A[,] = Weighted Supermatrix;
Ainit[,] = Weighted Supermatrix;
W[] = SumOfLines (A);
TotalSum = Sum (A[,]);
W[] = W[]/TotalSum;
// Calculate the first two powers of A
A[,] = A[,]*Ainit[,];
powerOfA++;
Add MemoryOfMatrices(A,1);
countOfMem++;
Add MemoryOfMatrices(A2,2);
countOfMem++;
repeat
  // Check if A is cyclic
  foreach matrix A'[,] in MemoryOfMatrices do
    if A = A' then
      k = IndexOf(A');
      return (W(Average (A1,...Anodes-k)));
    else
      countOfMem++;
      MemoryOfMatrices = Add (A,powerOfA);
    end
  end
  if countOfMem = Count (MemoryOfMatrices) then
    Wold = W;
    Anew = Ainit * Aold;
    powerOfA++; Calculate Wnew;
    // Simple ANP model
    if Wnew = Wold then Stop;
    ;
    // Hierarchy
    else if Wnew = 0 then return (Average (W1,...,Wcur));
    ;
    // Repeat the loop
    else
      Aold = Anew;
      A = Anew;
    end
  end
until Wnew = Wold;

```

---

## 6.3 Proposed Solution Algorithm

### 6.3.1 Basic Scheme

The proposed solution algorithm is a flexible way of solving a variety of RCPSP problems. It can be used both for the multi-objective and the single objective versions of project scheduling. It is also adaptable to any combination of the existing RCPSP variations, therefore, it can effectively and efficiently solve the simple RCPSP, the multi mode RCPSP, with or without generalised precedence constraints, having or not variable renewable and/or non renewable resource demands and requirements.

The backbone, of the whole process, is a Genetic Algorithm (GA) that acts as a moderator of the solution process. The proposed process essentially lets the GA decide which algorithm and decoding procedure is promising and work with it. This is achieved using additional genes to enable the evolutionary process to decide which combination of solution algorithm and decoding process is the best for the under examination problem. In other words, the evolution is used not only to find a good solution for the problem but also a good algorithm to solve the problem. The genetic algorithm adapts itself to the problem instance actually solved. This way not only the list of activities to be scheduled but also the algorithm itself and the decoding algorithm are subject to genetic optimisation.

Initially, the input data are analysed and transformed to a predefined form for ease of usage and preprocessing is done to eliminate redundant data, like resource types that are abundant and cannot affect the scheduling process, and ineffective or non executable modes. Then an initial solution set is randomly generated and crossover and mutation operators are used to generate the offspring population. At this point, the process is diversified based on the problem type being solved: a) single objective, b) multi-objective with a priori preferences (from the ANP) or c) multi-objective with a posteriori preferences (on the result set), as shown in Figure 6.4.

Individuals (chromosomes) are composed of four parts: the *ALGOgene* representing the solution algorithms that compete for survival, which are the best in class algorithms for project scheduling (Kolisch and Hartmann, 2006): tabu search (TS), simulated annealing (SA), particle swarm optimisation (PSO) and genetic algorithm(GA), these algorithms herein will be called auxiliary algorithms, the *SGSgene* denoting the usage of either the serial or the parallel adapted SGS, to generate the schedule corresponding to the chromosome, the *ActivityList* that is a permutation of the activities to be scheduled and the *ModeList* representing the selected execution modes for each activity. The last pair of lists represents the initial solution or, when grouped, the initial population that is fed to a solution algorithm defined by the *ALGOgene* of the chromosome.

The proposed adaptive GA described in Algorithm 6.14, in order to generate the offspring uses a two point crossover operator for the main body of the chromosome and a swap operator for the auxiliary algorithm and the SGS genes. It also mutates the two genes and the activity and mode lists with user defined probabilities ( $p_{mut_{ALGO}}, p_{mut_{SGS}}, p_{mut}$ ).

Therefore, the moderator GA is responsible for mutation, crossover and selection of the next generation and the auxiliary algorithms are responsible for the search of a sub-space of the solution space, based on the individuals that were given as input, the update of the given individuals in relation to the results found and the calculation of the fitness. As a result, the auxiliary algorithms effectuate a parallel, local or global search, depending on the algorithm, in the solution space of a chromosome or a group of chromosomes belonging to the current generation and exchange the given chromosomes with better ones, having better fitness value, as shown in Figure 6.5.

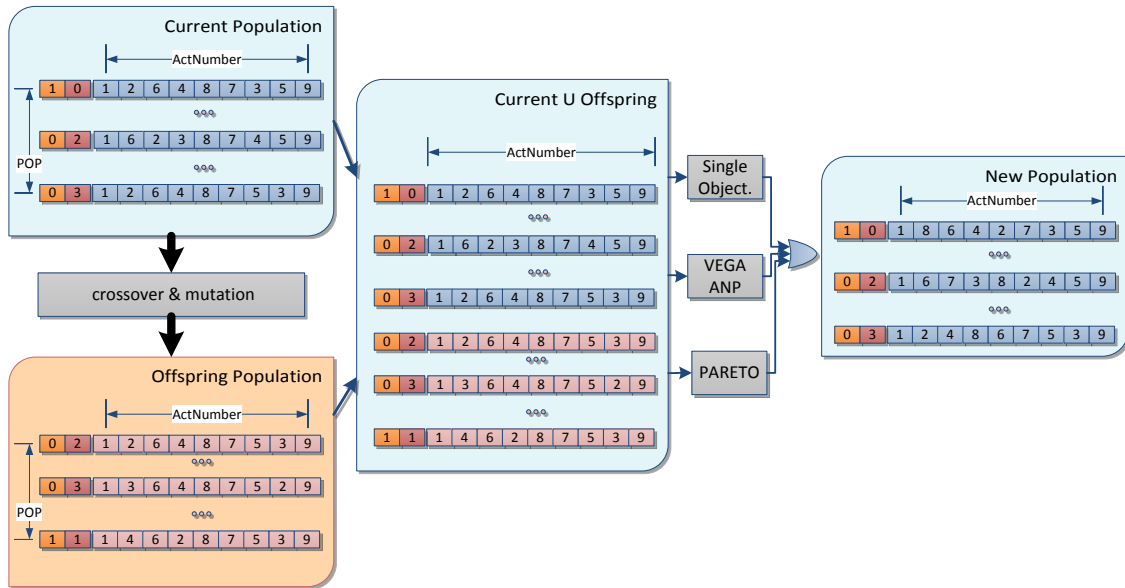


Fig. 6.4 Solution process moderator basic flow of events

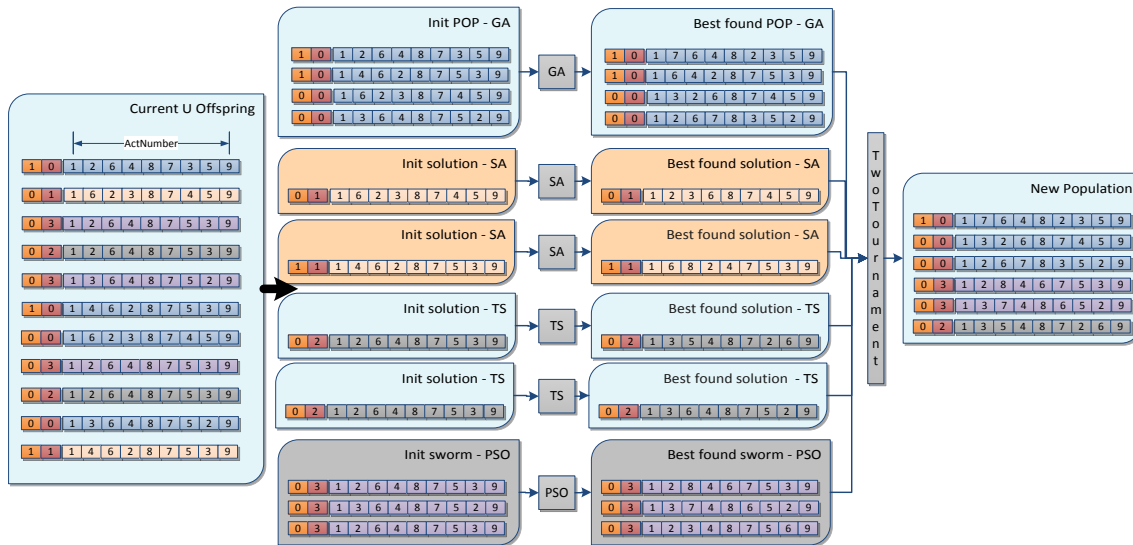


Fig. 6.5 Usage of auxiliary algorithms

The selection strategy and the fitness calculation is differentiated based on the specific problem type. In the case of single objective, fitness is calculated straightforward using the corresponding objective function on each decoded chromosome. If there are multiple objectives and a priori preferences, then the population is split in sub-populations, each one corresponding to one objective and the fitness is computed using the corresponding objective per population, then the sorted chromosomes of both the initial and the offspring sub-populations

**Algorithm 6.4: Moderator GA**


---

```

set populationSize= POP;
set problemType= UInput;
set crossoverType, probMutationALGO, probMutationSGS, probMutation;
set generation counter  $g = 0$ ;
set AlgoNum= 4;
set emptySlots= POP;
set ObjNum= UInput;
// generate initial population  $P_0$ 
for  $i = 0 \dots POP$  do
    |  $P_0 = P_0 \cup \text{GenerateRandomActList}$ ;
    |  $P_0[i].Algo = \text{imodAlgoNum}$ ;
end
for  $i = 0, \dots, POP/2$  do
    |  $P_0[i].SGS = \text{serialSGS}$ ;
    |  $P_0[i + POP/2].SGS = \text{parallelSGS}$ ;
end
 $P_g = P_0$ ;
while stopping criteria not met do
    | // generate offspring population  $P_{gchildren}$ 
    |  $P_{gchildren} = \text{Crossover } P_g$ ;
    |  $P_{gchildren} = \text{Mutate } P_{gchildren}$ ;
    | // combine current and offspring population
    |  $R_g = P_g \cup P_{gchildren}$ ;
    | // choose selection strategy based on input
    | switch problemType do
    | | case singleObjective
    | | | for  $i = 0 \dots POP$  do
    | | | | if  $R_g[i].Algo = SA$  then  $SA(R_g[i], \text{out Fit}(R_g[i]))$ ;
    | | | | ;
    | | | | if  $R_g[i].Algo = TS$  then  $TS(R_g[i], \text{out Fit}(R_g[i]))$ ;
    | | | | ;
    | | | | forall the  $R_g[i].Algo = GA$  do
    | | | | |  $GA(R_g[i], \text{out Fit}(R_g[i]))$ 
    | | | | end
    | | | | forall the  $R_g[i].Algo = PSO$  do
    | | | | |  $PSO(R_g[i], \text{out Fit}(R_g[i]))$ 
    | | | | end
    | | | end
    | | |  $P_g = \text{TwoTournamentSelection}(R_g)$ ;
    | | end
    | | case ANPmultiObjective  $\text{ANPMOGA}()$ ;
    | | ;
    | | case ParetoOptimality  $\text{ParetoGA}()$ ;
    | | ;
    | | endsw
    | |  $g = g + 1$ 
end

```

---

are used to fill the next generation in proportion to the given weights. In the third case, the Pareto optimality concept is explicitly utilised and each chromosome is assigned a fitness value based on its rank in the population and not on its actual objective function value. For those solutions that belong to the same Pareto front, if sorting is needed, then the Chebycheff metric, is used.

### 6.3.2 Preprocessing

In the herein modelled problem, generalised precedence constraints, preemptive activities, variable resource requirements and availabilities co-exist in the same instance and therefore a complex situation emerges when trying to generate a feasible schedule. To handle the complexity a preprocessing procedure is applied over the project data to lower the complexity of the process, by transforming the inputs in a more manageable form, prepare for the solution process and limit the search space.

Preemption is internally handled by generating sub-activities. If the user has defined specific split points, for example, as percentage of completion of the task, then these points are used to split the task and for each split point finish to start relations with zero lag are added to the list of constraints. In the second case, that splits can be randomly made over the duration of the activity, unitary splits are set by generating subtasks with one time unit of duration and connecting each part with the next one with start to finish relations with zero lag. After these transformations the problem can be handled as non preemptable, as preemption has been handled through splitting of the related tasks and the generation of additional finish to start relationships between the consecutive parts.

The given set of non renewable resources may contain resources that their availability is greater than the maximum demand, called redundant. These resource types do not constraint the scheduling of the activities as they are abundant, therefore cannot affect the scheduling process and can be omitted during the computational phase. A non renewable resource type is defined as redundant when the sum of the maximum usage of this resource type for all the activities, that is given by taking the mode that has the maximum resource usage, is less or equal to the available amount, as shown in Algorithm 6.15. However, resource requirements and availabilities are not constant over time, thus each time period should be examined separately and if the resource type is redundant in all the time periods then it can be eliminated.

---

#### Algorithm 6.5: Redundant Modes

---

```

input : Activities, ResourceRequirements, ResourceAvailabilities,
        NonRenewableResources, Modes
output: RedundantNonRenewableResources
forall the  $i \in \text{Activities}$  do
    forall the  $m \in \text{Modes}[i]$  do
        forall the  $k \in \text{NonRenewableResources}$  do
             $\text{sum}[k, m] = \text{sum}[k, m] + \text{ResourceRequirements}[i, k, m];$ 
        end
    end
end
forall the  $m \in \text{Modes}[i]$  do
    forall the  $k \in \text{NonRenewableResources}$  do
        if  $\text{sum}[k, m] > \text{ResourceAvailabilities}[k, m]$  then add (k, redundantRes);
        ;
    end
end
return (redundantRes);

```

---

Given modes, can be non executable, however, we do not imply that the project manager has entered a mode that cannot be executed, which is an easy to handle mistake, but that the requirements set by a mode when combined to the rest of the activities and given specific resource availabilities cannot be satisfied by any combination of modes of all the other

activities. A mode is defined as non executable when its resource demand for one or more resource types, when added to the minimal resource demands (per type) of all the other project activities, exceeds the availability for at least one time period, as shown in Algorithm 6.16.

---

**Algorithm 6.6:** Non executable Modes

---

```

input : Activities, ResourceRequirements, ResourceAvailabilities,
         NonRenewableResources, NonRenewableResources, Modes
output: NonExecutableModes
forall the  $k \in \text{NonRenewableResources}$  do
    set minResUsage[k]=minResUsage (k,Modes,ResourceRequirements);
    forall the  $i \in \text{Activities}$  do
        forall the  $m \in \text{Modes}[i]$  do
            if  $\text{ResourceRequirements}[i,k,m] + \text{minResUsage}[k] > \text{ResourceAvailabilities}[k]$ 
            then
                | add ( $\text{Modes}[i]$ ,nonExecutableModes);
            end
        end
    end
end
forall the  $k \in \text{RenewableResources}$  do
    set minResUsage[k]=minResUsage (k,Modes,ResourceRequirements);
    forall the  $i \in \text{Activities}$  do
        forall the  $m \in \text{Modes}[i]$  do
            if  $\text{ResourceRequirements}[i,k,m] > \text{ResourceAvailabilities}[k]$  then
                | add ( $\text{Modes}[i]$ ,nonExecutableModes);
            end
        end
    end
end
return (nonExecutableModes);

```

---

A mode beside not being possible to be executed can also be inefficient. A mode is called inefficient, when duration, renewable and non renewable demands are higher than those of all the other modes of the same activity, as shown in Algorithm 6.17. All inefficient modes should be omitted from the scheduling process as they cannot lead to a good solution.

**Algorithm 6.7:** Inefficient Modes

---

```

input : Activities, ResourceRequirements, ResourceAvailabilities,Resources
output: Inefficient Modes
forall the  $i \in Activities$  do
    forall the  $m_i \in Modes[i], m_j \in Modes[i], m_i \neq m_j$  do
        if  $d[m_i] \geq d[m_j]$  then
            forall the  $k \in Resources$  do
                if  $ResourceRequirements[i, k, m_i] \leq ResourceRequirements[i, k, m_j]$  then
                     $count++$ ;
                ;
            end
            if  $count = ResourceRequirements.length$  then  $add(m_i, i, inefficientModes)$ ;
            ;
        end
    end
end
return ( $inefficientModes$ );

```

---

**6.3.3 Proposed Schedule Generation Schemes for the extended RCPSP**

The majority of heuristics for project scheduling problems are based on the schedule generation scheme (SGS) that is an algorithm which schedules one activity in each step until a complete schedule is constructed. In this process, the set of activities that are eligible for scheduling are calculated for each step and a start time for the selected activity is computed in such a way that all resource and precedence constraints are fulfilled. The activity selection itself is not part of the SGS and it can be done for example by a priority rule or a genetic representation.

Two types of schedule generation schemes are available for the standard RCPSP, namely the parallel SGS and the serial SGS (Kolisch and Hartmann, 1999). The parallel SGS is based on activity incrementation, operates on the set of non-delay schedules and its search space might not contain the optimal solution. In each step, it schedules all those activities whose predecessors have already been scheduled and can be resource feasibly scheduled at that time instant. The serial SGS is based on activity incrementation, constructs active schedules, where no activity can be left shifted without delaying some other activity and its search space will always contain the optimal solution (Sprecher, 1994). In each step, it selects an eligible activity, whose predecessors have already been scheduled, and schedules it at the earliest resource feasible time.

The behaviour of both serial and parallel SGS can be differentiated when the standard RCPSP's assumptions about preemption, execution mode, activities precedence and resource types, requirements and availability, do not hold. Below, the most common schedule generation approaches for each case are summarised:

- a) splittable activities: when the split points are known in advance an easy way to handle splittable activities is by creating the corresponding sub-tasks and adding finish to start relationships between each couple of parts. In case of random splits each random set of split points would lead to a slightly different activity network and based on the optimisation



- objectives the most fitted would be selected through an iterative process. In any case, the standard series/parallel SGS can be used in the generated network with no other changes.
- b) multiple execution modes: usually the mode selection process is handled separately. After fixing the activities modes the standard SGS can be used.
  - c) generalised temporal constraints: in this case even the process of finding a feasible schedule is NP hard. Having to handle generalised precedence constraints elevates the complexity of usually simple calculations as the ES and LS of the activities composing the project. The existence of both maximal and minimal time lags leads in networks with cycles, where at least one arc in each cycle corresponds to a maximal time lag. Cycles of positive length do not make any sense as they correspond to constraints like:  $s_i \geq s_i + l$ ,  $l > 0$ . Cycles, having all of their arcs zero weighted, correspond to a set of activities that should be started at the same time. Early and Late Start of activities are calculated using the Floyd Warshall algorithm (Neumann and Morlock, 1993), as shown in Algorithm 6.18. The goal is to find the longest path  $l_{ij}$  between all pair of activities  $(i, j)$  with  $i, j \in V$ . In this algorithm, we start from the given generalised constraints and we convert all of them to start-to-start precedence constraints with lags, using the formulas introduced in section 5.5.1. The resulting data are used to form the initial distance matrix, as shown in equation 6.3.

$$distMatrix_{i,j} = \begin{cases} l_{ij} = lag_{ij}, & \forall (i, j) \in G \\ 0, & i = j \\ -\infty, & \text{otherwise} \end{cases} \quad (6.3)$$

The initial distance matrix shows the direct paths from node  $i$  to node  $j$ . Following an iterative process takes place, where at each iteration  $k + 1$ , the longest path from  $i$  to  $j$  such that any intermediate vertices on the path are chosen from the set  $\{1, 2, 3, \dots, k\}$ , is calculated. There are two possibilities either  $k$  is not a vertex on the path and the longest path has length  $l_{ij}^{(k)}$  or  $k$  is a vertex on the path and the longest path has length  $l_{ik}^{(k)} + l_{kj}^{(k)}$ . The last iteration,  $k = (|V| + 1)$  will give the longest path between each pair of activities  $(i, j)$ . In the final distance matrix, the first line, which corresponds to the dummy start activity, gives the ES of all the activities in the network. Similarly, the LS of the activities is calculated as  $LS_n - LS_i$ , where  $LS_n$  is the latest start time of the dummy end activity and can be equal either to  $ES_n$  or an upper bound  $T$  of project total duration and  $LS_i$  is the length of the longest path from node  $i$  to node  $n$ , as it is given by the last column of the final distance matrix, as shown in the example of Figure 6.6.

A precedence graph can be generated by creating edges for all  $l_{ij} > 0$  or  $l_{ij} = 0$  and  $l_{ji} < 0$  and eliminating redundant arcs (Neumann et al., 2002; Neumann and Zimmermann, 2002; Neumann and Schwindt, 2002).

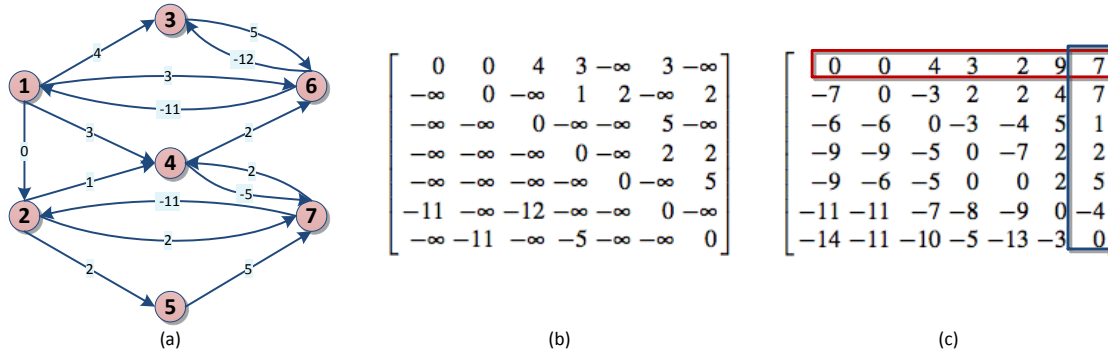


Fig. 6.6 (a) Example of graph with generalised precedence constraints, (b) initial distance matrix and (c) final distance matrix

---

### Algorithm 6.8: Floyd Warshall Calculations

---

```

input : Activities, actNum, GeneralisedPrecedenceRelations, Durations
output: ES[], LS[]
distMatrix[]=Calculate (GeneralisedPrecedenceRelations, dist[i, j]);
for k=1 to actNum do
    for i=1 to actNum do
        for j=1 to actNum do
            distMatrix[i, j] =
                max(distMatrixOld[i, j], distMatrixOld[i, k] + distMatrixOld[k, j]);
        end
    end
    distMatrixOld = distMatrix;
end
// set ES as the 1st line's values
for i=1 to actNum do ES[i]=distMatrix[0,i];
;
// set LS using the nth column's values
LS[actNum]=ES[actNum];
for i=1 to actNum do LS[i]=LS[actNum]-distMatrix[i, actNum];
;

```

---

The corresponding SGS is an extension of the standard SGS where an unscheduling step is added to handle the case when in some step the selected activity cannot be scheduled without breaking either some precedence or resource constraint.

- d) time dependent resource demands and availabilities: although the standard SGS might omit some active schedules and therefore the optimal solution, it has been used with very good results with no modifications (Hartmann, 2013)

In order to accommodate the requirements slight variations of serial and parallel Schedule Generation Scheme with unscheduling steps (Neumann et al., 2003) are used, as shown in Algorithms 6.19-6.21.

In the serial SGS with unscheduling step (Algorithm 6.19), the inputs are: a specific activity list, a fixed mode list, both defined by the chromosome, along with the precedence constraints and the resource requirements for each activity and resource type. It is an iterative process where at each step from all the activities that are not already scheduled, it is selected one,

using the priority given by the activity list and it is scheduled at the earliest resource feasible time. If the earliest resource feasible time is later than the latest start time of the selected activity then we perform an unscheduling step, otherwise the activity is scheduled and the ES and LS of the unscheduled activities are updated accordingly. To avoid infinite repetitions a maximum number of unscheduling steps is defined.

---

**Algorithm 6.9:** serial SGS with unscheduling
 

---

**input** : Activities, GeneralisedPrecedenceRelations, ResourceRequirements,  
ResourceAvailabilities, ModeList

**output:** FeasibleSchedule

set  $V = \{Activities\}$ ;

set  $C = \{i \in Activities \mid i \text{ completed}\}$ ;

set  $E = \{i \in Activities \mid i \text{ eligible}\}$ ;

set  $S_i = \text{start time of activity } i$ ;

set  $A = \{i \in C \mid S_i \leq t < S_i + d_i\}$ ;

set  $d_{i^*i} = \text{longest path from } i^* \text{ to } i$ ;

$S_0 = 0$ ;

$C = \{0\}$ ;

$u = 0$ ;

**while**  $C \neq V$  **do**

Select  $(j^* \in E)$ ;

$t^* = \min\{t \geq \{ES_{i^*}^c \mid r_k(S^c, T) + r_{ik} \leq R_k\}, \forall t \leq \tau < t + d_i, \forall k \in R\}$ ;

**if**  $t^* > LS_{j^*}$  **then**

$u = u + 1$ ;

Unschedule  $(j^*, t^* - LS_{j^*})$ ;

**else**

// schedule  $j^*$  at time  $t^*$

$S_{j^*} = t^*$ ;

$C = C \cup \{j^*\}$ ;

// update  $ES_j, LS_j$

**forall the**  $j \in V - C$  **do**

$ES_{j^*}^c = \max(ES_j^c, S_{j^*} + d_{j^*j})$ ;

$LS_{j^*}^c = \min(LS_j^c, S_{j^*} - d_{j^*j})$ ;

**end**

**end**

**end**

$S = S^c$ ;

**return** Schedule S;

---

The unscheduling step (Algorithm 6.20) consists in unscheduling all the already scheduled activities that are related to the under consideration activity  $j^*$  and affect the value of  $LS_{j^*}^c$  and right shift these activities for an amount of time equal to the difference of  $LS_{j^*}^c$  from the tested time instance  $t^*$ . If there are no activities to be unscheduled then no feasible schedule can be found and the algorithm exits otherwise, all the activities that due to the right shift could be now scheduled earlier,  $S_i > \min_{h \in U} S_h$ , are also unscheduled. Finally, the ES and LS of all the activities are re-calculated.

**Algorithm 6.10:** Unsheduling

---

```

input :  $j^*, \Delta$ 
output: Schedule, ES, LS
set  $U = \{i \in C \mid LS_{j^*} = S_i - d_{j^*i}\}$ ;
// no fs schedule is found
if  $0 \in U \parallel u > u_{max}$  then exit (0);
;
// right shift of activities  $i \in U$ 
forall the  $i \in U$  do
     $ES_i = S_i + \Delta$ ;
     $C = C - \{i\}$ ;
    // no fs schedule is found
    if  $ES_i > -d_{i0}$  then exit (0);
;
end
// unschedule all activities  $i$  with  $S_i > \min_{h \in U} S_h$ 
forall the  $j \in V - \{C\}$  do
     $ES_j = \max[d_{0j}, \max_{h \in U} d_{hj}]$ ;
     $LS_j = -d_{j0}$ ;
    forall the  $i \in C$  do
         $ES_j = \max(ES_j, S_i + d_{ij})$ ;
         $LS_j = \min(LS_j, S_i - d_{ji})$ ;
    end
end

```

---

The parallel SGS (Algorithm 6.21) uses the same input data as the serial SGS and again it generates a feasible schedule if there is one for the given data. However, this time the iterations are time based and during each iteration all activities that can be resource and time feasibly scheduled will be scheduled instead of having one activity being scheduled per iteration as it was the case in the serial SGS.

First, the eligible activities are defined as those whose all their "predecessors" have already been completed. A tentative time  $t^+$  is calculated as the minimum early start of the set of eligible activities. For all the eligible activities, the activity with the highest priority is selected based on the given selection rule and a time,  $t^*$ , that it can be resource feasibly scheduled is computed. If  $t^*$  is greater than the LS of the selected activity, an unschedule step takes place otherwise  $t^*$  is compared to the tentative time  $t^+$ , if it greater then the ES of all the activities are updated otherwise, the activity is scheduled and ES, LS are updated accordingly.

**Algorithm 6.11:** parallel SGS with unscheduling

---

```

input : Activities, GeneralisedPrecedenceRelations, ResourceRequirements,
         ResourceAvailabilities, ModeList
output: FeasibleSchedule
set  $V = \{Activities\}$ ;
set  $C = \{i \in Activities | i \text{ completed}\}$ ;
set  $S_i = \text{start time of activity } i$ ;
 $S_0 = 0$ ;
 $C = \{0\}$ ;
 $u = 0$ ;
while  $C \neq V$  do
  set  $G = \{i \in V - \{C\} | GenPred(i) \subset C\}$ ;
  set  $t^+ = \min_{i \in G} ES_i$ ;
  set  $E = \{i \in G | ES_i = t^+\}$ ;
  while  $E \neq \{\emptyset\}$  do
    // try to schedule activities at  $t^* \leq t^+$ 
     $j^* = \text{Select}\{j \in E\}$ ;
     $t^* = \min\{t \geq ES_{j^*} | r_k(S^c, \tau) + r_{j^*k} \leq R_k, t \leq \tau \leq t + d_j, k \in R\}$ ;
    if  $t^* > LS_{j^*}$  then
      Unschedule( $j^*, t^* - LS_{j^*}$ );
       $E = \{\emptyset\}$ ;
    else
      if  $t^* > t^+$  then
        forall the  $j \in V - \{C\}$  do
           $ES_j = \max(ES_j, t^* + d_{j^*j})$ 
        end
      else
        // schedule  $j^*$  at time  $t^*$ 
         $S_{j^*} = t^*$ ;
         $C = C \cup \{j^*\}$ ;
        forall the  $j \in V - \{C\}$  do
          // update ES, LS of all the activities
           $ES_j = \max(ES_j, S_{j^*} + d_{j^*j})$ ;
           $LS_j = \min(LS_j, S_{j^*} - d_{jj^*})$ 
        end
      end
    end
  end
end
 $S = S^c$ ;
return Schedule S ;

```

---

**6.3.4 Chromosomes**

The moderator GA is used to adapt the solution method and schedule generation scheme to the specific instance being solved. This is achieved by adding two new genes to the activity

list representation. The one is used for controlling the solution algorithm (*AlgoGene*) and the other for the schedule generation scheme selection (*SGSGene*), as shown in Figure 6.7. Furthermore, having multiple modes of execution per activity, an addition list containing the corresponding modes (*ModeList*), is needed.

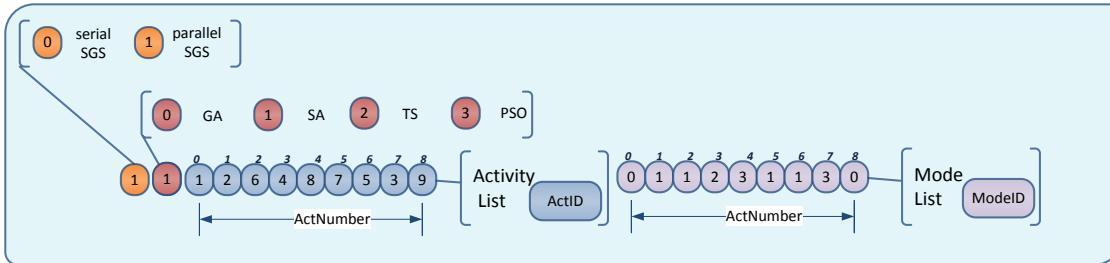


Fig. 6.7 GA chromosome

Each individual (chromosome), is a composite class formed by:

- the *SGSGene*, defining which decoding procedure should be used,
- the *AlgoGene*, defining the solution algorithm that should be applied to the chromosome,
- the *ActivityList*, consisting of the ID's of the project activities, where *ActNumber* is the total number of activities including the dummy start and end and
- the *ModeList* containing the selected execution mode for each activity.

To obtain the schedule corresponding to a chromosome, serial or parallel schedule generation scheme (SGS) with unscheduling is used, as described in section 6.3.3. The selection of the SGS algorithm is based on the value of the *SGSGene*, 0 for serial and 1 for parallel SGS. In these SGSs, instead of using priority rules to select which eligible activity should be scheduled next, the activities are taken in the order given by the *ActivityList*. Attention should be given to the fact that the SGS is used after applying the solution algorithm defined in the *AlgoGene* and not straightforward as it usually happens.

The *ActivityList* is a precedence feasible permutation of the activities, meaning that each activity is positioned after all its immediate predecessors. In the proposed process, the *ActivityList* is created by setting the dummy start and end to the first and last position of the vector and then randomly choosing the remaining activities. Afterwards, a time feasibility check is used to purge those permutations that do not satisfy the given generalised precedence constraints.

The *ModeList* defines for each activity which execution mode will be used, therefore it defines the duration and variable renewable and non renewable resource requirements of the corresponding activity. The *ModeList* is a vector of modes. The position of a mode in the list represents the activity ID to which it is related. Each mode in *ModeList* identifies the execution mode of the activity placed in the corresponding position of the *ActivityList*.

### 6.3.5 Initial Population

In the initial population, the special genes (*AlgoGene* and *SGSGene*) are given equal number of chromosomes, for example if the population size (*POP*) is 100 and we use 4 possible solution algorithms (*GA, TS, SA, PSO*) then we will have 25 chromosomes having

$AlgoGene = GA$  with half of them having  $SGSGene = 0$  and the rest  $SGSGene = 1$ , as shown in Figure 6.8.

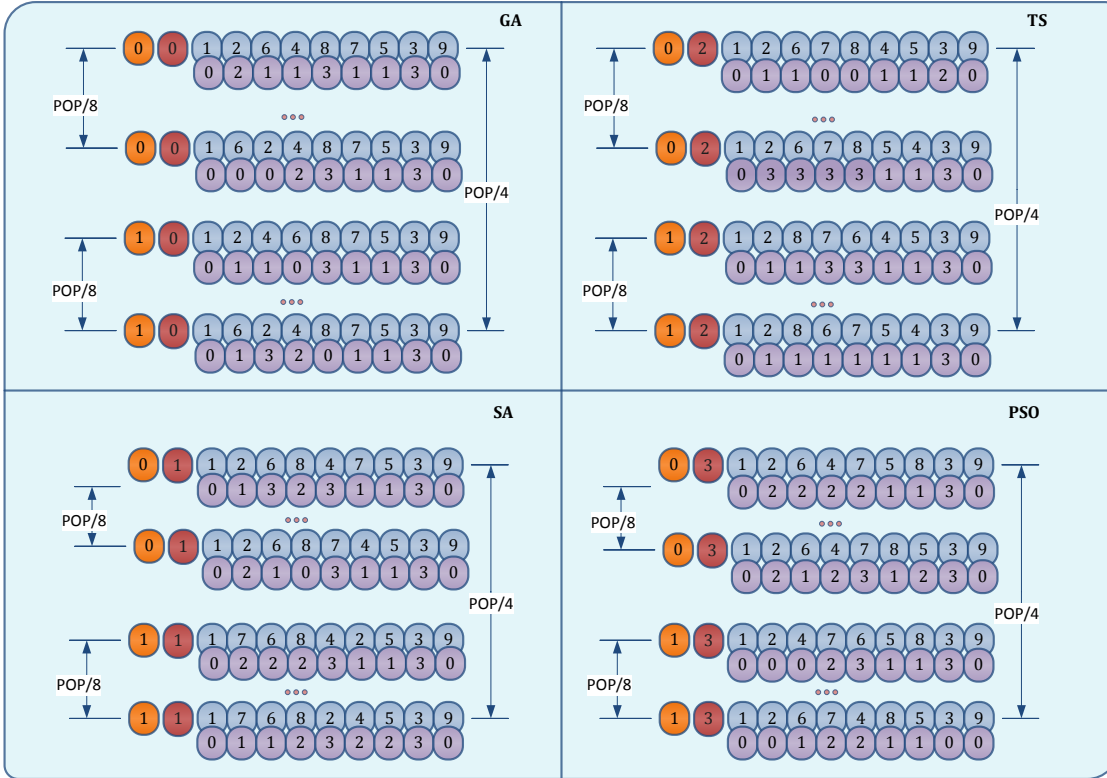


Fig. 6.8 Example of initial population

Activity and mode lists are randomly generated. In the case of activity list, positions 0 and ActNum-1 are set to activity id numbers 1 and ActNumber and the rest are randomly generated unique numbers in the span from 2 to ActNumber-1. Each generated list is checked against the generalised precedence constraints and if successful, the list is an *ActivityList* and it is used to form a chromosome.

The *ModeList* is formed similarly. For each activity, a mode is selected randomly in the span  $[0, modesNum[ActID] - 1]$  and all of them form the list of modes. No constraints need to be satisfied by the *ModeList* although a mode improvement process can be used to improve the selected list.

### 6.3.6 Operators

#### 6.3.6.1 Crossover

Based on the experimental results presented in Hartmann (1999), the two-point crossover that is an extension of the one-point crossover, was selected for both the *ActivityList* and

the *ModeList*. We randomly choose two chromosomes from the current population, let  $X$  (mother) and  $Y$  (father) be the parent chromosomes, then we randomly draw two integers,  $q_1$  and  $q_2$  with  $1 \leq q_1 < q_2 \leq ActNumber$  and we form two new chromosomes,  $XY$  (daughter) and  $YX$  (son), as shown in Figure 6.9.

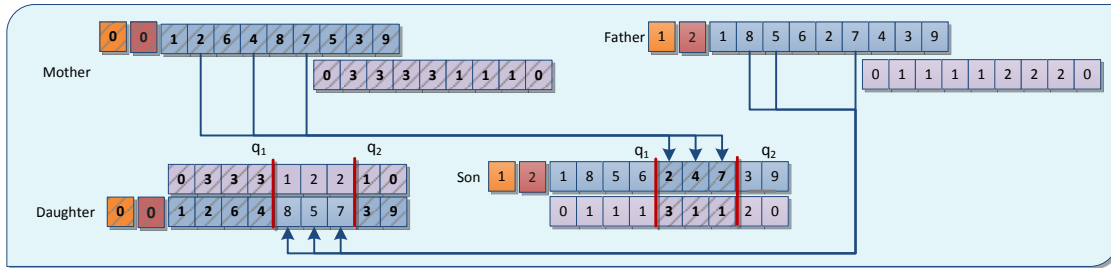


Fig. 6.9 Crossover operator

The  $XY$  chromosome will get from chromosome  $X$  the activities for the positions  $1, \dots, q_1$ , from  $Y$  the activities  $q_1 + 1, \dots, q_2$  and again from  $X$  the remaining  $q_2 + 1, \dots, ActNumber$ . Note that for the second and third part of the chromosome each time we take the lowest index from  $X(Y)$  that is not already included in the list. The  $YX$  chromosome is formed analogously. This operator has been proven that in case of FS precedence constraints always leads to precedence feasible lists (Hartmann, 1999). The additional genes (*AlgoGene* and *SGSGene*) are inherited from  $X$  chromosome for the  $XY$  child and from  $Y$  for the  $YX$  child.

### 6.3.6.2 Mutation

Mutation operators are used to diversify the population in ways that the crossover operator cannot do. This usually is achieved by introducing new combinations in the population to guarantee population diversity. There are three mutation operators, as shown in Figure 6.10.

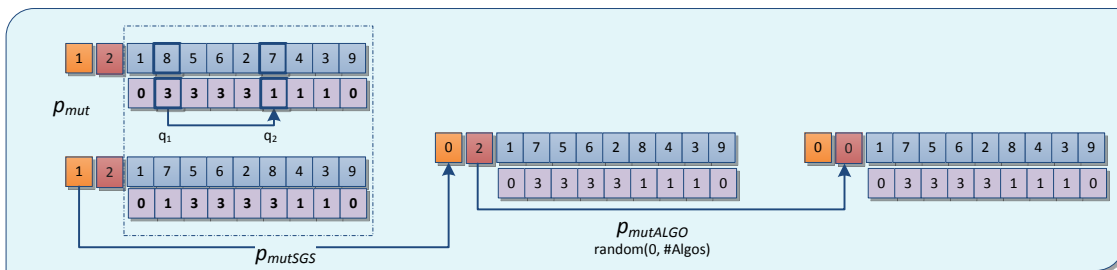


Fig. 6.10 Mutation operator

The first one is applied to the *ActivityList* and the *ModeList* and leads to the exchange of positions between activities  $j_{q_1}$  and  $j_{q_2}$  with a probability of  $p_{mut}$ . The second one is applied to the *SGSGene* where with a probability of  $p_{mutSGS}$  the *SGSGene* will get its complementary



value, therefore 0 changes to 1 and viceversa, this mutation operator is not very useful in the initial iterations as the *SGSGenes* are quite balanced in the population but it is very effective in latter stages where one of the two SGS algorithms has already dominated the population. The last mutation operator is applied to the *AlgoGene* where with a probability of  $p_{mut_{ALGO}}$  the *AlgoGene* will get a random value, another solution algorithm, different from its current value. The overall mutation is accepted only when the output results in a chromosome that leads to a generalised precedence feasible schedule.

### 6.3.7 Selection Strategy

After applying the crossover and mutation operators we have a total population size of  $2 \times POP$  and we want to keep only  $POP$  chromosomes. The selection of which chromosomes should pass to the next generation is done using the two tournament selection method. It involves randomly choosing two candidates from the current population, comparing their fitness values and removing from the population the less fit. Two randomly chosen individuals compete for survival. The one that has worse fitness value "dies" and it is removed from the population. The process is repeated until  $POP$  individuals remain "alive". A fixed population of size  $POP$  will require  $2 \times POP$  tournaments. This selection strategy is used as is, in case of single objective optimisation, otherwise, this selection strategy is embedded in a more complex optimisation process.

### 6.3.8 Multi-Objective Optimisation Process

Although, resource constrained scheduling is an inherently multi-objective problem, it has traditionally been solved considering only one objective, due to the difficulty of defining good heuristics that will handle this multi-objective combinatorial optimisation problem in a flexible and efficient way, leading to good approximations of the optimal solutions.

When constructing the pay off table  $P$ ,  $k \times k$ , where  $k$  the number of objectives to be pursued and  $p_{ij}$  the value of the objective  $i$  for the best schedule from the viewpoint of objective  $j$ , the diagonal defines an ideal schedule which is infeasible in general, but can be used to calculate the distance of the computed solutions from the ideal. Since, usually there is not a single solution that minimises all the objectives simultaneously, we aim at finding a set of solutions where at least one of the objectives is better than the others, the so called non dominated (Pareto) set. One solution  $y$ , a vector consisting of the values of the selected objectives, dominates a solution  $x$  if its corresponding vector is worse or equal to the  $x$  vector. A solution  $x$  is non dominated if there is no other feasible solution that dominates  $x$ .

The aim of meta-heuristics used to solve this kind of problems, is to obtain good approximations of the non dominated set of solutions, spread all over the frontier of those solutions. More specifically, the main goal of any multi-objective optimisation approach is the detection of the highest possible number of Pareto optimal solutions that correspond to an adequately spread Pareto front with the smallest possible deviation from the actual Pareto front. Ideally, the Pareto set found should be a subset of the Pareto optimal set, solutions should be uniformly distributed and diverse in order to provide the project manager a true picture of trade-offs, and the whole spectrum of the Pareto front should be captured, by investigating solutions at the extreme ends of the search space.

In the proposed solution two different approaches are provided: a) a Pareto ranking method where the solution set is ranked according to a predefined dominance rule that prioritises solutions that are non dominated or dominated by a few other solutions and penalises solutions located in regions of the objective functions space which are covered by densely populated sections of the Pareto front, and b) an iterative vector evaluated approach where the objectives are ranked using ANP and then the solution space is split in sub-spaces. Each sub-space is evaluated with respect to a different objective and the weights are used to define the part of each sub-space that will be used to form the aggregate solution space and move to the next iteration.

### 6.3.8.1 Pareto Optimality

The proposed pareto-ranking approach is enfolded in the adaptive genetic algorithm that moderates the solution process. It explicitly utilises the concept of Pareto dominance in evaluating the fitness of the solutions. The population is ranked according to a dominance rule, and then each solution is assigned a fitness value based on its rank in the population, not its actual objective function value. For those solutions that belong to the same Pareto front, if sorting is needed, then the Chebycheff metric, is used. Therefore, the closeness of two solutions  $x, x'$  is calculated as  $\|f(x) - f(y)\| = \max_k |f_k(x) - f_k(x')|$ , where  $k$  are the optimisation objectives and  $f_k$  the objective function corresponding to objective  $k$ .

The proposed approach, takes from NSGA II (Non dominated Sorting Genetic Algorithm) proposed by Deb et al. (2002) but instead of using the "crowding distance" to sort solutions belonging to the same Pareto front, the Chebycheff metric, is used. The initial population  $P_g$  as well as the offspring population are externally handled and the union of these populations,  $R_g$  is given by the moderator GA. This population is sorted according to non domination level and the best solutions of the combined population are passed to the next generation, as shown in Figure 6.11.

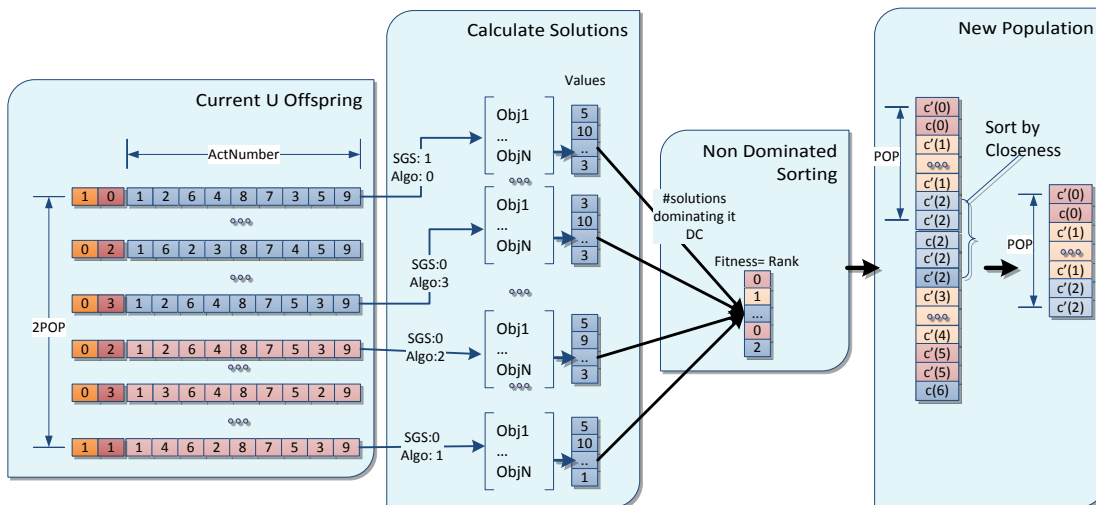


Fig. 6.11 Pareto GA

Each chromosome corresponds to a solution, which is assigned a fitness value based on its non domination level  $F_i$ ,  $i = 1, 2, \dots$ , where  $F_1$  is the best level. During the selection, solutions are taken with the non domination level order and if there is place in the next generation for the entire set of solutions corresponding to a non domination level then it is passed as it is otherwise, the solutions of the non domination level that will be partially copied to the next generation, are sorted based on their closeness, as it is given by the Chebycheff metric, and the most disperse solutions are kept, as shown in Algorithms 6.22 and 6.23.

---

**Algorithm 6.12:** Multi-objective Pareto Selection Strategy

---

```

InputFinput set (input  $R_g$ ) set (output  $P_{g+1}$ ) // calculate non dominated
sets for population  $R_g$ 
Non-Dominated-Sort ( $R_g$ , out  $F_i$ , out  $rank[\ ]$ );
// select chromosomes to pass to the next generation
foreach  $F_i$  do
    set  $sizeF = count(F_i)$ ;
    if  $emptySlots \geq sizeF$  then
        forall the solutions  $j \in F_i$  do
            Copy ( $chromo(j)$ ,  $P_{g+1}$ );
             $emptySlots = emptySlots - sizeF$ ;
        end
    else
        SortClosenessDesc ( $F_i$ );
        while  $emptySlots > 0$  do
            Take chromosome in desc order of closeness;
            Copy ( $chromo(j)$ ,  $P_{g+1}$ );
             $emptySlots = emptySlots - 1$ ;
             $j = j + 1$ ;
        end
    end
    set  $emptySlots = POP$ ;
end

```

---

**Algorithm 6.13:** Non dominated sorting of vector of solutions  $R$ 


---

```

// Calculate non dominated sets  $F_i$ 
foreach chromo  $i \in R$  do
    // Dominated Solutions by chromosome  $i$ 
    set  $DS_i = \emptyset$ ;
    // Domination Counter of chromosome  $i$ 
    set  $DC_i = 0$ ;
    foreach chromosome  $j \in R$  do
        if  $i$  dominates  $j$  then
            |  $DS_i = DS_i \cup j$ 
        else
            |  $DC_i = DC_i + 1$ ;
        end
    end
    if  $DC_i = 0$  then
        // rank is the fitness of chromosome  $i$ 
         $rank[i] = 1$ ;
        //  $F_i$  is the  $i$ -th non dominated front
         $F_1 = F_1 \cup i$ ;
    end
    set counter=1;
    while  $F_{counter} \neq \emptyset$  do
        // define set  $Q$  to temporary store members of the next
        front
        set  $Q = \emptyset$ ;
        foreach chromosome  $i \in F_{counter}$  do
            foreach chromosome  $j \in DS_i$  do
                |  $DC_j = DC_j - 1$ ;
                | if  $DC_j = 0$  then
                    | |  $rank[j] = counter + 1$ ;
                    | |  $Q = Q \cup \{j\}$ 
                | end
            end
        end
        end
         $counter = counter + 1$ ;
         $F_{counter} = Q$ ;
    end

```

---

**6.3.8.2 ANP based Optimality**

In the case that we have already calculated the weights corresponding to each objective, using ANP or some other MCDA method, then a weight vector is formed and used during this process. To approximate the Pareto optimal set by a set of non dominated solutions, a selection algorithm inspired by the the vector evaluated genetic algorithm (VEGA) is used, as shown in Algorithm 6.24. In this algorithm, the input is the union of parent and offspring populations generated by the moderator GA. This population  $P_g$ , with size  $2 \times POP$  is randomly divided

into  $K$  equal sized,  $N_s = 2POP/K$ , sub-populations;  $P_1, P_2, \dots, P_k$ . Then, each solution in subpopulation  $P_i$  is assigned a fitness value based on the corresponding objective function  $f_i$ .

**Algorithm 6.14:** Multi-objective ANP weighted Selection Strategy

```

input :  $P_g$ , ANPweights= $[w_1, \dots, w_k]$ ,  $K$  = total number of objectives,  $N_s = 2POP/K$ 
output:  $P_{g+1}$ 
// for each objective  $k$ 
for  $k = 1 \dots K$  do
    // for each chromosome  $i$ 
    for  $i = 1 + (k - 1)N_s \dots kN_s$  do
        | Fitness ( $i$ ) =ObjectiveFunction ( $k, i$ );
    end
     $P_{sub_k}$  =FormSubPopulation ( $P_g, 1 + (k - 1)N_s \dots kN_s$ );
end
Select from  $P_g$  by subpopulations;
 $P_{g+1}$  =Form ( $w_1 \times P'_{sub_1}, \dots, w_k \times P'_{sub_k}$ );
    
```

Crossover and mutation are performed by the moderator GA but the fitness and the final formulation of the next generation are effectuated by this algorithm. The process of splitting in subpopulations and using different objective functions for each one, is repeated to calculate the fitness of all the given chromosomes. Solutions are selected by comparing subpopulations corresponding to the same objective and keeping the best. The next generation's population is formed from the subpopulations, using proportional selection based on the given weight vector, as sketched in Figure6.12.

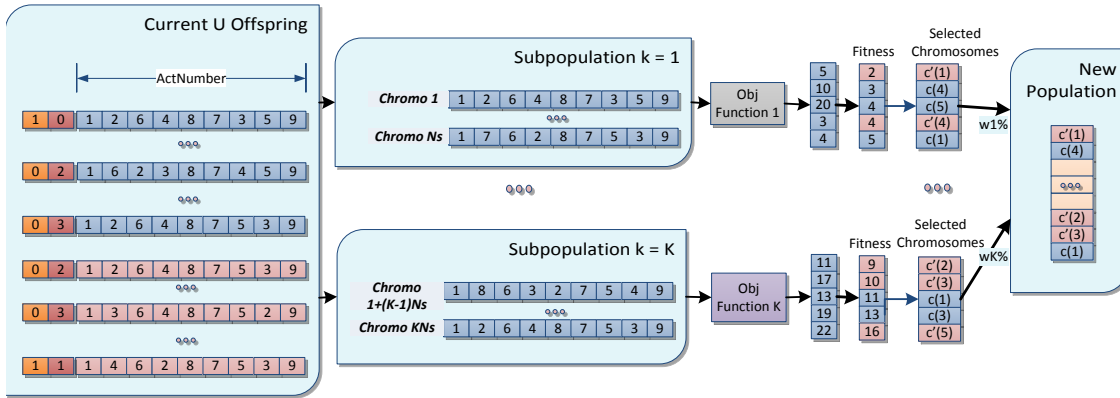


Fig. 6.12 ANP weighted multi-objective optimisation

**6.3.9 Auxiliary Solution Algorithms**

The proposed moderator GA uses chromosomes with embedded a solution algorithm, called auxiliary solution algorithm. These auxiliary algorithms are used only during the fitness calculation. More specifically, each time a fitness value is required instead of using the given by the *SGSGene* schedule generation scheme to produce the schedule corresponding to the chromosome, a more complex process is initialised. The given chromosome, or a group of

chromosomes that have the same *AlgoGene*, depending on the auxiliary algorithm's type, whether it needs a single initial solution or a group of them, is fed to the auxiliary algorithm along with the objective function and after an evolutionary solution process an updated chromosome or group of chromosomes along with their fitness is returned. However, in the auxiliary's algorithm body, the decoding process is effectuated using the SGS defined by the *SGSGene*.

Following, a set of best in class evolutionary algorithms of proven efficiency and effectiveness (Kolisch and Hartmann, 2006; Hartmann and Briskorn, 2010) in solving the single objective RCPSP, M-RCPSP and/or M-RCPSP/max are presented and used as auxiliary solution algorithms. The idea behind all these auxiliary algorithms concept is to use the moderator GA to handle the evolution of which algorithm will be used and diversify the population in a vast search space and use the auxiliary algorithms to try different meta-heuristic methods to search sub-sets of the search space, having a parallel search of different types done by algorithms that are generally good but each type reacts better than the rest in some specific instances.

### 6.3.9.1 Simulated Annealing

The Simulated Annealing (SA) approach proposed by Bouleiman and Lecocq (2003) as it was adapted to the herein described chromosome type and auxiliary algorithms concept, is shown in Algorithm 6.25. The moderator GA feeds this auxiliary algorithm with single chromosomes. All the chromosomes of the current and offspring population of the moderator GA having *AlgoGene* value equal to the SA's ID are used as initial solutions. The chromosome is the same as in the moderator GA but only the *ActivityList* and *ModeList* parts are handled within this algorithm. Each one causes a separate execution of the SA algorithm having one of the chromosomes as starting solution. This solution is used as basis to generate a so-called neighbourhood by slightly perturbing it. The new solution will be accepted and used to proceed the search when it is better than the current one or with a parametrisable probability (cooling temperature) even when it is worse. This parameter initially is set at such value to allow the acceptance of a large proportion of the generated solutions and it is gradually decreased to reduce the acceptance rate of less promising solutions. This prevents the algorithm from getting trapped in a local optimum at early stages. The algorithm is stopped as soon as a stopping criterion reaches a predetermined value.

Neighbourhood generation begins with the current solution and a randomly selected activity. The positions of this activity's latest predecessor *lp* and earliest successor *es* are calculated. Then the new position of the activity is randomly chosen within  $[lp, es]$ . The neighbour is obtained by a cyclical (left/right) shift of all the activities placed between the old and the new positions.

**Algorithm 6.15:** Simulated Annealing

---

```

input : Activities, GeneralisedPrecedenceRelations, ResourceRequirements,
        ResourceAvailabilities, Chromosome, ObjectiveFunction
output: Chromosome, Value
// define SA parameters
set  $N_0, h, T_{0max}, steps, cycles$ ;
// compute ES and LS times using Floyd-Warshall
Calculate-ES-LS (Activities, GeneralisedPrecedenceRelations);
// calculate initial solution's value using SGS
set  $x_0 = Chromosome, f_{x_0} = SGS(Chromosome)$ ;
set  $x_{current} = x_0, f_{x_{current}} = f_{x_0}$ ;
set  $x_{best} = x_0, f_{x_{best}} = f_{x_0}$ ;
// cooling chain
for  $C$  Chains do
    // set cooling temperature
     $T = T_{0max} = 20\%f_{x_0}$ ;
    // set number of moves for the 1st step
     $N_s = N_0$ ;
    for  $S$  Steps do
        // set cooling scheme
         $N_s = N_s \times (1 + h \times steps)$ ;
        for  $N_p$  Neighborhoods do
             $x' = \text{Generate-Neighborhood}(x_{current})$ ;
             $f_{x'} = SGS(x')$ ;
             $\Delta = f_{x'} - f_{x_{current}}$ ;
            if  $\Delta < 0$  then
                 $x_{current} = x', f_{x_{current}} = f_{x'}$ ;
                if  $f_{x'} < f_{x_{best}}$  then  $x_{best} = x', f_{x_{best}} = f_{x'}$ ;
                ;
                if  $f_{x_{best}} = \text{ValueOfCP}$  then exit;
                ;
            else
                if  $e^{-\Delta/T} > y_{random}$  then  $x_{current} = x', f_{x_{current}} = f_{x'}$ ;
                ;
            end
        end
        // update cooling temperature
         $T = a \times T$ ;
    end
end
return ( $x_{best}, f_{x_{best}}$ );

```

---

**6.3.9.2 Genetic Algorithm**

The Genetic Algorithms (GA) used is based on (Hartmann, 1998) genetic algorithm as shown in Algorithm 6.26. In this GA, chromosomes are the same as in the moderator GA but only the *ActivityList* and *ModeList* parts are handled within this algorithm. To obtain the cor-

responding schedule a form of serial schedule generation scheme (SGS) is used as defined by the (unchangeable within this GA) *SGSGene*. Two-point crossover and mutation on the *ActivityList* and *ModeList* are performed as usual. To compute the fitness of a chromosome, first, the related schedule is found and then the value of the objective function of that schedule gives the fitness.

---

**Algorithm 6.16:** Simple Genetic Algorithm

---

**input** : Activities, GeneralisedPrecedenceRelations, ResourceRequirements, ResourceAvailabilities, Chromosomes[], ObjectiveFunction, NumOfMoves, populationSize, crossoverType, probMutation, maxGenerations

**output:**  $P_g$ , Values[]

set generation counter  $g=0$  ;

set POP=populationSize;

set  $P_g$ =Chromosomes[];

**while**  $g < \text{maxGenerations}$  **do**

// two-point crossover operator

$P_{g\text{children}} = 2\text{p-Crossover } P_g$  ;

// one-point random mutation with probMutation

$P_{g\text{children}} = 1\text{p-Mutate } P_{g\text{children}}$  ;

$R_g = P_g \cup P_{g\text{children}}$  ;

// calculate fitness based on ObjectiveFunction

$\text{Values}_g = \text{ObjectiveFunction}(R_g)$ ;

$P_g = \text{TwoTournament}(R_g, \text{Values}_g)$ ;

$g = g + 1$ ;

**end**

---

### 6.3.9.3 Particle Swarm Optimisation

The PSO algorithm used as auxiliary solution algorithm is a new implementation as there were very limited implementation for the RCPSP and all of them had not optimal results. In the proposed approach the moderator GA, provides the initial population, called swarm, of individuals, called particles, that will be iteratively updated using information from both the local and the global search. This initial swarm is formed, as in the case of the GA, by all the chromosomes with *AlgoGene* value equal to the PSO's ID, in the current population of the moderator GA along with its offspring. The particles are the same as in the moderator GA but only the *ActivityList* and *ModeList* parts are handled within this algorithm. Each particle represents a solution, that for PSO is a candidate position. The particle is characterised by its position and velocity. In PSO each iteration's improvement is obtained by adjusting the particle's position and velocity based on it's overall best position (local best) and the best position ever found by all particles (global best). In Algorithm 6.27 is shown the PSO algorithm's formulation.



**Algorithm 6.17:** Particle Swarm Optimisation

---

```

input : Activities, GeneralisedPrecedenceRelations, ResourceRequirements,
        ResourceAvailabilities, Chromosomes[], ObjectiveFunction, NumOfMoves,
        swarmSize, maxIterations
output:  $Swarm_g$ , Values[]
set generation counter  $g=0$  ;
set  $Swarm_g=Chromosomes[]$ ;
set  $weight=1$  // inertia weight
set  $c_1=0.45$  // individual memory
set  $c_2=0.45$  // global memory
EvaluateLocalBest (ObjectiveFunction ( $Swarm_g$ ));
EvaluatGlobalBest (ObjectiveFunction ( $Swarm_g$ ));
while  $g < maxIterations$  do
    forall the  $i \in Swarm_g$  do
        set  $r_1=random(0,1)$ ;
        set  $r_2= random(0,1)$ ;
        // calculate velocity
         $velocity_g[i] =$ 
         $c_1U(0,1)(LocalBest[i] - Swarm_g[i]) + c_2U(0,1)(GlobalBest[i] - Swarm_g[i]);$ 
        // calculate new positions
         $Swarm_{g+1}[i] = Swarm_g[i] + velocity_g[i]$ ;
        // calculate fitness based on ObjectiveFunction
         $Values_g[i]=ObjectiveFunction (Swarm_g[i])$ ;
    end
    EvaluateLocalBest (ObjectiveFunction ( $Swarm_g$ ));
    EvaluatGlobalBest (ObjectiveFunction ( $Swarm_g$ ));
     $g = g + 1$ ;
end

```

---

**6.3.9.4 Tabu Search**

Tabu Search (TS) starts with a single solution used to create a neighbourhood and then all the generated solutions are evaluated and the best one is chosen and used in the next iteration. This process can very easily lead to cyclic moves around a local optimum. In order to avoid this problem a number of previous moves are stored in a memory like data-structure, the so-called tabu list, which is used to reject repeating moves that could lead back to a recently visited solution. Usually, a tabu status can be ignored only in the case that the proposed move would lead to a new overall best solution, based on the so called aspiration rule (Nonobe and Ibaraki, 2002b).

**Algorithm 6.18:** Tabu Search

---

```

input : Activities, GeneralisedPrecedenceRelations, ResourceRequirements,
         ResourceAvailabilities, Chromosome, ObjectiveFunction, NumOfMoves,
         MaxTryAdmissible, MaxTryBetter

output:  $f_{x_{best}}, x_{best}$ 
// define TS parameters
set  $TabuListC = TabuListNC = 0$  set  $TabuTenC = TabuTenNC = \sqrt{N/2}$ ;
// calculate initial solution's value using SGS
set  $x_0 = Chromosome$ ,  $f_{x_0} = Value(SGS(Chromosome))$ ;
set  $x_{current} = x_0$ ,  $f_{x_{current}} = f_{x_0}$ ; set  $x_{best} = x_0$ ,  $f_{x_{best}} = f_{x_0}$ ;
// critical CPact and non critical nCPact activities
Calculate-ES-LS (Activities, GeneralisedPrecedenceRelations);
foreach  $i \in Activities$  do
    if  $ES(i) = LS(i)$  then  $CPact = CPact \cup \{i\}$ ;
    else  $nCPact = nCPact \cup \{i\}$ ;
end
while  $NotFoundAdm \leq MaxTryAdmissible$  &  $NotFoundBetter \leq MaxTryBetter$  do
    // create list of candidate moves
    while  $cnt < NumOfMoves$  do
        tempActList=ActList;
        random  $Q_1, Q_2$ ;
        tempActList= MoveAct (ActList, $Q_1, Q_2$ );
        if CheckFeasible (ActList)=TRUE then
            SaveMove ( $Q_1, Q_2$ , CandidateList);
             $cnt = cnt + 1$ ;
        end
    end
    // choose the best admissible move
    set FindAdmissible= FindBetter=FALSE;
    foreach  $move(Q_1, Q_2) \in CandidateList$  do
        tempActList=MoveAct (ActList, $Q_1, Q_2$ );
        if  $value(SGS(tempActList)) < f_{x_{current}}$  then
            if CheckTabuStatus =TRUE then
                 $move(Q_1, Q_2)$  tabu-restricted
            else
                if CheckAspirationTest =TRUE then
                    set BestMove= $move(Q_1, Q_2)$ ;
                    FindAdmissible=TRUE;
                    NotFoundAdm=0;
                end
            end
        end
    end
    // make the best admissible move
    ActList= MoveAct (ActList, BestMove);
    if  $value(SGS(ActList)) < f_{x_{best}}$  then
         $f_{x_{best}} = value(SGS(ActList))$ ;
        FindBetter=TRUE;
    end
    if FindAdmissible=FALSE then NotFoundAdm=NotFoundAdm+1;
    ;
    if FindBetter=FALSE then NotFoundBetter=NotFoundBetter+1;
    ;
    UpdateTabuLists ();
end

```

---

The Tabu Search algorithm implemented here takes from the algorithm proposed by Nonobe and Ibaraki (2002b) with the difference that the moderator GA feeds this auxiliary algorithm with single chromosomes representing initial solutions. All the chromosomes of the current and offspring population of the moderator GA having *AlgoGene* value equal to the TS's ID are used for this purpose and each of them causes an independent execution of the TS algorithm, which is presented in Algorithm 6.28



## Chapter 7

# Computational Results and Evaluation

### 7.1 Implementation

The proposed system, consisting of the model, the preprocessing algorithms and the moderator algorithm along with the auxiliary solution algorithms described in chapter 6, was implemented using the *C#.NET* programming language. C# is an object-oriented programming language from Microsoft that aims to combine the computing power of C++ with the programming ease of Visual Basic. C# is based on C++ and contains features similar to those of Java.

The generated code provides a simple console application used for the experiments following described. Furthermore, the same routines were used in a Microsoft Project Add-In that provides the end-user with an easy to use interface from where project data in a predefined format can be converted to MS-Project files, the proposed algorithms along with the best in class algorithms from the literature can be applied to a specific data set and the results can be visualised in the same environment, as shown in Figure 7.1.

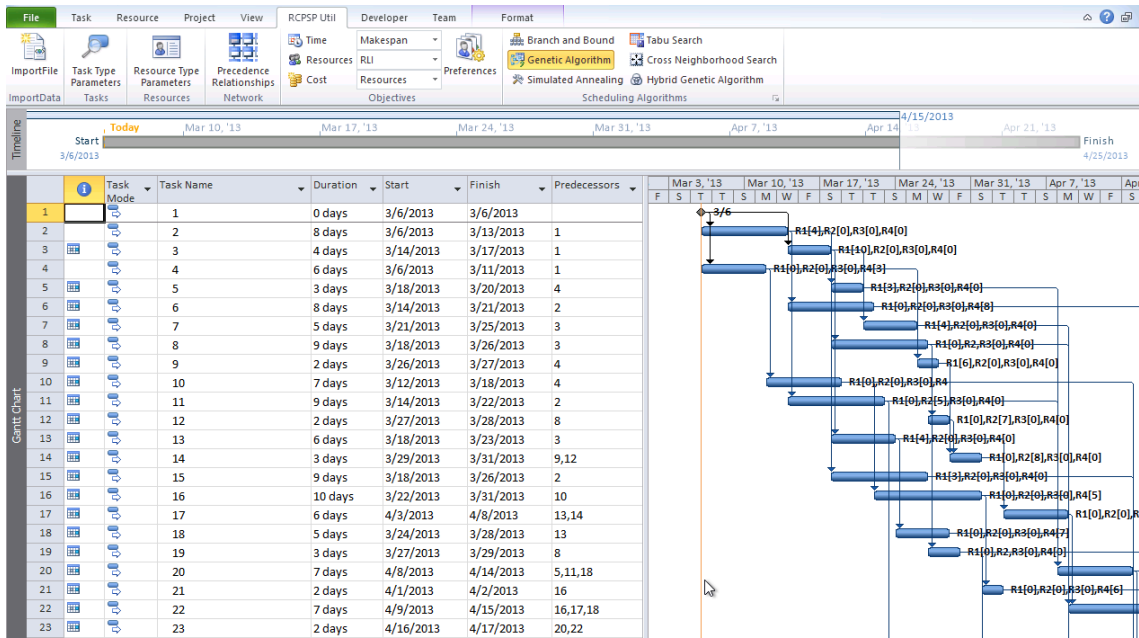


Fig. 7.1 EMO-RCSPS: a Ms Project Add-In

The Microsoft Project Add-In, called EMO-RCSPS, has been developed as an outcome of this Thesis. The idea was to enhance an existing application, which is widely used by project managers as MS Project, to provide a simple way to apply the proposed model and algorithms in real cases. EMO-RCSPS's functionality (in detail presented in Appendix C) can be grouped in two categories of new features:

- Input data handling features that provide the user ways to add information about the tasks, like multiple modes of execution, variable usage of resources on each task, handling of non renewable resources, etc.
- Scheduling features, which include the proposed algorithms along with other best in class algorithms for project scheduling, ways of handling multi-objective optimisation and visualising the resulting solution scenarios.

More specifically, the user can either import data from text files or create a project using the well known environment of MS Project. The first option is usually used by researchers as it supports all the common formats provided by PSP Lib (Kolisch et al., 1995). In the second case the project manager creates a project in MS Project by adding tasks, phases, resources and precedence relationships, in the usual way. Then the tasks that can have multiple modes of execution are selected and the modes are either automatically generated using the formula  $effort = work * duration$  and calculating all the possible integer combinations or the manager manually enter specific modes. Similarly are defined which activities are splittable and whether it is an automatic unitary split or splits can be done on specific user defined points (as percentage of execution). In cases that maximal lags should be added, the tasks and amount of lag in time units is manually entered as MS Project doesn't support this feature. The non renewable resources are added using a custom resource type and for each one a calendar setting the available amounts per time period is defined. Finally, resource demands that vary over time are also defined at a task level using a calendar like matrix to enter the resource requirements per time period of execution.

After having set up the project tasks, precedence relationships, resource requirements and availabilities, it is the time of the optimisation objectives set up and optional definition of weights for the selected objectives. Weights can either be entered directly or using ANP, that is externally implemented. Finally, the optimisation algorithm to be used is selected along with related options about the algorithm and the number of solution scenarios that should be generated.

## 7.2 Experiments design

Two different experiments were designed and implemented: a) a comparison of the results given by the proposed holistic model and algorithm to each specific problem that was integrated in our model, as to prove that the proposed method leads to at least as good results as the best known for each variation. In other words this experiment has as its goal to validate that the proposed algorithm solves efficiently each of the problem variations that were incorporated in the model without loss in quality of results or execution time. b) a comparison of the multi-objective approach to the single-objective results given on the above test cases adapted for the multi-objective case to illustrate the differentiation of the results based on the objectives that were set and which of the two multi-objective approaches was used.

As far as the algorithm parameters are concerned, we have defined their range of values through some rough computational tests. However, it should be noted that a precise tuning of these parameters would be needed to achieve the best performance of the algorithms, but as the scope of both experiments is the validation this is not of major concern.

## 7.3 Experimental Comparison to best in class algorithms

Five different standard sets of benchmark instances from the literature have been used, one for each RCPSP variation that was integrated in the proposed model. Therefore, benchmark instances for the RCPSP, PRCPSP, MRCPSP, RCPSP/max and RCPSP/t were used. These instances are all available in the project scheduling problem library *PSPLIB* except the RCPSP/t instances (for detailed information the reader is referred to Kolisch and Sprecher 1997 and were constructed by the project generator *ProGen*. In this study, for RCPSP, MRCPSP, RCPSP/max and RCPSP-t, problem instances were used. For RCPSP, PRCPSP and RCPSP/t the first set consists of 480 instances and the second set consists of 600 instances that have been generated by varying three parameters: network complexity (*NC*), resource factor (*RF*), and resource strength (*RS*). The network complexity reflects the average number of immediate successors of an activity or in other words the average of non-redundant precedence relations per activity. The resource factor is a measure of the number of resources requested per job. The resource strength describes the scarceness of the resource capacities as the ratio of available amount of resources of a specific type minus the minimum demand for this resource type to the difference of the minimum resource demand from the corresponding max resource demand when activities are scheduled at their critical path earliest start. For MRCPSP, the *c15*, *c21* and *j10* PSPLIB datasets were used. For MRCPSP/max the test set *MM30* with 270 instances with 30 activities, 3, 4, or 5 execution modes, 3 renewable resources, and 3 nonrenewable resources consisting of 270 instances with 100 activities, 3, 4, or 5 execution modes, 3 renewable resources, and 3 nonrenewable resources, were used.

Following, the average percentage deviation from the optimal makespan or from the best lower and upper bounds (for instances for which only heuristic solutions are known) as stated in the *PSPLIB* library at the time this research was performed, is reported and compared to the best known results for each instance. The goal is to validate the proposed holistic model by giving at least the same results with each problem type specific solution method and with the same or better accuracy.

For each instance 100 repetitions of the experiment were performed to get the average values. The experiments were executed using a computer with the following characteristics: Intel(R) Core(TM) 2 Duo CPU P8600 at 2.40 GHz and RAM 8.00 GB. An excerpt of the results obtained from this experiment for RCPSP, MRCPS and MRCPS/max are shown in Figures A.1 - A.3.

An excerpt of the results of the experiment is shown in Figures A.1 - A.3. The results gained from the execution of the proposed algorithm are compared to the optimum value, when it is known or the best value reached by any other heuristic in all other cases. Furthermore, the frequency of the optimum value is counted as to prove the effectiveness of the algorithm. In some cases the values calculated by this algorithm are lower than the known best values.



Filename	Min Dur	Max Dur	Optimal	Aver.Dev.	Frequency
J301_1.RCP	43	45	43	0,0%	70%
J301_2.RCP	47	51	47	0,0%	90%
J301_3.RCP	47	47	47	0,0%	100%
J301_4.RCP	62	62	62	0,0%	50%
J301_10.RCP	45	46	45	0,0%	60%
J303_1.RCP	72	72	72	0,0%	100%
J303_2.RCP	40	40	40	0,0%	100%
J303_3.RCP	57	57	57	0,0%	100%
J303_4.RCP	98	98	98	0,0%	100%
J303_5.RCP	53	53	53	0,0%	100%
J303_6.RCP	54	54	54	0,0%	100%
J303_7.RCP	48	48	48	0,0%	100%
J303_8.RCP	54	54	54	0,0%	100%
J303_9.RCP	65	65	65	0,0%	100%
J303_10.RCP	59	59	59	0,0%	100%
J304_1.RCP	49	49	49	0,0%	100%
...	...	...	...	...	...
J3033_1.RCP	65	65	65	0,0%	100%
J3033_2.RCP	60	60	60	0,0%	100%
J3033_3.RCP	55	56	55	0,0%	90%
J3033_4.RCP	77	78	77	0,0%	80%
J3033_5.RCP	53	53	53	0,0%	100%
J3033_6.RCP	59	59	59	0,0%	100%
J3033_7.RCP	58	58	58	0,0%	100%
J3033_8.RCP	61	61	61	0,0%	100%
J3033_9.RCP	65	68	65	0,0%	20%
J3033_10.RCP	53	53	53	0,0%	100%
J3034_1.RCP	68	68	68	0,0%	100%
J3034_2.RCP	44	44	44	0,0%	100%
J3034_3.RCP	69	69	69	0,0%	100%
J3034_4.RCP	67	67	67	0,0%	100%
J3034_5.RCP	63	63	63	0,0%	100%
J3034_6.RCP	52	52	52	0,0%	100%
J3034_7.RCP	58	58	58	0,0%	100%
...	...	...	...	...	...
J3048_1.RCP	63	63	63	0,0%	100%
J3048_2.RCP	54	54	54	0,0%	100%
J3048_3.RCP	50	50	50	0,0%	100%
J3048_4.RCP	57	57	57	0,0%	100%
J3048_5.RCP	58	58	58	0,0%	100%
J3048_6.RCP	58	58	58	0,0%	100%
J3048_7.RCP	55	55	55	0,0%	100%
J3048_8.RCP	44	44	44	0,0%	100%
J3048_9.RCP	59	59	59	0,0%	100%
J3048_10.RCP	54	54	54	0,0%	100%

Fig. 7.2 Single objective execution of j30 instances

Filename	Min Dur	Max Dur	Optimal	Aver.Dev. Opti	Frequency c
c154_3.mm	34	34	34	0,00%	100%
c158_3.mm	25	25	25	0,00%	100%
c158_4.mm	32	32	32	0,00%	100%
c159_1.mm	18	21	18	0,00%	99%
c159_2.mm	25	26	29	-13,79%	97%
c159_3.mm	22	24	22	0,00%	100%
c159_4.mm	17	21	17	0,00%	100%
c159_5.mm	21	22	21	0,00%	100%
c159_6.mm	20	22	20	0,00%	100%
c159_7.mm	24	28	24	4,17%	84%
c159_8.mm	34	40	34	0,00%	100%
c159_9.mm	28	29	28	0,00%	100%
c159_10.mm	32	32	32	0,00%	90%
c1510_1.mm	21	21	21	0,00%	100%
c1510_2.mm	17	18	17	0,00%	100%
c1510_3.mm	23	25	23	0,00%	100%
c1510_4.mm	39	42	39	0,00%	98%
c1510_5.mm	13	13	13	0,00%	100%
c1510_6.mm	32	32	32	0,00%	100%
...	...	...	...	...	...
c214_6.mm	36	36	36	0,00%	100%
c216_8.mm	36	36	36	0,00%	100%
c217_1.mm	40	41	40	0,00%	98%
c219_1.mm	28	30	30	0,00%	99%
c219_2.mm	29	32	29	0,00%	96%
c219_3.mm	26	28	26	0,00%	99%
c219_4.mm	26	28	26	0,00%	98%
c219_5.mm	29	30	29	0,00%	97%
c219_6.mm	22	24	22	4,55%	88%
c219_7.mm	25	28	29	0,00%	98%
c219_8.mm	21	24	21	0,00%	99%
c219_9.mm	28	33	28	0,00%	98%
c219_10.mm	21	21	21	0,00%	100%
c2110_1.mm	21	23	21	0,00%	99%
...	...	...	...	...	...
j102_2.mm	18	21	20	0,00%	98%
j102_4.mm	17	17	18	0,00%	100%
j102_5.mm	16	17	16	0,00%	99%
j102_6.mm	16	16	16	0,00%	100%
j102_7.mm	25	25	25	0,00%	100%
j102_9.mm	15	15	17	0,00%	100%
j102_10.mm	33	33	33	0,00%	100%
j103_2.mm	13	13	13	0,00%	100%
j103_3.mm	19	19	19	0,00%	100%
j103_4.mm	23	23	23	0,00%	100%
j103_5.mm	19	21	21	0,00%	100%

Fig. 7.3 Single objective execution of MRCPS P instances

Filename	Min Dur	Max Dur	UB	Aver.Dev. Optimum	Frequency of Opt
psp1.sch	42	49	42	0,0%	70%
psp2.sch	33	38	33	0,0%	90%
psp3.sch	46	46	46	0,0%	100%
psp4.sch	33	36	33	0,0%	50%
psp5.sch	25	29	25	0,0%	60%
psp6.sch	33	33	33	0,0%	100%
psp8.sch	39	39	39	0,0%	100%
psp9.sch	33	33	33	0,0%	100%
psp10.sch	32	32	32	0,0%	100%
psp11.sch	28	28	28	0,0%	100%
psp12.sch	25	25	25	0,0%	100%
psp13.sch	30	30	30	0,0%	100%
psp14.sch	35	35	35	0,0%	100%
psp15.sch	28	28	28	0,0%	100%
psp16.sch	26	26	26	0,0%	100%
psp17.sch	42	42	42	0,0%	100%
...	...	...	...	...	...
psp47.sch	26	26	26	0,0%	100%
psp48.sch	31	31	31	0,0%	100%
psp49.sch	18	18	18	0,0%	100%
psp50.sch	24	24	24	0,0%	100%
psp51.sch	26	26	26	0,0%	100%
psp52.sch	30	30	30	0,0%	100%
psp53.sch	28	28	28	0,0%	100%
psp54.sch	25	25	25	0,0%	100%
psp55.sch	38	38	38	0,0%	100%
psp56.sch	37	37	37	0,0%	100%
psp57.sch	30	30	30	0,0%	100%
psp58.sch	26	26	26	0,0%	100%
psp59.sch	24	24	24	0,0%	100%
psp60.sch	29	29	29	0,0%	100%
psp61.sch	40	40	40	0,0%	100%
psp62.sch	38	38	38	0,0%	100%
psp63.sch	30	30	30	0,0%	100%
psp64.sch	36	36	36	0,0%	100%
psp65.sch	25	25	25	0,0%	100%
psp66.sch	30	30	30	0,0%	100%
psp67.sch	38	38	38	0,0%	100%
psp68.sch	31	31	31	0,0%	100%
psp69.sch	32	32	32	0,0%	100%
psp70.sch	22	22	22	0,0%	100%
psp71.sch	33	33	33	0,0%	100%
psp72.sch	20	20	20	0,0%	100%
psp73.sch	34	34	34	0,0%	100%
psp74.sch	39	39	39	0,0%	100%
psp75.sch	33	33	33	0,0%	100%

Fig. 7.4 Single objective execution of MRCPSP/max instances

In Table 7.1, a summary of the experimental results is shown (see Appendix B for the analytical results). More specifically, the first column is the instance name, the second column shows the minimum duration calculated, the third the maximum duration calculated, the given optimum value is shown in the fourth column and then the average deviation from optimum and the frequency of appearance of the optimum values are shown. These results reveal that in all cases the proposed algorithm gives the same optimal or lower bound results with those that are published in PSPLib and in most cases has a higher accuracy with a percentage of deviation from the best known value lower than 2% for each category of cases. Therefore, the aim to be at least as good as the best known algorithm has been achieved.

**Table 7.1** Comparative results for single-objective instance

Instances	Average Deviation	Max Deviation	Optimal/UB)
<b>RCPSP J30</b>	0.25%	3%	96.7%
<b>RCPSP J120</b>	1.42%	8%	34.46%
<b>PRCPSP J30</b>	0.12%	2.5%	98.7%
<b>PRCPSP J120</b>	1.21%	5%	42.73%
<b>MRCPSP C15</b>	0.23%	1%	98.9%
<b>MRCPSP C21</b>	0.01%	1%	99.9%
<b>MRCPSP J10</b>	0.01%	0%	99.9%
<b>RCPSP-t J30</b>	0.05%	1%	99.7%
<b>RCPSP-t J120</b>	0.22%	1.5%	99.5%
<b>RCPSPmax J30</b>	0.12%	1.8%	90.12%

## 7.4 Experimental results for multi-objective optimisation

The problem instances used in this work for the multi-objective experiment were based on some of the instance sets used for the previous experiment. All instances consider the existence of two renewable and two nonrenewable resources and a maximum of three direct successors. The number of activities is 30 and 120 and each set consists of 10 instances. Each activity has a maximum of three alternative modes and minimal and maximal lags as well as generalised precedence constraints have been added based on the RCPSP-GPR instances. Furthermore, variable resource demand and availability is taken from the RCPS/t instances used in the previous experiment. The original instances have been modified due to the introduction of multiple objectives. Due dates and penalty factors for overconsumption of resources have been defined for each problem along with maximum resource related cost. It should be noted that the definition of the extra parameters may change considerably the difficulty of the initial problems. The resulting problems will be referred here as P30 and P120, according to the number of activities in each group of instances.

For each instance 10 repetitions of the experiment were effectuated to get the best values. The implementation of the algorithm was made in *C#.NET* programming language. The experiments were executed using a computer with the following characteristics: Intel(R) Core(TM) 2 Duo CPU P8600 at 2.40 GHz and RAM 8.00 GB.

In Table 8.1, an excerpt of the experimental results is shown, as the results cannot be compared to any other data set of the literature due to differences both on input data and objectives being pursued. More specifically, the results for the multi-objective cases are strongly related to the selected objectives and the used weights, even more in our case that the problem itself is also different as it is an extended version of the existing ones. Therefore, the comparison

to similar approaches would not provide additional information about the proposed solution approach.

**Table 7.2** Comparative results for multi-objective instances

Instance	Algorithm	Makespan	RLI	Cost	Robustness
<b>J301.1.1</b>	Pareto	43	124.10	4900	30
<b>J301.1.1</b>	ANP	45	133.10	4500	32
<b>J301.1.1</b>	Single Obj.	43	124.10	4900	30
<b>J301.2.2</b>	Pareto	47	176.18	5500	32
<b>J301.2.2</b>	ANP	47	173.28	5600	30
<b>J301.2.2</b>	Single Obj.	47	176.18	5500	32
<b>J301.3.6</b>	Pareto	47	153.02	5200	46
<b>J301.3.6</b>	ANP	47	158.65	5300	48
<b>J301.3.6</b>	Single Obj.	47	161.65	5600	42
<b>J301.4.7</b>	Pareto	62	185.10	6600	35
<b>J301.4.7</b>	ANP	64	187.63	5800	28
<b>J301.4.7</b>	Single Obj.	62	185.09	6800	21
<b>J3034.9.3</b>	Pareto	60	207.33	5400	40
<b>J3034.9.3</b>	ANP	60	208.32	5400	40
<b>J3034.9.3</b>	Single Obj.	60	207.82	5300	37

The results show that the combined usage of the two multi-objective approaches showcases more efficiently the available alternative schedules giving to the project manager more options to choose from. Furthermore, when comparing the results of the multi-objective approach to the single objective we see that often (57% of the cases) the multi-objective approach enhanced the solution given by the single-objective in relation to the other objectives without great loss (more than 5%) on the primary objective. It would be proper to compare the above results to other multi-objective approaches recently proposed, however shared datasets, are not available.



## **Chapter 8**

### **Case Study**

#### **8.1 Introduction**

In this section we focus on the multi-objective solution processes. Main goal is to show that the proposed model is usable, covers a great variety of different situations and returns a set of schedules that cover the goals set by the project manager.

To illustrate the process we take the phase of preliminary design of an actual project for the development of large scale spatial data infrastructure for terrestrial areas network, aiming at an accurate marking-out of the outer limits of terrestrial sites, updating, describing and delineating of terrestrial habitat types and complementing and correcting the existing databases. This project from now on will be referred as GIS-project. The preliminary sketch of activities and their interrelations, along with resource availabilities, resource needs by task and time period and types of relations among the activities are the initial inputs. Based on the proposed model and after interviewing the project manager the constraints, objectives and their weighting and degrees of freedom on the given constraints are decided. These degrees of freedom represent those constraints that can be translated in penalty functions to simplify the solution process and raise the possibilities to get feasible solutions and the hard limits on them. For example, after the discussion we know that although the cost should be minimised and there is a budget for the specific project, it is not a hard limit as it can vary between an upper and lower bound and even if it is outside these limits the proposed schedule can still be acceptable if it has very good resource profiles and makespan, as these two are the objectives with primary importance. Having modelled the project and defined all the needed inputs, we run the proposed algorithm with three different settings: a) as a single objective, b) using the given weights for the requested objectives and c) looking for pareto-optimal schedules and the three best schedules got by each method, are presented to the project manager. In case that the results are not satisfactory, the number of returned results per type would be changed and more scenarios with the same or even different priority settings would be generated until a solution that would fit to the precise needs of the specific company and project would be generated.

#### **8.2 Initial data**

The process begins with the definition of the activities, their precedence relationships, duration and resource requirements. This step is executed using for example the MS Project

software, as shown in Figure 8.1. However, this draft cannot give a complete picture of the situation as it does not include information about the alternative ways that some of the tasks can be executed, minimal and maximal lags have not been defined and resource availabilities and demands are considered stable over time. Some of these issues can be handled within the selected software tool, e.g. the definition of calendars for the definition of the resource availabilities but others require the usage of EMO-RCPSP (the proposed add in), as in the case of the multiple execution modes or the definition of maximal lags. The initial data show

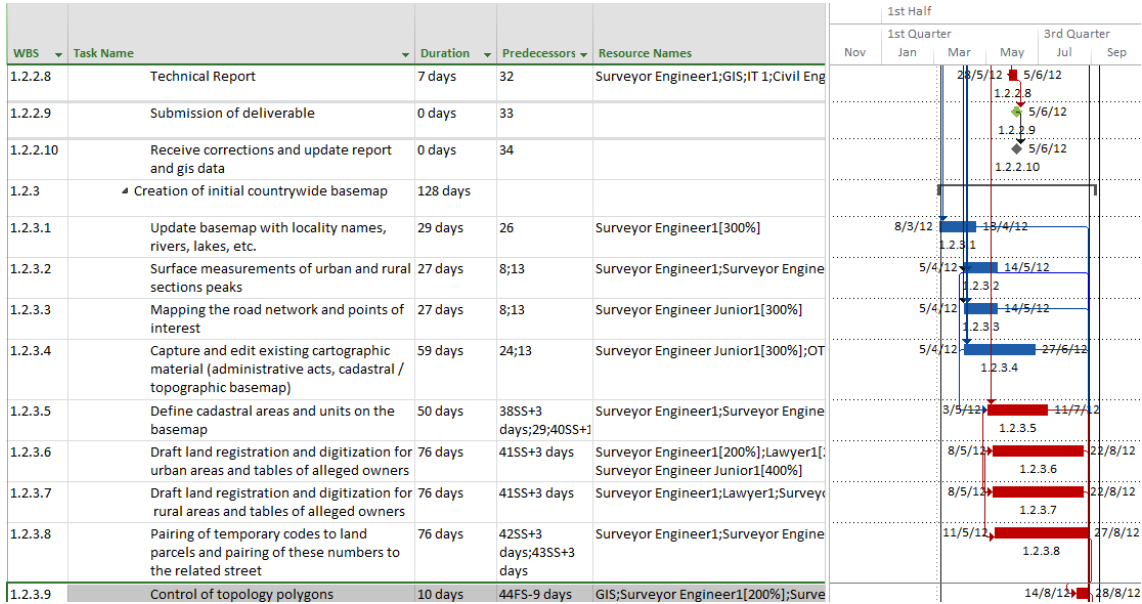


Fig. 8.1 Initial data of the GIS project

that even in the simplest approach there are issues related to the availability of the resources, especially in the case of junior lawyers and surveyor engineers, as shown in Figure 8.2. Therefore, the next step consists in discussing with the project manager in order to gather more details about the project at hand, its characteristics and all those environment parameters that can affect the way that the tasks will be executed or even the goals of the project itself. This discussion will lead to decisions about: a) the components of the project, b) the goals to be pursued and c) existing constraints, soft and hard limits.

### 8.2.1 Execution modes

The majority of tasks can be executed in different modes that can be automatically defined by keeping the total work unchanged and modifying the number of resource used and the duration of the task. Furthermore, the existence of similar types of resources having different performance rate and correspondingly different cost (e.g. Civil Engineer and junior Civil Engineer) leads to another group of different execution modes. Additionally, there are a few tasks that can be executed using a different combination of resources e.g. administrative work can be done by administrative staff but if needed can also be executed by a junior



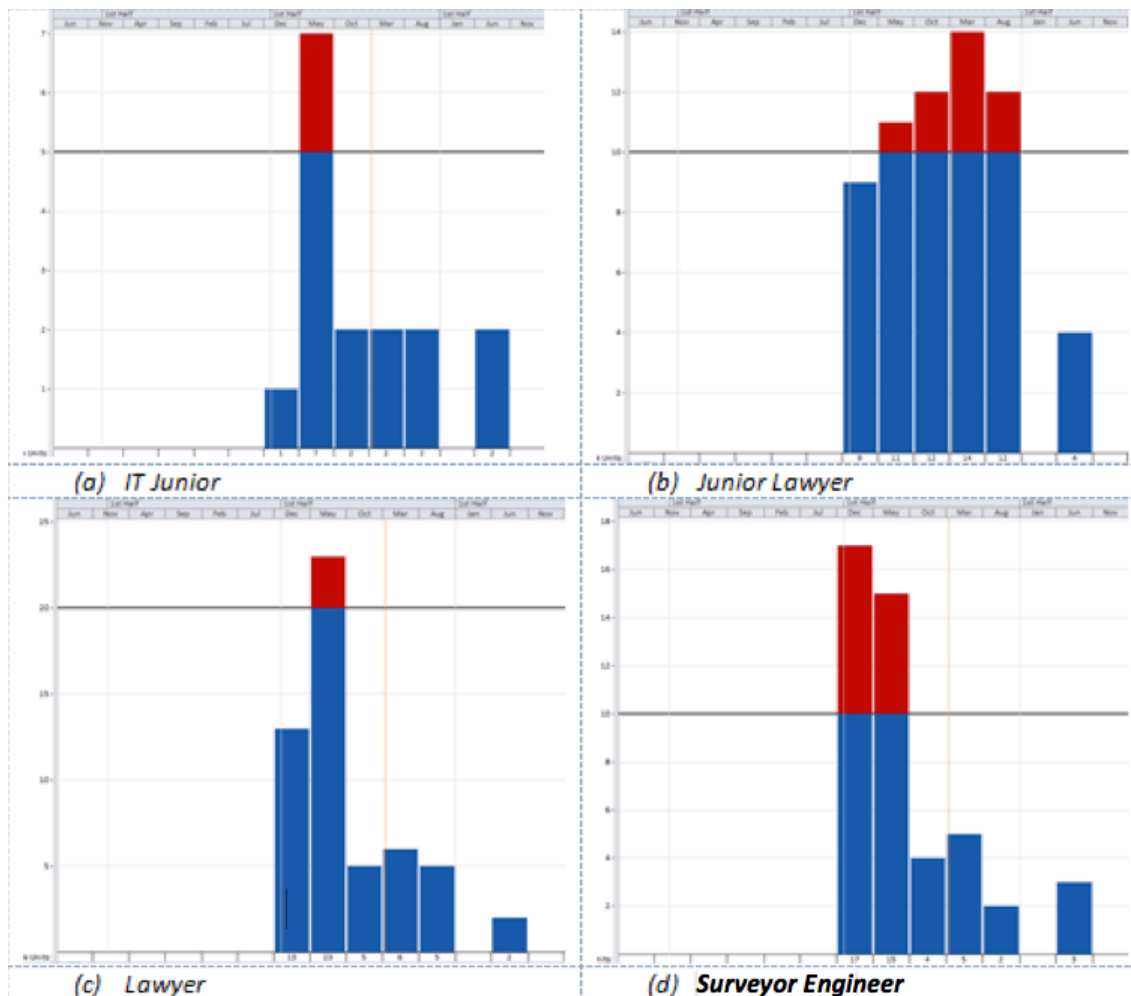


Fig. 8.2 Overallocated resources

lawyer or the digitization process can be done by a resource and a low cost scanner or be almost automated using a more efficient professional scanner and an IT person to monitor the process, as shown in Figure 8.3

On the other hand, having a project consisting of almost 200 tasks makes the definition of multiple modes for each and every task a tedious process. Therefore, the most effective solution is to set automated multiple execution modes on all the tasks and then for tasks that have substantially big duration when compared to the total project duration or tasks that cause delays to the project due to their resource requirements, specific alternative execution modes can be defined. The definition of multiple execution modes should be an iterative process starting from the simplest cases and adding more alternatives as the resulting solution scenarios don't fit the specific situation.

Task Name	Duration	Resource Names
<b>Update basemap with locality names</b>	<b>29 days</b>	Surveyor Engineer1[3]
	<b>29 days</b>	Surveyor Engineer Junior1[6]
	<b>58 days</b>	Surveyor Engineer Junior1[3]
	<b>29 days</b>	Surveyor Engineer Junior1[3], Other1[3]
<b>Draft land registration and digitization for urban areas and tables alleged</b>	<b>76 days</b>	Surveyor Engineer1[2], Lawyer1[2], Surveyor Engineer Junior [4]
	<b>76 days</b>	Lawyer Junior1[2], Surveyor Engineer Junior [8]
	<b>30 days</b>	Surveyor Engineer1[4], Lawyer1[4]
	<b>100 days</b>	Lawyer Junior1[4], Surveyor Engineer Junior [2], Other1[8]

Fig. 8.3 Defining multiple execution modes

### 8.2.2 Preemption

Generally, preemption allows activities to be stopped and restarted later on at no additional cost. Some activities can be preempted at any time without creating any problem in their execution, other activities can be preempted only at specific time instances that define the completion of a sub-task and a few other activities cannot be preempted at any stage of their execution otherwise either the cost will be critically raised or the quality of the end product will be raised. For example, in this GIS-project all tasks requiring external measurements on specific locations would not be sensible to be preempted as this would lead to send at different times at the same location, sometimes far away from the land offices, the same team, raising the cost and multiplying the set up times for the needed equipment. Furthermore, there are tasks that although can be split, it does not make sense to split them before the completion of a specific sub-task, e.g. the "Installation of hardware and software" can be split on the completion of each hardware part but not in the middle of the installation of a software component, as shown in Figure 8.4

Task Name	Duration	Preemption	Preemption Points
Installations of equipment, hardware and software	20 days	Y	2,5,5,5,1,1,1
Installation of Fire and Security systems	15 days	N	
New hires training	25 days	Y	5,5,5,5,5
Preparation of hard copy material	20 days	Y	unitary

Fig. 8.4 Defining task preemption

### 8.2.3 Variability of resource availabilities and requirements

The variability of resource availabilities is defined using the resource calendars. In this GIS project the definition of the non renewable resource availabilities e.g. equipment, hardware, office supplies was more important than that of human resources, as at this stage, there are not named resources but just generic resource types with unknown work schedules. The only renewable resources that their availabilities could and were defined in detail, were the GIS expert, that is only one for all the project and the senior IT staff, Civil Engineers, Topographers and Lawyers, due to their pre-existing commitments on other projects.

Furthermore, there were specific tasks that required different type of resources at different stages of the task, e.g. the task to correct the data base entries based on the results of duplicate entries check and then identify the correct entry and update the entries can be handled as a three steps process that does not require all the resource types available from the beginning of the task to its completion, as shown in Figure 8.5. In this kind of situations the variability of resource requirements comes at hand as it is quite is to define in which period of the task each resource type is required and in what extend, e.g. the IT resource as soon as provides the lists of the duplicate entries can be dismissed from that task, likewise the lawyer is minimally needed during the phase of updating the entries.

Task Name	Duration	Sequence of actions by role
Correct and add notes to data entries based on the results of legal control and cross verification of data using automated validation algorithms	20 days	Validate entries (IT) --> Legal control (Lawyer) --> Correct entries (Surveyor Engineer)
Cross check and validate duplicate entries referring to land owners	14 days	Validate entries (IT) --> Identify owners (Lawyer, Other) --> Correct entries (Surveyor Engineer)

Fig. 8.5 Defining task resource requirements per step

Another way of handling this type of situations would be to split each task in this category, in subtasks and arrange accordingly the resource requirements. In small projects the latter method would be preferred but on medium and large projects it would add unnecessary complexity.

### 8.2.4 Constraints

The GIS Project is characterised by deadlines attached to the completion of each phase (there are three phases in total) of the project. The deadlines are set by the Greek Registry Office

and are considered hard limits. Furthermore, tasks related to the land offices operation and the submission of statements by the landowners have very well defined start and end dates related to the requirements set by the contracting authority. The set of deadlines are modelled as time based soft constraints, therefore schedules missing the deadlines will not be omitted but a penalty factor weighted by the criticality of the corresponding deadline will be added to the value of the time objective (project duration).

The rest of the constraints originate from the precedence relations, the resource availabilities and requirements and the additional relations generated by allowing the split of specific activities.

### **8.2.5 Objectives**

The project manager selected three optimisation objectives: the total project duration, the cost and the smoothness of the resource profiles based on the company's requirements and the specifics of this GIS project. The reasoning behind this choice was based on the following facts:

- the project duration, although difficult to minimise due to the existence of inflexible dates for the execution of specific tasks, e.g. the gathering of statements cannot be done earlier than the required dates, is very important because it can be considered as a competitive advantage for the customer,
- the project cost which is related to the number of resources used, the type of contracts signed by each resource type (salaries, work based contracts, temporary contracts), the work time of each resource and its cost, should be minimised as to make the project profitable for the company
- smoothness of resource profile, to avoid unwanted hiring and firing of staff during the project execution that could lead to lowering the performance of the project team.

These objectives were weighted by the project manager using the ANP method and the model described in chapter 6 and using the ANP Solver software tool (<http://kkiry.simor.mech.ntua.gr/Rokou/ANPWEB/>), shown in Figure 8.6.

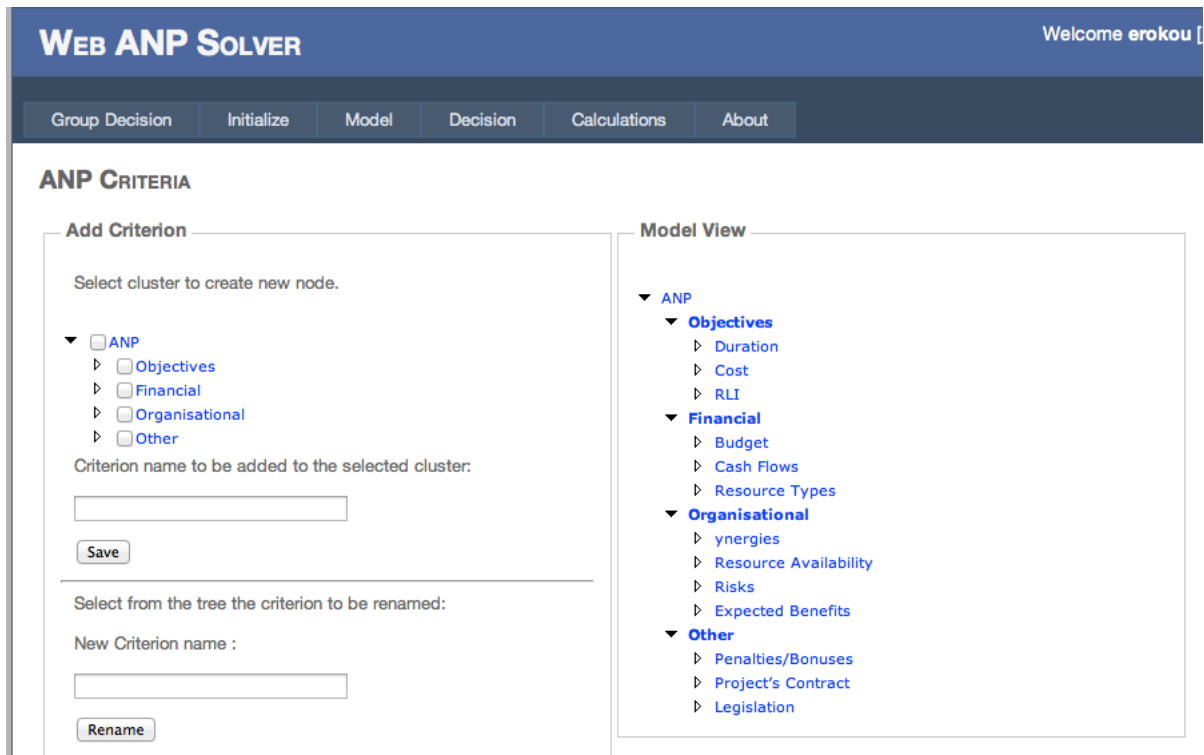


Fig. 8.6 Weighting the objectives using the ANP Solver

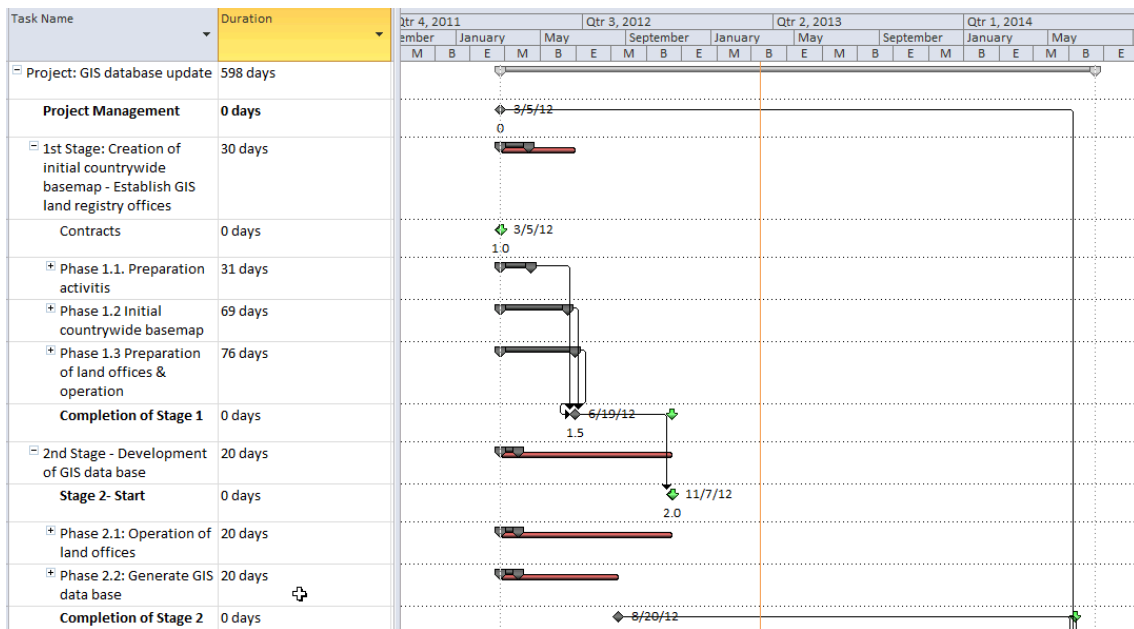
The final weights of the objectives as were given by the limit matrix were: project duration 0.23, cost 0.45 and RLI 0.37.

### 8.3 Solution scenarios

Based on the above preferences and input data a set of 9 solution scenarios were generated using the three proposed approaches: a) single objective optimisation of project duration, b) pareto-optimal solutions taking into consideration the project managers preferences over the objectives and c) classic pareto-optimal approach. In Table 8.1 the results got by each approach are shown. Due to the existence of deadlines on basic activities there are very limited differences on the project duration of the tasks related to the land offices receipt of statements. However, the other phases of the project can be executed in a variety of ways giving a total project duration that ranges from 1024 days which was the initial duration given by the MS Project tool to 589 days that is a high cost but very quick solution, that is the best schedule based on the time objective and it is shown in Figure 8.7.

**Table 8.1** Comparative results for GIS project

Algorithm	Makespan	RLI	Cost
Pareto	598	3.221	32.000
ANP	780	2.930	29.000
Single Obj.	598	3.221	32.000
Pareto	610	3.890	30.000
ANP	780	2.480	24.000
Single Obj.	600	3.870	28.700
Pareto	600	3.760	29.000
ANP	1000	2.100	15.000
Single Obj.	890	2.560	17.000



**Fig. 8.7** Best schedule found by pareto optimal approach - 598 days

Summarising, the preliminary design of an actual project for the development of large scale spatial data infrastructure for terrestrial areas network was presented to illustrate the proposed approach. Based on the proposed model and after interviewing the project manager the constraints, objectives and their weighting and degrees of freedom on the given constraints were decided. Having modelled the project and defined all the needed inputs, the proposed algorithm with three different settings: a) as a single objective, b) using the given weights for the requested objectives and c) looking for pareto-optimal schedules was executed. The best schedules got by each method, were presented to the project manager, in order to choose the best fitting schedule for the situation at hand.

## Chapter 9

# General Discussion & Conclusions

**Abstract** In this chapter, potential impact and significance of the conducted study, implications for researchers and practitioners and possible directions for further research on this subject, are discussed.

### 9.1 Summary of PhD Thesis contribution

Project scheduling involves the development of a project baseline schedule which specifies for each activity the precedence and resource feasible start and completion dates, the amounts of the various resource types that will be needed during each time period and as a result the corresponding budget required for the execution of the project.

Scheduling problems have been investigated since the late fifties, stimulated by the need to improve and facilitate project management. Project scheduling is a complex problem that every project manager faces in the beginning of each project and the consequences of an ill designed schedule can seriously endanger the successful project execution and completion. Applications can be found in diverse industries such as construction, software development, etc. In addition, project scheduling is very attractive for researchers, mainly those related to operational research, because the models in this area are rich and, hence, difficult to solve. In this Thesis, a new holistic conceptual and mathematical formulation integrating the standard RCPSP with its most common variations, namely preemption of activities, multiple modes, generalised precedence constraints with minimal and maximal lags and variable resource needs and availabilities, was also proposed. Based on this model an adaptive algorithm handling single objective and multi-objective cases either with prioritisation of the objectives or pareto optimality, was proposed. The moderator algorithm has as main role to manage the process and select the best auxiliary solution algorithm to be used based on the instance currently being solved. It was experimentally proven that the usage of the moderator algorithm raises the accuracy of the results without harming the execution time. Therefore we can have a reliable way for solving any scheduling problem having features spanning from the standard RCPSP to any of the above mentioned variations. This way project managers have a way of modelling their project in a single step and transparent process. Therefore, we have a unified model, that is reliable and accommodates the needs of project scheduling in practice, keeping at the same time great flexibility on what kind of solutions and at what degree each objective should be pursued. Summarising, this PhD Thesis has the following innovative parts:

- a holistic mathematic model integrating all the known extensions and variations of the resource constrained project scheduling problem, namely, preemption, multiple modes of execution, generalised precedence constraints, variable resources availabilities and requests over time and its binary formulation,
- an adaptive hybrid algorithm that handles the selection of the auxiliary algorithms that will be used for the solution of each problem's instance and the ways that the schedules will be generated (s-GS or p-SGS) both for single and multi-objective cases,
- achievement of good optimisation results as the proposed moderator algorithm always gives results at least as good as the best known methods and in some cases best values than those currently reported in the literature,
- a ready to use Add In for MS Project including the proposed algorithm, tools for setting up the project in a more flexible manner (i.e. adding details about the activities execution modes, points of preemption, resource types and time dependent availabilities, objectives and their priorities, etc.) and generating solution scenarios,
- takes into consideration systemic factors that affect project scheduling for the model generation and Add In development.

The next step on this research is to enhance it by adding a mechanism for automatically dealing with infeasibilities instead of interactively doing it. Further experimentation on the multi-objective side of the problem focusing on the comparison with the existing approaches is expected to give valuable insights.

## 9.2 Potential Impact and Significance

### 9.2.1 Implications for researchers

The proposed work consists of two parts, first a model that proposes a holistic view of the project scheduling problem and second, an adaptive solution algorithm to solve it more efficiently.

The holistic model is meant to cover the majority of cases encountered in practical situations and give a central reference point for the resource constrained project scheduling problem. This way researchers either trying to develop better exact or heuristics for the problem will not need to refer or re-apply to each facet of the problem but can directly work on all the facets.

Additionally, by using this holistic model the new solution methods will be ready to use and easily comparable one to another instead of checking each method against the different variations and often having very good methods for "research-wise" interesting facets and very few for those situations that are encountered more often in practice.

Furthermore, by generating of a conceptual and mathematical model of the problem having as a basis the resource constrained project scheduling problem and including all the deterministic extensions and variations of this problem, a new more practical approach of the project scheduling is proposed. Hopefully, this will lead to the development of solution methods that can be easily applied in practice and correspond to a realistic problem instead of having very good solutions for just a part of the problem.

In the end the idea is to have a generic model fitting more or less complex situations and leading to a joint research effort for the development of better solution methods that will actually cover the most significant aspects of the project scheduling problem.



On the other hand, the solution method it is also inspired by the holistic approach, as it does not provide single solutions with specific predefined objectives. Instead, it provides a set of tools that can be used to generate a mix of different solution scenarios and lets the project manager to make all the decisions. It is more of a support tool than a problem solver.

Furthermore, acknowledging the fact that a variety of good solution algorithms already exist but no way to choose the best for each instance being solved, the logic of the evolutionary algorithms is put on service to handle the algorithm selection. It is a simple but very effective way of handling a complex problem whose solution mechanics are not well defined yet. However, it would be interesting to gather usage data and create a knowledge data base containing the best fitted algorithms and try to extract patterns relating the problem instance characteristics to the optimal solution algorithm.

Additionally, the mixing of single and multi-objective approaches, with or without weights of the objectives gives a less mathematical but more practical look at how a problem should be solved. Maybe, having a good understanding of how to solve problems on well defined situations, the next step is to respond to the demand of solving less well-defined and without clear goals problems or even better propose solution scenarios without giving unique and inflexible solutions, as problems actually are more than the sum of inputs and outputs.

Summarising, the proposed work is a first step toward flexible models of complicated situations, adaptable solution methods and results aiming at supporting the decision makers without giving aphorisms about which solution is the best one and which is not.

### ***9.2.2 Implications for practitioners***

It is known that project scheduling is a multi-facet problem affected by a plethora of systemic parameters that cannot be easily taken into consideration in a quantitative model. This study aims at giving a tool to support the project manager on scheduling the project by providing a number of alternative solution scenarios to select the one that best fits the situation and not just a unique solution that can act as "one size fits all". It is expected that the inputs and objectives are set based on the current situation, taking into consideration all those uncountable parameters that are not part of the model but play a role on how the schedule should be. Even so, the provision of tools for ranking the objectives using quantitative and qualitative criteria (through an ANP model) and the ability to have mixed solution scenarios are valuable tools in handling the complexity of this problem.

In this Thesis, we proposed a solution process able to appropriately handle any situation from small and every day projects to large scale problems with complex activity relations, splittable activities, multiple execution modes, time windows and variability on the resource requirements and availability. And a solution algorithm capable of adapting itself and providing different ways of handling the specific instance of the problem at hand, based on:

- the number of objectives to be pursued,
- the type of the problem: simple, multi-mode, with generalised precedence constraints, minimal and maximal time lags, preemption or any mix of these features and
- the number and type of solution scenarios that are desired: from single objective optimisation of a range of objectives to multi-objective optimisation using weighted sum and/or pareto-optimal solutions, or a mix of the above,

The idea is to provide a model that works as a bunch of LEGO bricks, you can use it to model from the simplest problem to the most complicated without need of deep mathematical or other knowledge beside, the notions related to the project to be scheduled.

The practitioner gets to use a simple interface to set up the problem by defining the various inputs, like the tasks and their precedence relations, the different ways that some important (under the specific circumstances) tasks can be executed and which are the objectives and the role that should have in finding the optimal solutions. No decision is irreversible, on the contrary the project manager has a great amount of freedom to experiment on the effect of his/her changes on the final results and choose the solution that best fits the situation at hand. Summarising, the proposed work provides a unified approach of the project scheduling problems and provides the practitioner with an easy to use and flexible decision support tool, where the decision to take is how to schedule the project given specific circumstances. Moving a pass forward from existing partial approaches where the objective is to optimise the way a specific aspect of the problem is solved, we show that it is possible without affecting the quality of the results to have a holistic approach of the problem overcoming the raised complexity through the flexibility on the constraints and the generation of several good solutions that cover different needs and show off the trade offs between the selected objectives.

### 9.3 Future Work

Due to the complexity of the project scheduling problem and the difficulties encountered when solving even simple instances and much more when handling its various extensions and variations, even today, there is a lack of generic models that integrate all the different facets of a project that should be scheduled and provide a solution process. However, in practice projects often fail to fall precisely in one of the existing cases. For example a project can have some tasks that are splittable but not all and at the same time some activities with variable resource demands and a few tasks with hard deadlines. In such a case either some of the features should be omitted to fall in one of the existing problem types or a multi-phase approach, handling each situation separately, should be followed. Furthermore, there is the issue of whether a single or multi-objective approach can be followed, that even if it seems a straightforward decision, it is not, as managers are used to "what-if" scenarios and would prefer to have both options and evaluate the different results against the case at hand using just one or two objectives.

To fill in this research gap, a holistic model was proposed in this Thesis in order to provide a way to define all the desired characteristics, and provide a solution process that will generate project schedules adaptable to different project settings, organisational sizes and strategies and scalable according to the size and criticality of the undergoing project. Furthermore, a solution process that is simple and quick enough to permit immediate re-runs for the generation of alternative scenarios, is proposed to give the opportunity to the project manager (and/or the group of decision makers) responsible for the definition and final selection of the baseline schedule to have a satisfactory number of alternatives to discuss on and choose from.

The next step on this research is to enhance it by adding a mechanism for automatically dealing with infeasibilities instead of interactively doing it. Further experimentation on the multi-objective side of the problem focusing on the comparison with the existing approaches it is expected to give valuable insight. Beside these very concise next steps, there are a few short-term and long-term goals to be achieved. Starting from the specific proposed approach, a less deterministic approach could greatly enhance the quality of the given inputs as they could be more realistic, affecting the quality of the results.

Furthermore, more objectives and not only quantitative but also qualitative could be incorporated as to embrace the environmental parameters that affect project scheduling in practice. Although, qualitative objectives should be in some way, even approximatively, quantified, it would be a step toward the right direction.

On the other hand, there are a lot of technical issues that would be interesting to work at. For example, a multi-project approach to handle multiple project scheduling and gather data from the withstanding IT system about the other ongoing projects and those that are being planned to be executed on the same time span. An other interesting extension, would be to provide a model that handles both reactive and proactive scheduling as to support the project manager not only during scheduling but also when the schedule should be updated due to internal or external changes.

Summarising, the proposed approach is a first step on handling optimisation problems in a more practical and less mathematical manner without denying the need of quantification and specific solution generation. However, there are a lot of issues that should be handled and aspects of this and similar problems that should be examined before reaching a stable point where the models and the solution methods provided by the researcher's community match the practical problems as they are without compromises, assumptions and simplifications. This is the ultimate goal.



## References

- ISO 21500. Iso 21500:2012 project management. Technical report, International Organization for Standardization, 2012.
- B. Abbasi, S. Shadrokh, and J. Arkat. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. *Applied Mathematics and Computation*, 180(1):146–152, 2006.
- Russell Lincoln Ackoff, Shiv K Gupta, and J Sayer Minas. *Scientific method: Optimizing applied research decisions*, volume 41. Wiley New York, 1962.
- M. A. Al-Fawzan and M. Haouari. A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics*, 96(2):175–187, 2005.
- R. Alvarez-Valdes and J.M. Tamarit. *Heuristic Algorithm for Resource Constrained Project Scheduling: A Review and An Empirical Analysis*, pages 114–134. Elsevier, Amsterdam, 1989.
- Haldun Aytug, Mark A. Lawley, Kenneth McKay, Shantha Mohan, and Reha Uzsoy. Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research*, 161(1):86–110, 2005.
- Ray Ball, SP Kothari, and Charles E Wasley. Can we implement research on stock trading rules? *The Journal of Portfolio Management*, 21(2):54–63, 1995.
- F. Ballestin. A genetic algorithm for the resource renting problem with minimum and maximum time lags. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4446 LNCS:25–35, 2007.
- F. Ballestin, V. Valls, and S. Quintanilla. Pre-emption in resource-constrained project scheduling. *European Journal of Operational Research*, 189(3):1136–1152, 2008.
- Carlos A Bana e Costa, Leonardo Ensslin, Émerson C Cornêa, and Jean-Claude Vansnick. Decision support systems in action: integrated application in a multicriteria decision aid process. *European Journal of Operational Research*, 113(2):315–335, 1999.
- M. Bartusch, R. H. Mahrng, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16(1):199–240, 1988.
- Arthur Battram. *Navigating complexity: The essential guide to complexity theory in business and management*. Spiro Pr, 1998.
- Colin E. Bell and Kwangho Park. Solving resource-constrained project scheduling problems by a\* search. *Naval Research Logistics*, 37(1):61–84, 1990. Cited By (since 1996): 31 Export Date: 7 September 2011 Source: Scopus.
- L. Bianco, P. Dell’Olmo, and M. G. Speranza. Heuristics for multimode scheduling problems with dedicated resources. *European Journal of Operational Research*, 107(2):260–271, 1998.
- J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.
- F. F. Buctor. Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research*, 49(1):3–13, 1990.
- F. F. Buctor. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34(8):2335–2351, 1996. Cited By (since 1996): 50 Export Date: 5 October 2011 Source: Scopus.
- F. Bomsdorf and U. Derigs. A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. *OR Spectrum*, 30(4):751–772, 2008.
- K. Bouleimen and H. Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2):268–281, 2003.

- Peter Brucker. *Scheduling algorithms*. Springer, 2007.
- Peter Brucker, Andreas Drexl, Rolf Mahrng, Rolf Hhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999.
- K. M. Calhoun, R. F. Deckro, J. T. Moore, J. W. Chrissis, and J. C. Van Hove. Planning and re-planning in project and production scheduling. *Omega*, 30(3):155–170, 2002.
- C. C. B. Cavalcante, C. Carvalho De Souza, M. W. P. Savelsbergh, Y. Wang, and L. A. Wolsey. Scheduling projects with labor constraints. *Discrete Applied Mathematics*, 112(1-3):27–52, 2001.
- M. Cervantes, A. Lova, P. Tormos, and F. Barber. A dynamic population steady-state genetic algorithm for the resource-constrained project scheduling problem. volume 5027 LNAI, pages 611–620. 2008. Export Date: 7 October 2011 Source: Scopus.
- A. P. Chassiakos and S. P. Sakellariopoulos. Time-cost optimization of construction projects with generalized activity constraints. *Journal of Construction Engineering and Management*, 131(10):1115–1124, 2005.
- Peter Checkland. Soft systems methodology: a thirty year retrospective. *Systems Research and Behavioral Science*, 17: 11–58, 2000.
- Peter Checkland and Mark Winter. Process and content: two ways of using ssm. *Journal of the Operational Research Society*, 57(12):1435–1441, 2005.
- W. N. Chen, J. Zhang, H. S. H. Chung, R. Z. Huang, and O. Liu. Optimizing discounted cash flows in project scheduling—an ant colony optimization approach. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 40(1):64–77, 2010. Cited By (since 1996): 5 Export Date: 7 October 2011 Source: Scopus Art. No.: 5196736.
- N. Christofides, R. Alvarez-Valdes, and J. M. Tamarit. Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, 29(3):262–273, 1987.
- C West Churchman. The x of x\*+. *Management Science*, 9(6):351–357, 1963.
- Dale F Cooper. Heuristics for scheduling resource-constrained projects: An experimental investigation. *Management Science*, 22(11):1186–1194, 1976.
- Kenneth G Cooper. The rework cycle: benchmarks for the project manager. *Project Management Journal*, 24(1):17–21, 1993.
- Kerry Costello, Lynn Crawford, Lesley Bentley, and Julien Pollack. Connecting soft systems thinking with project management practice: An organizational change casestudy. *Systems Theory and Practice in the Knowledge Age*, pages 47–54, 2002.
- J. Damay, A. Quilliot, and E. Sanlaville. Linear programming based algorithms for preemptive and non-preemptive rcpsp. *European Journal of Operational Research*, 182(3):1012–1022, 2007.
- David W Daniel. Hard problems in a soft world. *International Journal of Project Management*, 8(2):79–83, 1990.
- Thomas H Davenport, Jeanne G Harris, and Susan Cantrell. Enterprise systems and ongoing process change. *Business Process Management Journal*, 10(1):16–26, 2004.
- Edward W. Davis. Project scheduling under resource constraints - historical review and categorization of procedures. *AIIE Trans*, 5(4):297–313, 1973. Cited By (since 1996): 55 Export Date: 7 September 2011 Source: Scopus.
- Edward W. Davis and James H. Patterson. Comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science*, 21(8):944–955, 1975. Cited By (since 1996): 150 Export Date: 5 October 2011 Source: Scopus.
- Bert De Reyck and Willy Herroelen. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119(2):538–556, 1999.
- K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.
- Dieter Debels and Mario Vanhoucke. A bi-population based genetic algorithm for the resource-constrained project scheduling problem. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganá, HeowPueh Lee, Youngsong Mun, David Taniar, and ChihJengKenneth Tan, editors, *Computational Science and Its Applications – ICCSA 2005*, volume 3483 of *Lecture Notes in Computer Science*, pages 378–387. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25863-6.
- E. Demeulemeester and W. Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12):1803–1818, 1992.
- E. Demeulemeester, W. Herroelen, W. P. Simpson, S. Baroum, J. H. Patterson, and K. K. Yang. On a paper by christofides et al. for solving the multiple-resource constrained, single project scheduling problem. *European Journal of Operational Research*, 76(1):218–228, 1994. Cited By (since 1996): 11 Export Date: 7 September 2011 Source: Scopus.
- E. Demeulemeester, B. De Reyck, B. Foubert, W. Herroelen, and M. Vanhoucke. New computational results on the discrete time/cost trade-off problem in project networks. *Journal of the Operational Research Society*, 49(11): 1153–1163, 1998.

- E. L. Demeulemeester and W. S. Herroelen. An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem. *European Journal of Operational Research*, 90(2):334–348, 1996.
- E. L. Demeulemeester and W. S. Herroelen. A branch-and-bound procedure for the generalized resource-constrained project scheduling problem. *Operations Research*, 45(2):201–212, 1997. Cited By (since 1996): 23 Export Date: 7 September 2011 Source: Scopus.
- Ebru Demirkol, Sanjay Mehta, and Reha Uzsoy. Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109(1):137–141, 1998.
- L. Deng, Y. Lin, W. Zheng, and Y. Xi. Incorporating justification in the particle swarm optimization for the rcpsp. *International Journal of Innovative Computing, Information and Control*, 4(9):2315–2324, 2008. Cited By (since 1996): 4 Export Date: 7 October 2011 Source: Scopus.
- Robert H. Doersch and James H. Patterson. Scheduling a project to maximize its present value: A zero-one programming approach. *Management Science*, 23(8):882–889, 1977. Cited By (since 1996): 49 Export Date: 7 September 2011 Source: Scopus.
- U. Dorndorf, E. Pesch, and T. Phan-Huy. Time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalized precedence constraints. *Management Science*, 46(10):1365–1384, 2000.
- Ulrich Dorndorf and Erwin Pesch. Evolution based learning in a job shop scheduling environment. *Computers & Operations Research*, 22(1):25–40, 1995.
- A. Drexl and A. Kimms. Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society*, 52(3):340–351, 2001.
- L. E. Drezet and J. C. Billaut. A project scheduling problem with labour constraints and time-dependent activities requirements. *International Journal of Production Economics*, 112(1):217–225, 2008.
- Deborah Duarte, Andrea Lewis, Edward J Hoffman, and Dale Crossman. A career development model for project management workforces. *Journal of Career Development*, 22(2):149–164, 1995.
- S. E. Elmaghraby. An algebra for the analysis of generalized activity networks. *Management Science*, 10(3):494–514, 1964.
- S.E. Elmaghraby. *Activity networks: project planning and control by network models*. Wiley-Interscience publication. Wiley, 1977. ISBN 9780471238614.
- S. S. Erenguc, T. Ahn, and D. G. Conway. Resource constrained project scheduling problem with multiple crashable modes: an exact solution method. *Naval Research Logistics*, 48(2):107–127, 2001.
- Jay W Forrester. Industrial dynamics after the first decade. *Management Science*, 14(7):398–415, 1968.
- Jay W Forrester. Principles of systems, 2nd preliminary edition, 1980.
- Jay Wright Forrester. *Industrial dynamics*, volume 2. MIT press Cambridge, MA, 1961.
- B. Franck, K. Neumann, and C. Schwindt. Project scheduling with calendars. *OR Spektrum*, 23(3):325–334, 2001.
- Delbert R Fulkerson. A network flow computation for project cost curves. *Management Science*, 7(2):167–178, 1961.
- Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman New York, 1979.
- Eliyahu M Goldratt. *Critical chain*. Gower Publishing Company, Limited, 1997.
- R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimisation and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:236–287, 1979.
- M. Hapke, A. Jaskiewicz, and R. Slowinski. Interactive analysis of multiple-criteria project scheduling problems. *European Journal of Operational Research*, 107(2):315–324, 1998.
- S. Hartmann. Project scheduling under limited resources: Models, methods and applications. *Project Scheduling under Limited Resources*, 1999.
- S. Hartmann. Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, 102(1-4): 111–135, 2001. Cited By (since 1996): 69 Export Date: 9 January 2012 Source: Scopus.
- S. Hartmann. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, 49(5):433–448, 2002.
- S. Hartmann. Project scheduling with resource capacities and requests varying with time: A case study. *Flexible Services and Manufacturing Journal*, 25(1-2):74–93, 2013. Export Date: 14 March 2013 Source: Scopus.
- S. Hartmann and A. Drexl. Project scheduling with multiple modes: A comparison of exact algorithms. *Networks*, 32(4):283–297, 1998. Cited By (since 1996): 35 Export Date: 9 January 2012 Source: Scopus.
- S. Hartmann and R. Kolisch. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2):394–407, 2000.
- S. Anke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750, 1998.
- S. Anke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.
- W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306, 2005.

- W. Herroelen, B. De Reyck, and E. Demeulemeester. Resource-constrained project scheduling: A survey of recent developments. *Computers and Operations Research*, 25(4):279–302, 1998.
- W. S. Herroelen, P. Van Dommelen, and E. L. Demeulemeester. Project network models with discounted cash flows a guided tour through recent developments. *European Journal of Operational Research*, 100(1):97–121, 1997.
- Willy Herroelen and Roel Leus. The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3):550–565, 2004.
- Willy Herroelen, Erik Demeulemeester, and Bert De Reyck. *A classification scheme for project scheduling*. Springer, 1999.
- Willy S. Herroelen and Erik L. Demeulemeester. Project management and scheduling. *European Journal of Operational Research*, 90(2):197–199, 1996.
- John H Holland. Adaption in natural and artificial systems. 1975.
- O. Icmeli and S. S. Erenguc. A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows. *Computers and Operations Research*, 21(8):841–853, 1994.
- O. Icmeli and S. S. Erenguc. A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows. *Management Science*, 42(10):1395–1408, 1996a.
- O. Icmeli and S. S. Erenguc. The resource constrained time/cost tradeoff project scheduling problem with discounted cash flows. *Journal of Operations Management*, 14(3):255–275, 1996b.
- Oya Icmeli and S Selcuk Erenguc. A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows. *Management Science*, 42(10):1395–1408, 1996c.
- B. Jarboui, N. Damak, P. Siarry, and A. Rebai. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195(1):299–308, 2008.
- R. Jin, W. Chen, and T. W. Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, 23(1):1–13, 2001.
- L. Kaplan. Resource-constrained project scheduling with preemption of jobs. Unpublished Phd Dissertation University of Michigan (1988), 1988.
- Honoring Dr Richard M Karp. On the computational complexity. *Networks*, 5:45–68, 1975.
- JB Kelley Jr and Pa Fort Washington. Cpm present and future. in *New Horizons in Industrial Engineering*, Spartan Books, Inc., Baltimore, Maryland, 1963.
- JAGM Kerbosh and HJ Schell. Network planning by the extended metra potential method. *Report KS-1.1, University of Technology, Eindhoven, Department of Industrial Engineering*, 1975.
- J. L. Kim and R. D. Ellis. Comparing schedule generation schemes in resource-constrained project scheduling using elitist genetic algorithm. *Journal of Construction Engineering and Management*, 136(2):160–169, 2010.
- K. W. Kim, M. Gen, and G. Yamazaki. Hybrid genetic algorithm with fuzzy logic for resource-constrained project scheduling. *Applied Soft Computing Journal*, 2(3):174–188, 2003. Cited By (since 1996): 24 Export Date: 7 October 2011 Source: Scopus.
- A. Kimms. Maximizing the net present value of a project under resource constraints using a lagrangian relaxation based heuristic with tight upper bounds. *Annals of Operations Research*, 102(1-4):221–236, 2001.
- R. Klein. Project scheduling with time-varying resource constraints. *International Journal of Production Research*, 38(16):3937–3952, 2000.
- R. Klein and A. Scholl. Progress: Optimally solving the generalized resource-constrained project scheduling problem. *Mathematical Methods of Operations Research*, 52(3):467–488, 2000.
- P. Kobylanski and D. Kuchta. A note on the paper by m. a. al-fawzan and m. haouari about a bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics*, 107(2):496–501, 2007.
- R. Kolisch. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14(3):179–192, 1996.
- R. Kolisch. Integrated scheduling, assembly area- and part-assignment for large-scale, make-to-order assemblies. *International Journal of Production Economics*, 64(1):127–141, 2000.
- R. Kolisch and A. Drexel. Local for multi-mode resource-constrained project. *IIE Transactions (Institute of Industrial Engineers)*, 29(11):987–999, 1997.
- R. Kolisch and S. Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 2006. Cited By (since 1996): 196 Export Date: 14 March 2013 Source: Scopus.
- R. Kolisch and R. Padman. An integrated survey of deterministic project scheduling. *Omega*, 29(3):249–272, 2001.
- R. Kolisch and A. Sprecher. Psplib - a project scheduling problem library. *European Journal of Operational Research*, 96(1):205–216, 1997. Cited By (since 1996): 244 Export Date: 13 January 2012 Source: Scopus.
- R. Kolisch, A. Sprecher, and A. Drexel. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41(10):1693–1703, 1995.



- Rainer Kolisch and Sönke Hartmann. Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In *Project Scheduling*, pages 147–178. Springer US, 1999.
- Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.
- V. J. Leon and R. Balakrishnan. Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *OR Spektrum*, 17(2-3):173–182, 1995. Cited By (since 1996): 19 Export Date: 7 September 2011 Source: Scopus.
- James P Lewis. *Mastering project management: Applying advanced concepts of systems thinking, control and evaluation, resource allocation*. McGraw-Hill, 1998.
- Ming Li, Yuanbiao Zhang, Weigang Jiang, and Jianwen Xie. A particle swarm optimization algorithm with crossover for resource constrained project scheduling problem. In *Proceedings of the 2009 IITA International Conference on Services Science, Management and Engineering, SSME '09*, pages 69–72, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3729-0.
- Ting-Peng Liang, Jason Chia-Hsien Wu, James J. Jiang, and Gary Klein. The impact of value diversity on information system development projects. *International Journal of Project Management*, 30(6):731 – 739, 2012. ISSN 0263-7863. [ce:title;European Academy of Management \(EURAM 2011\) Conference;ce:title;.](#)
- Xiaoxiang Liu, Weigang Jiang, Jianwen Xie, and Yitian Jia. A new resource constrained project scheduling problem. In *Information Processing, 2009. APCIP 2009. Asia-Pacific Conference on*, volume 1, pages 476–480, 2009.
- L. L. Lorenzoni, H. Ahonen, and A. G. d Alvarenga. A multi-mode resource-constrained scheduling problem in the context of port operations. *Computers and Industrial Engineering*, 50(1-2):55–65, 2006.
- Rolf H Mahrting. Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Operations Research*, 32(1):89–120, 1984.
- V. Maniezzo and A. Mingozzi. Project scheduling problem with irregular starting time costs. *Operations Research Letters*, 25(4):175–182, 1999.
- Mike McMaster. Foresight: Exploring the structure of the future. *Long Range Planning*, 29(2):149–155, 1996.
- J. J. M. Mendes, J. F. Gonçalves, and M. G. C. Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers and Operations Research*, 36(1):92–109, 2009. Cited By (since 1996): 29 Export Date: 7 October 2011 Source: Scopus.
- Louis Mesnard and Erik Dietzenbacher. On the interpretation of fixed input coefficients under aggregation\*. *Journal of Regional Science*, 35(2):233–243, 1995.
- M. Mika, G. Waligora, and J. Weglarz. Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. *European Journal of Operational Research*, 164(3 SPEC. ISS.):639–668, 2005.
- A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44(5):714–729, 1998. Cited By (since 1996): 97 Export Date: 5 September 2011 Source: Scopus.
- J. R. Montoya-Torres, E. Gutierrez-Franco, and C. PirachicÁ!n-Mayorga. Project scheduling with limited resources using a genetic algorithm. *International Journal of Project Management*, 28(6):619–628, 2010. Export Date: 7 October 2011 Source: Scopus.
- E Muoneke. On the stochastic powers of nonnegative reducible matrices. *LINEAR ALGEBRA APPLIC.*, 90:57–63, 1987.
- Jaroslav Nabrzyski and Jan Weglarz. Knowledge-based multiobjective project scheduling problems. In Jan Weglarz, editor, *Project Scheduling*, volume 14 of *International Series in Operations Research & Management Science*, pages 383–411. Springer US, 1999. ISBN 978-1-4613-7529-6.
- K. Neumann and M. Morlock. *Operations Research*. Hanser, 1993.
- K. Neumann and C. Schwindt. Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, 56(3):513–533, 2002.
- K. Neumann and J. Zimmermann. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research*, 127(2):425–443, 2000.
- K. Neumann and J. Zimmermann. Exact and truncated branch-and-bound procedures for resourceconstrained project scheduling with discounted cash flows and general temporal constraints. *Cent Eur J Oper Res*, 10:357–380, 2002.
- K. Neumann, C. Schwindt, and J. Zimmermann. Recent results on resource-constrained project scheduling with time windows: Model, solution methods, and applications. *Central European Journal of Operations Research*, 10(1): 113–148, 2002.
- K. Neumann, C. Schwindt, and J. Zimmermann. *Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions*. Springer, 2003. ISBN 9783540401254.
- Richard D. Noble. Mathematical modelling in the context of problem solving. *Mathematical Modelling*, 3(3):215–219, 1982.

- K. Nonobe and T. Ibaraki. Formulation and tabu search algorithm for the resource constrained project scheduling problem. *Essays and Surveys in Metaheuristics*, pages 557–588, 2002a.
- Koji Nonobe and Toshihide Ibaraki. *Formulation and tabu search algorithm for the resource constrained project scheduling problem*, pages 557–588. Springer, 2002b.
- Semih Onut, Umut R. Tuzkaya, and ErÅşin Torun. Selecting container port via a fuzzy anp-based approach: A case study in the marmara region, turkey. *Transport Policy*, 18(1):182–193, 2011.
- Linet Ozdamar and Gunduz Ulusoy. Survey on the resource-constrained project scheduling problem. *IIE Transactions (Institute of Industrial Engineers)*, 27(5):574–586, 1995.
- James H. Patterson and Walter D. Huber. Horizon-varying, zero-one approach to project scheduling. *Management Science*, 20(6):990–998, 1974.
- JH Patterson, R Slowinski, FB Talbot, and J Weglarz. An algorithm for a general class of precedence and resource constrained scheduling problems. *Advances in project scheduling*, (Part I):3–28, 1989.
- Vincent Van Peteghem and Mario Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2):409–418, 2010.
- E Pinson, C Prins, and F Rullier. Using tabu search for solving the resource-constrained project scheduling problem. In *Proceedings of the 4th international workshop on project management and scheduling*, pages 102–106, 1994.
- PMI. A guide to the project management body of knowledge: Pmbok guide. Project Management Institute, 2012.
- Alan Pritsker, Lawrence Watters, and P. M. Wolfe. Multiproject scheduling with limited resources—a zero-one programming approach. *Mgmt Science*, 16(1):93–108, 1969.
- S. Proon and M. Jin. A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. *Naval Research Logistics*, 58(2):73–82, 2011.
- Alexandre Rodrigues and John Bowers. The role of system dynamics in project management. *International Journal of Project Management*, 14(4):213–220, 1996.
- E. Rokou and K. Kirytopoulos. Project resource leveling and robustness optimization using anp, 8 - 11 July 2012.
- J Rosenhead and J Mingers. Rational analysis for a problematic world revisited: Problem structuring methods for complexity. *Uncertainty and Conflict: Wiley and Sons*, 2001.
- Bernard Roy. *Multicriteria methodology for decision aiding*, volume 12. Springer, 1996.
- Bernard Roy and Daniel Vanderpooten. An overview on the european school of mcda: Emergence, basic features and current worksâ? *European Journal of Operational Research*, 99(1):26–27, 1997.
- T. L. Saaty. *Decision making with dependence and feedback: the analytic network process*. RWS Publications, Pittsburgh, 1996.
- T. L. Saaty and M. Sagir. An essay on rank preservation and reversal. *Mathematical and Computer Modelling*, 49(5-6): 1230–1243, 2009. Cited By (since 1996): 3 Export Date: 14 March 2013 Source: Scopus.
- SE Sampson and EN Weiss. Local search techniques for the generalized resource constrained project scheduling problem. *Naval Res Logist*, 40:665–675, 1993.
- M. Sevaux and S. Dauzere-Peres. Genetic algorithms to minimize the weighted number of late jobs on a single machine. *European Journal of Operational Research*, 151(2):296–306, 2003. Cited By (since 1996): 39 Export Date: 10 January 2012 Source: Scopus.
- David B Shmoys and Eva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(1-3):461–474, 1993.
- R. Slowinski. Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research*, 7(3):265–273, 1981.
- Jerome Spanier. Thoughts about the essentials of mathematical modelling. *Mathematical Modelling*, 1(1):99–108, 1980.
- A. Sprecher. *Resource-constrained project scheduling: exact methods for the multi-mode case*. Lecture notes in economics and mathematical systems. Springer-Verlag, 1994. ISBN 9783540578345.
- A. Sprecher, S. Hartmann, and A. Drexl. An exact algorithm for project scheduling with multiple modes. *OR Spectrum*, 19(3):195–203, 1997. Cited By (since 1996): 52 Export Date: 9 January 2012 Source: Scopus.
- International Organization for Standardization. *ISO 8402: 1994: Quality Management and Quality Assurance-Vocabulary*. International Organization for Standardization, 1994.
- Joel P. Stinson, Edward W. Davis, and Basheer M. Khumawala. Multiple resource-constrained scheduling using branch and bound. *AIEE Trans*, 10(3):252–259, 1978.
- F. Brian Talbot. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *MAN-AGE SCI*, V 28(N 10):1197–1210, 1982. Cited By (since 1996): 121 Export Date: 7 September 2011 Source: Scopus.
- F. Brian Talbot and James H. Patterson. Efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science*, 24(11):1163–1174, 1978.

- Chunqiao Tan. A multi-criteria interval-valued intuitionistic fuzzy group decision making with choquet integral-based topsis. *Expert Systems with Applications*, In Press, Corrected Proof, 2013. group decision making expanding TOPSIS by new fuzzy operation for geom.mean calculation.
- L. V. Tavares. A review of the contribution of operational research to project management. *European Journal of Operational Research*, 136(1):1–18, 2002.
- P. R. Thomas and S. Salhi. A tabu search approach for the resource constrained project scheduling problem. *Journal of Heuristics*, 4(2):123–139, 1998. Cited By (since 1996): 39 Export Date: 21 February 2012 Source: Scopus.
- L. Y. Tseng and S. C. Chen. A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 175(2):707–721, 2006. Cited By (since 1996): 31 Export Date: 7 October 2011 Source: Scopus.
- G. Ulusoy and L. Ozdamar. Constraint-based perspective in resource constrained project scheduling. *International Journal of Production Research*, 32(3):693–705, 1994.
- M. Vanhoucke, E. Demeulemeester, and W. Herroelen. Maximizing the net present value of a project with linear time-dependent cash flows. *International Journal of Production Research*, 39(14):3159–3181, 2001.
- M. Vanhoucke, E. Demeulemeester, and W. Herroelen. Discrete time/cost trade-offs in project scheduling with time-switch constraints. *Journal of the Operational Research Society*, 53(7):741–751, 2002.
- M. Vanhoucke, E. Demeulemeester, and W. Herroelen. Progress payments in project scheduling problems. *European Journal of Operational Research*, 148(3):604–620, 2003.
- M. Vanhoucke, J. Coelho, D. Debels, B. Maenhout, and L. V. Tavares. An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, 187(2): 511–524, 2008.
- V. A. Varma, R. Uzsoy, J. Pekny, and G. Blau. Lagrangian heuristics for scheduling new product development projects in the pharmaceutical industry. *Journal of Heuristics*, 13(5):403–433, 2007.
- Ana Viana and Jorge Pinho de Sousa. Using metaheuristics in multiobjective resource constrained project scheduling. *European Journal of Operational Research*, 120(2):359–374, 2000.
- Guilherme Ernani Vieira, Fabio Favaretto, and Paulo Cesar Ribas. Comparing genetic algorithms and simulated annealing in master production scheduling problems. In *Proceeding of 17th International Conference on Production Research, Blacksburg, Virginia, USA*, 2003.
- H. Wang, T. Li, and D. Lin. Efficient genetic algorithm for resource-constrained project scheduling problem. *Transactions of Tianjin University*, 16(5):376–382, 2010.
- J. Weglarz. On certain models of resource allocation problems. *Kybernetes*, 9(1):61–66, 1980.
- Charles Raymond White. *An algorithm for finding optimal or near optimal solutions to the production scheduling problem*. PhD thesis, Purdue University, 1963.
- Margaret M Wiecek, Matthias Ehrgott, Georges Fadel, and Jose Rui Figueira. Multiple criteria decision making for engineering. *Omega*, 36(3):337–339, 2008.
- Jerome D Wiest. A heuristic model for scheduling large projects with limited resources. *Management Science*, 13(6): B–359, 1967.
- M Winter. Problem structuring in project management: an application of soft systems methodology (ssm). *Journal of the Operational Research Society*, 57(7):802–812, 2006.
- Eric F Wolstenholme. Qualitative vs quantitative modelling: the evolving balance. *Journal of the Operational Research Society*, pages 422–428, 1999.
- S. Xie, B. Bao, and J. Chen. Genetic algorithm based on activities resource competition relation for the rcpsp. volume 6377 LNCS, pages 350–356. 2010.
- Cheng Xu and Wu Cheng. Hybrid algorithm for project scheduling with capacity constraint\*. *Journal of Systems Engineering and Electronics*, 19(5):1041–1046, 2008.
- H. Zhang, X. Li, H. Li, and F. Huang. Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction*, 14(3):393–404, 2005. Cited By (since 1996): 56 Export Date: 21 February 2012 Source: Scopus.
- H. Zhang, H. Li, and C. M. Tam. Permutation-based particle swarm optimization for resource-constrained project scheduling. *Journal of Computing in Civil Engineering*, 20(2):141–149, 2006. Cited By (since 1996): 10 Export Date: 7 October 2011 Source: Scopus.



## **Appendix A**

### **Experimental Results**

#### **A.1 Exerpt of analytical results RCPSP**

For each instance 100 repetitions of the experiment were effectuated to get the average values. The implementation of the algorithm was made in C#.NET programming language. The experiments were executed using a computer with the following characteristics: Intel(R) Core(TM) 2 Duo CPU P8600 at 2.40 GHz and RAM 8.00 GB.

Filename	Min Dur	Max Dur	Optimal	Aver.Dev. Optimum	Frequency of Opt.	Jobs Number	Res Number	Res Strength 1	Res Strength 2	Res Strength 3	Res Strength 4	Resource Factor	Network Completed
J301_1RCP	43	45	43	0.0%	70%	30	4	0.18	0.20	0.00	0.21	0.25	1.50
J301_2RCP	47	51	47	0.0%	96%	30	4	0.22	0.20	0.14	0.22	0.25	1.50
J301_3RCP	47	47	47	0.0%	100%	30	4	0.33	0.00	0.19	0.22	0.25	1.50
J303_1RCP	72	72	72	0.0%	100%	30	4	0.75	0.67	0.71	0.67	0.25	1.50
J303_2RCP	40	40	40	0.0%	100%	30	4	0.75	0.75	0.67	0.67	0.25	1.50
J303_3RCP	57	57	57	0.0%	100%	30	4	0.71	0.67	0.75	0.73	0.25	1.50
J303_4RCP	98	98	98	0.0%	100%	30	4	1.00	0.67	0.00	0.00	0.25	1.50
J303_5RCP	53	53	53	0.0%	100%	30	4	0.67	0.80	0.75	0.71	0.25	1.50
J303_6RCP	54	54	54	0.0%	100%	30	4	0.71	0.75	0.67	0.73	0.25	1.50
J303_7RCP	48	48	48	0.0%	100%	30	4	0.00	0.80	0.73	0.70	0.25	1.50
J303_8RCP	54	54	54	0.0%	100%	30	4	0.70	1.00	0.67	0.67	0.25	1.50
J303_9RCP	65	65	65	0.0%	100%	30	4	0.75	0.75	0.67	0.75	0.25	1.50
J303_10RCP	59	59	59	0.0%	100%	30	4	0.67	0.50	0.75	0.69	0.25	1.50
J304_1RCP	49	49	49	0.0%	100%	30	4	0.00	1.00	1.00	1.00	0.25	1.50
J304_2RCP	60	60	60	0.0%	100%	30	4	1.00	1.00	1.00	1.00	0.25	1.50
J304_3RCP	47	47	47	0.0%	100%	30	4	1.00	1.00	1.00	1.00	0.25	1.50
J304_4RCP	57	57	57	0.0%	100%	30	4	1.00	1.00	1.00	1.00	0.25	1.50
J304_5RCP	59	59	59	0.0%	100%	30	4	1.00	1.00	1.00	1.00	0.25	1.50
J304_6RCP	45	45	45	0.0%	100%	30	4	1.00	0.00	1.00	1.00	0.25	1.50
J304_8RCP	55	55	55	0.0%	100%	30	4	1.00	1.00	1.00	1.00	0.25	1.50
J304_9RCP	38	38	38	0.0%	100%	30	4	1.00	1.00	1.00	1.00	0.25	1.50
J304_10RCP	48	48	48	0.0%	100%	30	4	1.00	1.00	1.00	1.00	0.25	1.50
J3017_2RCP	68	68	68	0.0%	100%	30	4	0.17	0.14	0.00	0.20	0.25	1.50
J3017_3RCP	60	60	60	0.0%	100%	30	4	0.00	0.18	0.00	0.20	0.25	1.50
J3017_4RCP	49	49	49	0.0%	100%	30	4	0.20	0.22	0.00	0.00	0.25	1.50
J3017_5RCP	47	48	47	0.0%	96%	30	4	0.14	0.17	0.17	0.22	0.25	1.50
J3017_6RCP	63	63	63	0.0%	100%	30	4	0.19	0.20	0.17	0.33	0.25	1.50
J3017_7RCP	57	57	57	0.0%	100%	30	4	0.00	0.18	0.17	0.20	0.25	1.50
J3017_8RCP	61	61	61	0.0%	100%	30	4	0.21	0.18	0.20	0.14	0.25	1.50
J3017_9RCP	48	48	48	0.0%	100%	30	4	0.21	0.17	0.23	0.00	0.25	1.50
J3017_10RCP	66	66	66	0.0%	100%	30	4	0.17	0.22	0.20	0.00	0.25	1.50
J3018_1RCP	53	53	53	0.0%	100%	30	4	0.50	0.56	0.00	0.67	0.25	1.50
J3018_2RCP	55	55	55	0.0%	100%	30	4	0.50	0.50	0.50	0.50	0.25	1.50
J3018_3RCP	56	56	56	0.0%	100%	30	4	0.50	0.50	0.50	1.00	0.25	1.50
J3018_4RCP	70	70	70	0.0%	100%	30	4	0.50	0.57	0.50	0.50	0.25	1.50

Fig. A.1 Single objective execution of j30 instances - part I

Filename	Min Dur	Max Dur	Optimal	Aver.Dev. Optimum	Frequency of Opt	Jobs Number	Res Number	Res Strength	Res Streng	Res Strength 3	Res Strength 4	Resource Factor	Network Complexity
J0418_7.RCP	48	48	48	0.0%	100%	30	4	0.50	0.50	0.67	0.50	0.25	1.81
J0418_8.RCP	52	52	52	0.0%	100%	30	4	0.50	1.00	0.50	0.50	0.50	1.81
J0418_9.RCP	47	47	47	0.0%	100%	30	4	0.60	0.67	0.00	0.56	0.25	1.81
J0418_10.RCP	49	49	49	0.0%	100%	30	4	0.00	0.50	0.56	0.57	0.25	1.81
J0419_1.RCP	40	40	40	0.0%	100%	30	4	0.75	0.71	0.71	0.75	0.25	1.81
J0419_2.RCP	58	58	58	0.0%	100%	30	4	0.50	0.00	1.00	0.69	0.25	1.81
J0419_3.RCP	83	83	83	0.0%	100%	30	4	0.75	0.00	0.67	0.50	0.25	1.81
J0419_4.RCP	39	39	39	0.0%	100%	30	4	0.67	0.50	0.70	0.67	0.25	1.81
J0419_5.RCP	48	48	48	0.0%	100%	30	4	0.80	0.00	0.67	0.73	0.25	1.81
J0419_6.RCP	49	49	49	0.0%	100%	30	4	0.80	0.67	0.70	0.73	0.25	1.81
J0419_7.RCP	57	57	57	0.0%	100%	30	4	0.75	0.75	0.80	0.67	0.25	1.81
J0419_8.RCP	55	55	55	0.0%	100%	30	4	0.71	0.80	0.00	0.67	0.25	1.81
J0419_9.RCP	38	38	38	0.0%	100%	30	4	0.75	0.73	0.75	0.70	0.25	1.81
J0419_10.RCP	47	47	47	0.0%	100%	30	4	0.75	0.73	0.50	0.67	0.25	1.81
J0431_1.RCP	65	65	65	0.0%	100%	30	4	0.00	0.00	0.23	0.14	0.25	2.13
J0431_2.RCP	60	60	60	0.0%	100%	30	4	0.25	0.18	0.00	0.14	0.25	2.13
J0431_3.RCP	55	56	55	0.0%	96%	30	4	0.18	0.18	0.14	0.33	0.25	2.13
J0431_4.RCP	77	78	77	0.0%	80%	30	4	0.14	0.20	0.17	0.25	0.25	2.13
J0431_5.RCP	53	53	53	0.0%	100%	30	4	0.20	0.33	0.14	0.00	0.25	2.13
J0431_6.RCP	59	59	59	0.0%	100%	30	4	0.25	0.14	0.22	0.14	0.25	2.13
J0431_7.RCP	58	58	58	0.0%	100%	30	4	0.25	0.17	0.00	0.00	0.25	2.13
J0431_8.RCP	61	61	61	0.0%	100%	30	4	0.25	0.00	0.25	0.17	0.25	2.13
J0431_10.RCP	53	53	53	0.0%	100%	30	4	0.00	0.33	0.00	0.20	0.25	2.13
J0434_1.RCP	68	68	68	0.0%	100%	30	4	0.50	0.56	0.50	0.57	0.25	2.13
J0434_2.RCP	44	44	44	0.0%	100%	30	4	0.00	0.00	0.60	0.50	0.25	2.13
J0434_3.RCP	69	69	69	0.0%	100%	30	4	0.00	0.00	0.00	0.00	0.25	2.13
J0434_4.RCP	67	67	67	0.0%	100%	30	4	0.57	0.50	0.00	0.50	0.25	2.13
J0434_5.RCP	63	63	63	0.0%	100%	30	4	0.50	0.57	0.50	1.00	0.25	2.13
J0434_6.RCP	52	52	52	0.0%	100%	30	4	0.67	0.56	0.50	0.50	0.25	2.13
J0434_7.RCP	58	58	58	0.0%	100%	30	4	0.56	0.50	0.50	0.56	0.25	2.13
J0434_8.RCP	58	58	58	0.0%	100%	30	4	0.60	0.00	0.50	0.56	0.25	2.13
J0434_9.RCP	60	60	60	0.0%	100%	30	4	0.67	0.50	0.00	0.50	0.25	2.13
J0434_10.RCP	47	47	47	0.0%	100%	30	4	1.00	0.60	0.55	0.50	0.25	2.13
J0435_1.RCP	57	57	57	0.0%	100%	30	4	0.70	0.68	0.75	0.00	0.25	2.13
J0435_2.RCP	53	53	53	0.0%	100%	30	4	0.80	0.67	0.80	0.75	0.25	2.13
J0435_3.RCP	60	60	60	0.0%	100%	30	4	0.71	0.75	0.80	0.00	0.25	2.13

Fig. A.2 Single objective execution of j30 instances - part II

Filename	Min Dur	Max Dur	Optimal	Aver.Dev. Optimum	Frequency of Opt	Jobs Number	Res Number	Res Strength	Res Strength	Res Strength	Res Strength	Res Strength	Res Strength	Res Strength	Resource Factor	Network Complexity
J0035_6.RCP	58	58	58	0.0%	100%	30	4	0.67	0.00	0.00	0.67	0.71	0.71	0.71	0.25	2.13
J0035_7.RCP	61	63	61	0.0%	100%	30	4	0.67	0.75	0.75	0.75	0.75	0.75	0.75	0.25	2.13
J0035_8.RCP	63	63	63	0.0%	100%	30	4	0.71	0.50	0.50	0.67	0.71	0.71	0.71	0.25	2.13
J0035_9.RCP	59	59	59	0.0%	100%	30	4	0.00	0.00	0.00	0.50	0.80	0.80	0.80	0.25	2.13
J0035_10.RCP	59	59	59	0.0%	100%	30	4	0.75	1.00	1.00	0.75	0.80	0.80	0.80	0.25	2.13
J0036_1.RCP	66	66	66	0.0%	100%	30	4	1.00	0.00	0.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_2.RCP	44	44	44	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_3.RCP	61	61	61	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_4.RCP	59	59	59	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_5.RCP	64	64	64	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_6.RCP	46	46	46	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_7.RCP	56	56	56	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_8.RCP	63	63	63	0.0%	100%	30	4	1.00	0.00	0.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_9.RCP	59	59	59	0.0%	100%	30	4	1.00	0.00	0.00	1.00	1.00	1.00	1.00	0.25	2.13
J0036_10.RCP	59	59	59	0.0%	100%	30	4	0.00	1.00	1.00	0.00	1.00	1.00	1.00	0.25	2.13
J007_1.RCP	55	55	55	0.0%	100%	30	4	0.73	0.71	0.71	0.67	0.50	0.51	0.51	1.50	1.50
J007_2.RCP	42	42	42	0.0%	100%	30	4	0.69	0.72	0.72	0.67	0.67	0.71	0.71	0.51	1.50
J007_3.RCP	42	42	42	0.0%	100%	30	4	0.70	0.72	0.72	0.71	0.71	0.71	0.71	0.51	1.50
J007_4.RCP	44	44	44	0.0%	100%	30	4	0.70	0.73	0.73	0.71	0.71	0.71	0.71	0.51	1.50
J007_5.RCP	44	45	44	0.0%	70%	30	4	0.73	0.71	0.71	0.69	0.73	0.73	0.73	0.51	1.50
J007_6.RCP	35	35	35	0.0%	100%	30	4	0.71	0.72	0.72	0.70	0.69	0.69	0.69	0.51	1.50
J007_7.RCP	50	50	50	0.0%	100%	30	4	0.68	0.75	0.75	0.71	1.00	1.00	1.00	0.51	1.50
J007_8.RCP	44	44	44	0.0%	100%	30	4	0.75	0.72	0.72	0.75	0.75	0.75	0.75	0.51	1.50
J007_9.RCP	60	60	60	0.0%	100%	30	4	0.67	0.69	0.69	0.68	0.71	0.71	0.71	0.51	1.50
J007_10.RCP	49	49	49	0.0%	100%	30	4	0.71	0.70	0.70	0.71	0.70	0.70	0.70	0.51	1.50
J008_1.RCP	44	44	44	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_2.RCP	51	51	51	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_3.RCP	53	53	53	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_4.RCP	48	48	48	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_5.RCP	58	58	58	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_6.RCP	47	47	47	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_7.RCP	41	41	41	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_8.RCP	51	51	51	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_9.RCP	39	39	39	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50
J008_10.RCP	67	67	67	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.51	1.50

Fig. A.3 Single objective execution of j30 instances - part III



Filename	Min Dur	Max Dur	Optimal	Aver Dev. Optimum	Frequency of Opt.	Jobs Number	Res Number	Res Strength	Res Strength	Res Strength	Res Strength 3	Res Strength 4	Resource Factor	Network Complexity
55 J3038_9.RCP	63	64	63	0.0%	96%	30	4	0.55	0.55	0.50	0.50	0.50	0.51	2.13
56 J3038_10.RCP	60	61	60	0.0%	80%	30	4	0.50	0.50	0.50	0.50	0.50	0.51	2.13
57 J3039_1.RCP	55	55	55	0.0%	100%	30	4	0.71	0.69	0.67	0.67	0.68	0.51	2.13
58 J3039_2.RCP	54	54	54	0.0%	100%	30	4	0.75	0.72	0.69	0.69	0.67	0.51	2.13
59 J3039_3.RCP	54	54	54	0.0%	100%	30	4	0.73	0.70	0.71	0.71	0.73	0.51	2.13
60 J3039_4.RCP	53	53	53	0.0%	100%	30	4	0.67	0.69	0.68	0.68	0.70	0.51	2.13
61 J3039_5.RCP	55	55	55	0.0%	100%	30	4	0.67	0.70	0.75	0.75	0.69	0.51	2.13
62 J3039_6.RCP	69	70	69	0.0%	60%	30	4	0.68	0.67	0.68	0.68	0.72	0.51	2.13
63 J3039_7.RCP	56	56	56	0.0%	100%	30	4	0.72	0.75	0.75	0.75	0.75	0.51	2.13
64 J3039_8.RCP	67	67	67	0.0%	100%	30	4	0.68	0.70	0.71	0.71	0.71	0.51	2.13
65 J3039_9.RCP	64	65	64	0.0%	60%	30	4	0.69	0.71	0.70	0.70	0.71	0.51	2.13
66 J3039_10.RCP	60	60	60	0.0%	100%	30	4	0.72	0.73	0.67	0.67	0.69	0.51	2.13
67 J3040_1.RCP	51	51	51	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
68 J3040_2.RCP	56	56	56	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
69 J3040_3.RCP	57	57	57	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
70 J3040_4.RCP	57	57	57	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
71 J3040_5.RCP	65	65	65	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
72 J3040_6.RCP	60	60	60	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
73 J3040_7.RCP	46	46	46	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
74 J3040_8.RCP	57	57	57	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
75 J3040_9.RCP	64	64	64	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.51	2.13
01 J3011_8.RCP	62	63	62	0.0%	97%	30	4	0.68	0.71	0.71	0.71	0.72	0.76	1.50
02 J3011_9.RCP	67	67	67	0.0%	100%	30	4	0.71	0.71	0.70	0.70	0.69	0.76	1.50
03 J3011_10.RCP	38	40	38	0.0%	80%	30	4	0.70	0.71	0.69	0.69	0.69	0.76	1.50
04 J3012_1.RCP	47	47	47	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
05 J3012_2.RCP	46	46	46	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
06 J3012_3.RCP	37	37	37	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
07 J3012_4.RCP	63	63	63	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
08 J3012_5.RCP	47	47	47	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
09 J3012_6.RCP	53	53	53	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
10 J3012_7.RCP	55	55	55	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
11 J3012_8.RCP	35	35	35	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
12 J3012_9.RCP	52	52	52	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
13 J3012_10.RCP	57	57	57	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	1.50
41 J3044_1.RCP	50	50	50	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	2.13
42 J3044_2.RCP	54	54	54	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76	2.13

Fig. A.4 Single objective execution of j30 instances - part IV

Filename	Min Dur	Max Dur	Optimal	Aver Dev. Optimum	Frequency of Opt.	Jobs Number	Res Number	Res Strength	Res Streng	Res Strength 3	Res Strength 4	Resource Factor	Network Complexity
J0044_4.RCP	57	57	57	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76
J0044_5.RCP	55	55	55	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76
J0044_6.RCP	56	56	56	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76
J0044_7.RCP	42	42	42	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76
J0044_8.RCP	49	49	49	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76
J0044_9.RCP	64	64	64	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76
J0044_10.RCP	63	63	63	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	0.76
J0044_10.RCP	61	63	61	0.0%	30%	30	4	0.50	0.52	0.50	0.50	0.50	1.00
J0015_1.RCP	46	46	46	0.0%	100%	30	4	0.71	0.71	0.71	0.71	0.71	1.00
J0015_2.RCP	47	47	47	0.0%	100%	30	4	0.70	0.70	0.69	0.69	0.69	1.00
J0015_3.RCP	48	48	48	0.0%	100%	30	4	0.69	0.70	0.68	0.69	0.69	1.00
J0015_4.RCP	48	48	48	0.0%	100%	30	4	0.71	0.70	0.69	0.71	0.71	1.00
J0015_5.RCP	59	64	58	1.7%	0%	30	4	0.71	0.71	0.71	0.68	0.68	1.00
J0015_6.RCP	67	67	67	0.0%	100%	30	4	0.71	0.68	0.72	0.72	0.69	1.00
J0015_7.RCP	47	47	47	0.0%	100%	30	4	0.69	0.71	0.68	0.70	0.70	1.00
J0015_8.RCP	50	50	50	0.0%	100%	30	4	0.68	0.71	0.72	0.68	0.69	1.00
J0015_9.RCP	54	54	54	0.0%	100%	30	4	0.68	0.71	0.71	0.71	0.71	1.00
J0015_10.RCP	65	65	65	0.0%	100%	30	4	0.72	0.68	0.68	0.69	0.69	1.00
J0016_1.RCP	51	51	51	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_2.RCP	48	48	48	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_3.RCP	36	36	36	0.0%	40%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_4.RCP	47	47	47	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_5.RCP	51	51	51	0.0%	80%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_6.RCP	51	51	51	0.0%	60%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_7.RCP	34	34	34	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_8.RCP	44	44	44	0.0%	10%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_9.RCP	44	44	44	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0016_10.RCP	51	51	51	0.0%	100%	30	4	1.00	1.00	1.00	1.00	1.00	1.50
J0031_1.RCP	43	43	43	0.0%	100%	30	4	0.70	0.70	0.69	0.71	0.71	1.00
J0031_2.RCP	63	63	63	0.0%	100%	30	4	0.71	0.69	0.71	0.69	0.69	1.00
J0031_3.RCP	58	58	58	0.0%	100%	30	4	0.70	0.71	0.70	0.70	0.69	1.00
J0031_4.RCP	50	50	50	0.0%	100%	30	4	0.71	0.70	0.71	0.70	0.70	1.00
J0031_5.RCP	52	56	52	0.6%	10%	30	4	0.72	0.68	0.68	0.72	0.72	1.00
J0031_6.RCP	53	53	53	0.0%	100%	30	4	0.70	0.70	0.71	0.70	0.70	1.00
J0031_7.RCP	61	65	61	0.0%	70%	30	4	0.68	0.69	0.69	0.71	0.73	1.00
J0031_8.RCP	58	58	58	0.0%	100%	30	4	0.71	0.71	0.71	0.71	0.70	1.00

Fig. A.5 Single objective execution of j30 instances - part V



## A.2 Exerpt of analytical results MRCPSP

Filename	Min Dur	Max Dur	Optimal	Aver.Dev. Opti	Frequency o
c154_3.mm	34	34	34	0,00%	100%
c158_3.mm	25	25	25	0,00%	100%
c158_4.mm	32	32	32	0,00%	100%
c159_1.mm	18	21	18	0,00%	99%
c159_2.mm	25	26	29	-13,79%	97%
c159_3.mm	22	24	22	0,00%	100%
c159_4.mm	17	21	17	0,00%	100%
c159_5.mm	21	22	21	0,00%	100%
c159_6.mm	20	22	20	0,00%	100%
c159_7.mm	24	28	24	4,17%	84%
c159_8.mm	34	40	34	0,00%	100%
c159_9.mm	28	29	28	0,00%	100%
c159_10.mm	32	32	32	0,00%	90%
c1510_1.mm	21	21	21	0,00%	100%
c1510_2.mm	17	18	17	0,00%	100%
c1510_3.mm	23	25	23	0,00%	100%
c1510_4.mm	39	42	39	0,00%	98%
c1510_5.mm	13	13	13	0,00%	100%
c1510_6.mm	32	32	32	0,00%	100%
...	...	...	...	...	...
c214_6.mm	36	36	36	0,00%	100%
c216_8.mm	36	36	36	0,00%	100%
c217_1.mm	40	41	40	0,00%	98%
c219_1.mm	28	30	30	0,00%	99%
c219_2.mm	29	32	29	0,00%	96%
c219_3.mm	26	28	26	0,00%	99%
c219_4.mm	26	28	26	0,00%	98%
c219_5.mm	29	30	29	0,00%	97%
c219_6.mm	22	24	22	4,55%	88%
c219_7.mm	25	28	29	0,00%	98%
c219_8.mm	21	24	21	0,00%	99%
c219_9.mm	28	33	28	0,00%	98%
c219_10.mm	21	21	21	0,00%	100%
c2110_1.mm	21	23	21	0,00%	99%
...	...	...	...	...	...
j102_2.mm	18	21	20	0,00%	98%
j102_4.mm	17	17	18	0,00%	100%
j102_5.mm	16	17	16	0,00%	99%
j102_6.mm	16	16	16	0,00%	100%
j102_7.mm	25	25	25	0,00%	100%
j102_9.mm	15	15	17	0,00%	100%
j102_10.mm	33	33	33	0,00%	100%
j103_2.mm	13	13	13	0,00%	100%
j103_3.mm	19	19	19	0,00%	100%
j103_4.mm	23	23	23	0,00%	100%
j103_5.mm	19	21	21	0,00%	100%

Fig. A.6 Single objective execution of MRCPSP instances

### A.3 Exerpt of analytical results MRCPSP/max

Filename	Min Dur	Max Dur	UB	Aver.Dev. Optimum	Frequency of Opt
psp1.sch	42	49	42	0,0%	70%
psp2.sch	33	38	33	0,0%	90%
psp3.sch	46	46	46	0,0%	100%
psp4.sch	33	36	33	0,0%	50%
psp5.sch	25	29	25	0,0%	60%
psp6.sch	33	33	33	0,0%	100%
psp8.sch	39	39	39	0,0%	100%
psp9.sch	33	33	33	0,0%	100%
psp10.sch	32	32	32	0,0%	100%
psp11.sch	28	28	28	0,0%	100%
psp12.sch	25	25	25	0,0%	100%
psp13.sch	30	30	30	0,0%	100%
psp14.sch	35	35	35	0,0%	100%
psp15.sch	28	28	28	0,0%	100%
psp16.sch	26	26	26	0,0%	100%
psp17.sch	42	42	42	0,0%	100%
...	...	...	...	...	...
psp47.sch	26	26	26	0,0%	100%
psp48.sch	31	31	31	0,0%	100%
psp49.sch	18	18	18	0,0%	100%
psp50.sch	24	24	24	0,0%	100%
psp51.sch	26	26	26	0,0%	100%
psp52.sch	30	30	30	0,0%	100%
psp53.sch	28	28	28	0,0%	100%
psp54.sch	25	25	25	0,0%	100%
psp55.sch	38	38	38	0,0%	100%
psp56.sch	37	37	37	0,0%	100%
psp57.sch	30	30	30	0,0%	100%
psp58.sch	26	26	26	0,0%	100%
psp59.sch	24	24	24	0,0%	100%
psp60.sch	29	29	29	0,0%	100%
psp61.sch	40	40	40	0,0%	100%
psp62.sch	38	38	38	0,0%	100%
psp63.sch	30	30	30	0,0%	100%
psp64.sch	36	36	36	0,0%	100%
psp65.sch	25	25	25	0,0%	100%
psp66.sch	30	30	30	0,0%	100%
psp67.sch	38	38	38	0,0%	100%
psp68.sch	31	31	31	0,0%	100%
psp69.sch	32	32	32	0,0%	100%
psp70.sch	22	22	22	0,0%	100%
psp71.sch	33	33	33	0,0%	100%
psp72.sch	20	20	20	0,0%	100%
psp73.sch	34	34	34	0,0%	100%
psp74.sch	39	39	39	0,0%	100%
psp75.sch	33	33	33	0,0%	100%

Fig. A.7 Single objective execution of MRCPSP instances

## **A.4 Case Study**

### ***A.4.1 Activities***

WBS	Task Name	Duration	Start	Finish	Resource Names
	<b>Project: GIS database update</b>	<b>1024 days</b>	<b>5/3/2012</b>	<b>9/3/2016</b>	
<b>0</b>	<b>Project Management</b>	<b>1023 days</b>	<b>5/3/2012</b>	<b>8/3/2016</b>	<b>Project Manager</b>
<b>1</b>	<b>1st Stage: Creation of initial countrywide basemap - Establish GIS land registry offices</b>	<b>131 days</b>	<b>5/3/2012</b>	<b>5/9/2012</b>	
1.0	Contracts	0 days	5/3/2012	5/3/2012	
<b>1.1</b>	<b>Phase 1.1. Preparation activities</b>	<b>31 days</b>	<b>5/3/2012</b>	<b>17/4/2012</b>	
1.1.1	Project Initialisation	5 days	5/3/2012	9/3/2012	
1.1.1.2	Project analysis	5 days	5/3/2012	9/3/2012	GIS,Surveyor Engineer1
1.1.1.3	Kick-off meeting	1 day	5/3/2012	5/3/2012	Civil Engineer1,Surveyor Engineer1,Lawyer1
1.1.1.4	Data receipts from Greek Land Registry	0 days	5/3/2012	5/3/2012	
1.1.2	Preparation- Supply of equipment - Hardware and Software installations - Quality control	23 days	5/3/2012	4/4/2012	
1.1.2.1	Hardware and software supply	15 days	5/3/2012	23/3/2012	Surveyor Engineer1,IT 1,Civil Engineer1,GIS
1.1.2.2	Installations of equipment, hardware and software	15 days	14/3/2012	3/4/2012	IT 1,GIS
1.1.2.3	Hire additional personnel	9 days	5/3/2012	15/3/2012	Civil Engineer1
1.1.2.4	Training of new hires	14 days	16/3/2012	4/4/2012	GIS,IT 1,Surveyor Engineer1,Surveyor Engineer Junior1[21]
1.1.3	Quality Control Programme (Q.C.P)	31 days	5/3/2012	17/4/2012	
1.1.3.1	Timeschedule activities	9 days	5/3/2012	15/3/2012	Civil Engineer1,Surveyor Engineer1
1.1.3.2	Quality Plan	12 days	16/3/2012	2/4/2012	Civil Engineer1,Surveyor Engineer1,Lawyer1
1.1.3.3	Quality Plan Approval	1 day	3/4/2012	3/4/2012	Civil Engineer1
1.1.3.4	Submission of Q.C.P.	0 days	3/4/2012	3/4/2012	
1.1.3.5	Corrections (if required)	9 days	4/4/2012	17/4/2012	Civil Engineer1,Surveyor Engineer1,Lawyer1
1.1.3.6	Resubmission (if required)	0 days	17/4/2012	17/4/2012	
<b>1.2</b>	<b>Phase 1.2 Initial countrywide basemap</b>	<b>131 days</b>	<b>5/3/2012</b>	<b>5/9/2012</b>	
1.2.1	Collection and evaluation of pre-existing material	23 days	5/3/2012	4/4/2012	
1.2.1.1	Data collection	23 days	5/3/2012	4/4/2012	Lawyer1[4],Surveyor Engineer1[2]
1.2.1.2	Data evaluation	17 days	12/3/2012	3/4/2012	Lawyer1[3],Surveyor Engineer1[5]
1.2.2	City and country limits definitions	66 days	5/3/2012	5/6/2012	
1.2.2.1	Representation of collected data in visio and Iso	3 days	5/3/2012	7/3/2012	GIS,Surveyor Engineer1[2]
1.2.2.2	Corrections based on administrative and legal data	33 days	5/4/2012	22/5/2012	Lawyer1[4],Surveyor Engineer1[2]
1.2.2.3	Compare results with existing from the Greek Land Registry	9 days	5/3/2012	15/3/2012	GIS,Surveyor Engineer1
1.2.2.4	Add limits based on land use classification	19 days	5/4/2012	2/5/2012	GIS,Surveyor Engineer1,Surveyor Engineer Junior1[3]
1.2.2.5	Add limits of settlements	14 days	5/4/2012	25/4/2012	GIS,Surveyor Engineer1,Surveyor Engineer Junior1[3]
1.2.2.6	Add limits within the cities' Plan	19 days	5/4/2012	2/5/2012	GIS,Surveyor Engineer1,Surveyor Engineer Junior1[3]
1.2.2.7	Check polygonal topologies	3 days	23/5/2012	25/5/2012	GIS,Surveyor Engineer1[2]
1.2.2.8	Technical Report	7 days	28/5/2012	5/6/2012	Surveyor Engineer1,GIS,IT 1,Civil Engineer1,OTHER1
1.2.2.9	Submission of deliverable	0 days	5/6/2012	5/6/2012	
1.2.2.10	Receive corrections and update report and gis data	0 days	5/6/2012	5/6/2012	
1.2.3	Creation of initial countrywide basemap	128 days	8/3/2012	5/9/2012	
1.2.3.1	Update basemap with locality names, rivers, lakes, etc.	29 days	8/3/2012	18/4/2012	Surveyor Engineer1[3]
1.2.3.2	Surface measurements of urban and rural sections peaks	27 days	5/4/2012	14/5/2012	Surveyor Engineer1,Surveyor Engineer Junior1[4],OTHER1
1.2.3.3	Mapping the road network and points of interest	27 days	5/4/2012	14/5/2012	Surveyor Engineer Junior1[3]
1.2.3.4	Capture and edit existing cartographic material (administrative acts, cadastral / topographic basemap)	59 days	5/4/2012	27/6/2012	Surveyor Engineer Junior1[3],OTHER1
1.2.3.5	Define cadastral areas and units on the basemap	50 days	3/5/2012	11/7/2012	Surveyor Engineer1,Surveyor Engineer Junior1[6]
1.2.3.6	Draft land registration and digitization for urban areas and tables of alleged owners	76 days	8/5/2012	22/8/2012	Surveyor Engineer1[2],Lawyer1[2],Surveyor Engineer Junior1[4]
1.2.3.7	Draft land registration and digitization for rural areas and tables of alleged owners	76 days	8/5/2012	22/8/2012	Surveyor Engineer1,Lawyer1,Surveyor Engineer Junior1[5]
1.2.3.8	Pairing of temporary codes to land parcels and pairing of these numbers to the related street	76 days	11/5/2012	27/8/2012	Surveyor Engineer1,Surveyor Engineer Junior1,OTHER1

WBS	Task Name	Duration	Start	Finish	Resource Names
1.2.3.9	Control of topology polygons	10 days	14/8/2012	28/8/2012	GIS, Surveyor Engineer1[2], Surveyor Engineer Junior1
1.2.3.10	Technical report of initial countrywide basemap	6 days	29/8/2012	5/9/2012	Surveyor Engineer1, GIS, Civil Engineer1, OTHER1
1.2.3.11	Submission of technical report	0 days	5/9/2012	5/9/2012	
<b>1.3</b>	<b>Phase 1.3 Preparation of land offices &amp; operation</b>	<b>76 days</b>	<b>5/3/2012</b>	<b>19/6/2012</b>	
1.3.1	Preparation of land offices	55 days	5/3/2012	21/5/2012	
1.3.1.1	Receive offices and post orthophotomaps	10 days	5/3/2012	16/3/2012	Surveyor Engineer1
1.3.1.2	Hardware supplies	5 days	19/3/2012	23/3/2012	Surveyor Engineer1, GIS, Civil Engineer1, IT 1
1.3.1.3	New hires	20 days	19/3/2012	13/4/2012	Civil Engineer 1
1.3.1.4	Set up infrastructure	19 days	19/3/2012	12/4/2012	Surveyor Engineer1
1.3.1.5	Cabling and network connectivity	20 days	13/4/2012	11/5/2012	IT 1
1.3.1.6	Installations of equipment, hardware and software	20 days	13/4/2012	11/5/2012	GIS, IT 1
1.3.1.7	Installation of Fire and Security systems	15 days	13/4/2012	4/5/2012	Surveyor Engineer1, Civil Engineer1
1.3.1.8	New hires training	25 days	17/4/2012	21/5/2012	Lawyer1[3], Surveyor Engineer1[4]
1.3.1.9	Preparation of hard copy material	20 days	5/3/2012	30/3/2012	Surveyor Engineer1, OTHER1
1.3.1.10	Technical control of IT systems	4 days	14/5/2012	17/5/2012	GIS, IT 1
1.3.1.11	Checking readiness for the smooth operation of the land offices and receipt of approval	1 day	18/5/2012	18/5/2012	GIS, IT 1, Surveyor Engineer1
1.3.3	Operation of land offices	22 days	21/5/2012	19/6/2012	
1.3.3.1	Start of operation	0 days	21/5/2012	21/5/2012	
1.3.3.2	Compliance with the observations of the Greek Land Registry	20 days	21/5/2012	15/6/2012	Surveyor Engineer1, GIS, IT 1
1.3.3.3	Support services	22 days	21/5/2012	19/6/2012	Lawyer Junior1[5], IT Junior1[7], Surveyor Engineer1[2], Lawyer1[8], OTHER1[3], Surveyor Engineer Junior1[9]
1.3.3.4	Approval of Greek Land Registry	0 days	19/6/2012	19/6/2012	
<b>1.4</b>	<b>Completion of Stage 1</b>	<b>0 days</b>	<b>5/9/2012</b>	<b>5/9/2012</b>	
<b>2</b>	<b>2nd Stage - Development of GIS data base</b>	<b>398 days</b>	<b>5/3/2012</b>	<b>20/9/2013</b>	
<b>2.0</b>	<b>Stage 2- Start</b>	<b>0 days</b>	<b>7/11/2012</b>	<b>7/11/2012</b>	
<b>2.1</b>	<b>Phase 2.1: Operation of land offices</b>	<b>175 days</b>	<b>5/3/2012</b>	<b>7/11/2012</b>	
2.1.0	Statements collection	0 days	7/11/2012	7/11/2012	
2.1.1	Collection and preprocessing of statements of land owners living in Greece	1 day	5/3/2012	5/3/2012	
2.1.2	Collection and preprocessing of statements of land owners living abroad	1 day	5/3/2012	5/3/2012	
2.1.3	Collection and preprocessing of overdue statements of land owners living abroad	1 day	5/3/2012	5/3/2012	
<b>2.1.4</b>	<b>Completion of stage 2 data collection</b>	<b>0 days</b>	<b>5/3/2012</b>	<b>5/3/2012</b>	
<b>2.2</b>	<b>Phase 2.2: Generate GIS data base</b>	<b>398 days</b>	<b>5/3/2012</b>	<b>20/9/2013</b>	
2.2.1	Preparation and submission of 1st intermediate database	209 days	5/3/2012	24/12/2012	
2.2.1.1	Process and input all data to the central database	164 days	5/3/2012	22/10/2012	Lawyer1[3], Lawyer Junior1[3], OTHER1
2.2.1.2	Data retrieval and migration of data of locked entries	164 days	15/3/2012	1/11/2012	Lawyer1[2], OTHER1[2], Lawyer Junior1[3]
2.2.1.3	Correct and add notes to data entries based on the results of legal control and cross verification of data using sampling methods	20 days	2/11/2012	29/11/2012	Lawyer Junior1[3], OTHER1[3]
2.2.1.4	Correct and add notes to data entries based on the results of legal control and cross verification of data using automated validation algorithms	20 days	2/11/2012	29/11/2012	IT 1, Lawyer Junior1[2], OTHER1[4], IT Junior1[2], Surveyor Engineer Junior1[5]
2.2.1.5	Cross check and validate duplicate entries referring to land owners	14 days	30/11/2012	19/12/2012	IT 1, Lawyer Junior1[3], OTHER1[3], IT Junior1, Surveyor Engineer1
2.2.1.6	Define land limits and owners of each building and pair it to unique KAEK code	164 days	5/3/2012	22/10/2012	Surveyor Engineer1, Surveyor Engineer Junior1[3], OTHER1[4]



WBS	Task Name	Duration	Start	Finish	Resource Names
2.2.1.7	Define land boundaries where no data have been given by the owners	20 days	23/10/2012	19/11/2012	Surveyor Engineer1, Surveyor Engineer Junior1[3], OTHER1
2.2.1.8	Enumerate buildings and geographically position them	20 days	23/10/2012	19/11/2012	Surveyor Engineer Junior1[4]
2.2.1.9	Topological validation of polygon based spatial data and update of the digital cadastral spatial data base	14 days	20/11/2012	7/12/2012	GIS, Surveyor Engineer1[2], Surveyor Engineer Junior1
2.2.1.10	Generate description and geospatial data bases	3 days	20/12/2012	24/12/2012	GIS, IT 1
2.2.1.11	Submission of 1st intermediate data base	0 days	24/12/2012	24/12/2012	
2.2.1.12	Cross check and validate duplicate entries referring to land boundaries	14 days	5/3/2012	22/3/2012	IT 1, Lawyer Junior1[3], OTHER1[3], IT Junior1, Surveyor Engineer Junior1
2.2.2	<i>Preparation and submission of 2nd intermediate database</i>	109 days	23/10/2012	27/3/2013	
2.2.2.1	Process and input all data to the central database	96 days	23/10/2012	7/3/2013	Lawyer1[3], Lawyer Junior1[4], OTHER1
2.2.2.2	Data retrieval and migration of data of locked entries	87 days	2/11/2012	6/3/2013	Lawyer1, OTHER1[2], Lawyer Junior1[3]
2.2.2.3	Correct and add notes to data entries based on the results of legal control and cross verification of data using sampling methods	7 days	7/3/2013	15/3/2013	Lawyer Junior1[4], OTHER1[3], Surveyor Engineer Junior1
2.2.2.4	Correct and add notes to data entries based on the results of legal control and cross verification of data using automated validation algorithms	7 days	7/3/2013	15/3/2013	Lawyer Junior1[3], OTHER1
2.2.2.5		7 days	7/3/2013	15/3/2013	IT 1, OTHER1[4], Lawyer Junior1[2], IT Junior1[2], Surveyor Engineer Junior1[4]
2.2.2.6	Cross check and validate duplicate entries referring to land boundaries	5 days	18/3/2013	22/3/2013	IT 1, Surveyor Engineer Junior1, OTHER1[3], Lawyer Junior1[3]
2.2.2.7	Define land limits and owners of each building and pair it to unique KAEK code	94 days	23/10/2012	5/3/2013	Surveyor Engineer1, Surveyor Engineer Junior1[4], OTHER1[4]
2.2.2.8	Define land boundaries where no data have been given by the owners	7 days	6/3/2013	14/3/2013	Surveyor Engineer1, OTHER1, Surveyor Engineer Junior1[3]
2.2.2.9	Enumerate buildings and geographically position them	7 days	6/3/2013	14/3/2013	Surveyor Engineer Junior1[4]
2.2.2.10	Topological validation of polygon based spatial data and update of the digital cadastral spatial data base	5 days	15/3/2013	21/3/2013	GIS, Surveyor Engineer1[2], Surveyor Engineer Junior1
2.2.2.11	Generate description and geospatial data bases	2 days	26/3/2013	27/3/2013	GIS, IT 1
2.2.2.12	Submission of 2nd intermediate data base	0 days	27/3/2013	27/3/2013	
2.2.3	<i>Preparation and submission of 3rd intermediate database</i>	94 days	8/3/2013	19/7/2013	
2.2.3.1	Process and input all data to the central database	79 days	8/3/2013	28/6/2013	Lawyer1[3], OTHER1, Lawyer Junior1[2]
2.2.3.2	Data retrieval and migration of data of locked entries	73 days	15/3/2013	27/6/2013	Lawyer1, OTHER1[2], Lawyer Junior1[3]
2.2.3.3	Correct and add notes to data entries based on the results of legal control and cross verification of data using sampling methods	10 days	28/6/2013	11/7/2013	Lawyer Junior1[3], OTHER1[3]
2.2.3.4	Correct and add notes to data entries based on the results of legal control and cross verification of data using automated validation algorithms	10 days	28/6/2013	11/7/2013	Lawyer Junior1[3], OTHER1, Surveyor Engineer Junior1
2.2.3.5	Cross check and validate duplicate entries referring to land owners	10 days	28/6/2013	11/7/2013	IT 1, Lawyer Junior1[2], OTHER1[4], IT Junior1, Surveyor Engineer Junior1[4]
2.2.3.6	Cross check and validate duplicate entries referring to land boundaries	5 days	12/7/2013	18/7/2013	IT 1, Lawyer1, Lawyer Junior1[2], OTHER1[3], Surveyor Engineer Junior1
2.2.3.7	Define land limits and owners of each building and pair it to unique KAEK code	73 days	8/3/2013	20/6/2013	Surveyor Engineer1, Surveyor Engineer Junior1[4], OTHER1
2.2.3.8	Define land boundaries where no data have been given by the owners	10 days	21/6/2013	4/7/2013	Surveyor Engineer1, OTHER1, Surveyor Engineer Junior1[3]
2.2.3.9	Enumerate buildings and geographically position them	10 days	21/6/2013	4/7/2013	Surveyor Engineer Junior1[4]
2.2.3.10	Topological validation of polygon based spatial data and update of the digital cadastral spatial data base	6 days	5/7/2013	12/7/2013	GIS, Surveyor Engineer1[2], Surveyor Engineer1, Surveyor Engineer Junior1
2.2.3.11	Generate description and geospatial data bases	1 day	19/7/2013	19/7/2013	GIS, IT 1

WBS	Task Name	Duration	Start	Finish	Resource Names
2.2.3.12	Submission of 3rd intermediate data base	0 days	19/7/2013	19/7/2013	
2.2.4	<i>Preparation for final submission</i>	59 days	1/7/2013	20/9/2013	
2.2.4.1	Edit entries and enter to the central database. Check legal issues related to the landowners	58 days	1/7/2013	19/9/2013	Lawyer1[3],Lawyer Junior1[4],OTHER1
2.2.4.2	Migrate data from local data bases to Greek Land Registry central database	37 days	8/7/2013	28/8/2013	Lawyer1[2],OTHER1[2],Lawyer Junior1[2]
2.2.4.3	Correct and complete missing data based on owners statements	7 days	29/8/2013	6/9/2013	Lawyer Junior1[3],OTHER1[3]
2.2.4.4	Update data base based on comments by the Greek Land Registry	7 days	29/8/2013	6/9/2013	Lawyer Junior1[3],OTHER1,IT Junior1
2.2.4.5	Correct data base based on the results of sampling	7 days	29/8/2013	6/9/2013	IT 1,OTHER1[4],Lawyer Junior1[2],IT Junior1,Surveyor Engineer Junior1[4]
2.2.4.6	Remove duplicate entries	5 days	9/9/2013	13/9/2013	IT 1,Lawyer Junior1[3],OTHER1[3]
2.2.4.7	Validate data tracking and delineation of properties. Validate geometric compatibility of parcels included in the database. Assign specific codes to the properties of all the registred landowners	44 days	1/7/2013	30/8/2013	Surveyor Engineer1[2],Surveyor Engineer Junior1[4],OTHER1[4]
2.2.4.8	Define land boundaries where no data have been given by the owners	7 days	2/9/2013	10/9/2013	Surveyor Engineer1,OTHER1,Surveyor Engineer Junior1[3]
2.2.4.9	Enumerate buildings and geographically position them	7 days	2/9/2013	10/9/2013	Surveyor Engineer Junior1[4]
2.2.4.10	Topological validation of polygon based spatial data and update of the digital cadastral spatial data base	4 days	11/9/2013	16/9/2013	GIS,Surveyor Engineer1[2],Surveyor Engineer Junior1
2.2.4.11	Generate description and geospatial data bases	4 days	17/9/2013	20/9/2013	GIS,IT 1
2.2.4.12	Technical report of implementation process	5 days	16/9/2013	20/9/2013	GIS,IT 1,OTHER1
2.2.4.13	Final submission of scanned land owners statements	0 days	20/9/2013	20/9/2013	
2.2.4.14	Report for internal control	0 days	20/9/2013	20/9/2013	
2.2.4.15	Submit final report of the implementation process	0 days	20/9/2013	20/9/2013	
2.5	<b>Completion of Stage 2</b>	0 days	20/9/2013	20/9/2013	
3	<b>3rd Stage: Final Submission</b>	<b>446 days</b>	<b>10/6/2014</b>	<b>9/3/2016</b>	
3.0	<b>Stage 3 -Start</b>	<b>0 days</b>	<b>10/6/2014</b>	<b>10/6/2014</b>	
3.7	Finalise elaboration of the collected data and resulting tables	4 days	10/6/2014	13/6/2014	IT 1,Lawyer1[2],Surveyor Engineer1[2],GIS,Lawyer Junior1[3],OTHER1[4],IT Junior1[2],Surveyor Engineer Junior1
3.8	Final submission of collected data and resulting tables	0 days	8/3/2016	8/3/2016	
3.9	Final technical report	12 days	10/6/2014	3/7/2014	Civil Engineer1,Surveyor Engineer1,Lawyer Junior1,OTHER1
3.10	Submission of final technical report	0 days	3/7/2014	3/7/2014	
3.13	<b>Completion of Stage 3</b>	<b>0 days</b>	<b>8/3/2016</b>	<b>8/3/2016</b>	

***A.4.2 Gantt chart***

ID	WBS	1st Half						1st Half						1st Half						1st Half					
		1st Quarter		3rd Quarter		1st Quarter		3rd Quarter		1st Quarter		3rd Quarter		1st Quarter		3rd Quarter		1st Quarter		3rd Quarter					
		Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	
0																									
1	0	3/5/12																							
2	1																								
3	1.0	3/5/12																							
4	1.1																								
5	1.1.1																								
6	1.1.1.1.2	3/5/12 3/9/12																							
7	1.1.1.1.3	3/5/12 3/5/12																							
8	1.1.1.1.4	3/5/12																							
9	1.1.2																								
10	1.1.2.1	3/5/12 3/23/12																							
11	1.1.2.2	3/14/12 4/3/12																							
12	1.1.2.3	3/5/12 3/15/12																							
13	1.1.2.4	3/16/12 4/4/12																							
14	1.1.3																								
15	1.1.3.1	3/5/12 3/15/12																							
16	1.1.3.2	3/16/12 4/2/12																							
17	1.1.3.3	4/3/12 4/3/12																							
18	1.1.3.4	4/3/12																							
19	1.1.3.5	4/4/12 4/17/12																							
20	1.1.3.6	4/17/12																							
21	1.2																								
22	1.2.1																								
23	1.2.1.1	3/5/12 4/4/12																							
24	1.2.1.2	3/12/12 4/3/12																							
25	1.2.2																								
26	1.2.2.1	3/5/12 3/7/12																							







ID	WBS	1st Half						1st Half						1st Half						1st Half									
		1st Quarter			3rd Quarter			1st Quarter			3rd Quarter			1st Quarter			3rd Quarter			1st Quarter			3rd Quarter						
		Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May
93	2.2.2.4							3/7/13					3/15/13																
94	2.2.2.5							3/7/13					3/15/13																
95	2.2.2.6							3/18/13					3/22/13																
96	2.2.2.7																												
97	2.2.2.8							3/6/13					3/14/13																
98	2.2.2.9							3/6/13					3/14/13																
99	2.2.2.10							3/15/13					3/21/13																
100	2.2.2.11							3/26/13					3/27/13																
101	2.2.2.12																												
102	2.2.3																												
103	2.2.3.1							3/8/13					6/28/13																
104	2.2.3.2							3/15/13					6/27/13																
105	2.2.3.3												6/28/13				7/11/13												
106	2.2.3.4												6/28/13				7/11/13												
107	2.2.3.5												6/28/13				7/11/13												
108	2.2.3.6												7/12/13				7/18/13												





ID	WBS	1st Half					1st Half					1st Half					1st Half					1st Half							
		1st Quarter			3rd Quarter		1st Quarter			3rd Quarter		1st Quarter			3rd Quarter		1st Quarter			3rd Quarter		1st Quarter			3rd Quarter				
		Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May	Jul	Sep	Nov	Jan	Mar	May
126	2.2.4.11										9/17/13	9/20/13																	
127	2.2.4.12										9/16/13	9/20/13																	
128	2.2.4.13											9/20/13																	
129	2.2.4.14											9/20/13																	
130	2.2.4.15											9/20/13																	
131	2.5											9/20/13																	
132	3																												
133	3.0																												
134	3.7																												
135	3.8																												
136	3.9																												
137	3.10																												
138	3.13																												





Task		Inactive Task		Start-only	
Split		Inactive Milestone		Finish-only	
Milestone		Inactive Summary		Deadline	
Summary		Manual Task		Critical	
Project Summary		Duration-only		Critical Split	
External Tasks		Manual Summary Rollup		Progress	
External Milestone		Manual Summary			



**Appendix B**  
**Implemented Code – Main Modules**

```

class ModeratorGA
{
    public List<CommonVars.schedule> ModGA(CommonVars.EmoData inputData,
CommonVars.problemParams probParams)
    {
        Functions.AuxiliaryFunctions AuxFuns = new AuxiliaryFunctions();
        SingleObjectiveGA soGA = new SingleObjectiveGA();
        MOGA moGA = new MOGA();
        Pareto parGA = new Pareto();

        List<CommonVars.schedule> resultSchedules = new List<CommonVars.schedule>();
        int jobNum = inputData.jobNum;
        int GEN = probParams.GEN;
        int numOfRepetitions = probParams.numOfRepetitions;

        //Repetition
        for (int repetitions = 0; repetitions < numOfRepetitions; repetitions++)
        {
            Stopwatch stopWatch = new Stopwatch();
            stopWatch.Start();
            Console.WriteLine("Repetition {0}-----", repetitions + 1);

            // Initial Population

            CommonVars.adaptiveChromosome[] InPopulation =
AuxFuns.InitialPopulation(inputData, probParams);

            // Generation

            int NumOfGenerations = 0;
            CommonVars.adaptiveChromosome[] CurrentPopulation = InPopulation;

            while (NumOfGenerations < GEN)
            {
                NumOfGenerations++;

                // Crossover
                CommonVars.adaptiveChromosome[] ChildrenPopulation =
AuxFuns.CrossoverM_act(CurrentPopulation, probParams,inputData);
                // Mutation
                ChildrenPopulation = AuxFuns.MutationM(ChildrenPopulation,
probParams, inputData);

                //Mode repair and improvement

                // Current Union Offspring
                CommonVars.adaptiveChromosome[] UnionPopulation = new
CommonVars.adaptiveChromosome[2 * probParams.POP];
                for (int i = 0; i < probParams.POP; i++)
                {
                    UnionPopulation[i] = CurrentPopulation[i];
                    UnionPopulation[i + probParams.POP] = ChildrenPopulation[i];
                }
                for (int i = 0; i < UnionPopulation.Length; i++)
                {
                    if (AuxFuns.checkModeNonRenFS(UnionPopulation[i].modeList,
inputData, probParams) == false)

```

```

        AuxFuns.ModeRepair(ref UnionPopulation[i], inputData);
    }

    switch (probParams.caseType)
    {
        case 0: CurrentPopulation=soGA.SingleObjGA(inputData, probParams,
UnionPopulation);
            break;
        case 1: CurrentPopulation=moGA.moGA(inputData, probParams,
UnionPopulation);
            break;
        case 2: CurrentPopulation=parGA.ParetoGA(inputData, probParams,
UnionPopulation);
            break;
        default:
            Console.WriteLine("Not acceptable case type option");
            break;
    }

    Console.WriteLine("-----
");
    for (int i = 0; i < CurrentPopulation.Length; i++)
        Console.WriteLine("{0}\t--{1:N2}--{2}",
CurrentPopulation[i].fit.makespan, CurrentPopulation[i].fit.RLI,
CurrentPopulation[i].Algo);
    Console.WriteLine("-----
");

    stopWatch.Stop();
    // Get the elapsed time as a TimeSpan value.
    TimeSpan ts1 = stopWatch.Elapsed;

    string elapsedTime1 = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
ts1.Hours, ts1.Minutes, ts1.Seconds,
ts1.Milliseconds / 10);
    Console.WriteLine("RunTime " + elapsedTime1);

}
// Console.WriteLine("Min of repetition: " + generMin);
// int minSchDuration = generMin;

////////////////////////////////////

stopWatch.Stop();
// Get the elapsed time as a TimeSpan value.
TimeSpan ts = stopWatch.Elapsed;

string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
ts.Hours, ts.Minutes, ts.Seconds,
ts.Milliseconds / 10);
Console.WriteLine("RunTime " + elapsedTime);

Array.Sort(CurrentPopulation,

```

```

        delegate(CommonVars.adaptiveChromosome x, CommonVars.adaptiveChromosome y) { return
x.fit.makespan.CompareTo(y.fit.makespan); });

        AuxFuns.HandleExcelFile(repetitions, CurrentPopulation[0], probParams,
elapsedTime);

        //AuxFuns.HandleExcelFile(repetitions, filename, elapsedTime,
minSchDuration);
        for(int i=0;i<jobNum;i++)
            Console.Write(CurrentPopulation[0].SGSstat.startTimes[i]+"--");
            Console.WriteLine();

            for (int i = 0; i < jobNum; i++)
                Console.Write(CurrentPopulation[0].modeList[i] + "--");
                Console.WriteLine();
        }

        return resultSchedules;
    }
}

```



```

public class SingleObjectiveGA
{
    public CommonVars.adaptiveChromosome[] SingleObjGA(CommonVars.EmoData inputData,
CommonVars.problemParams probParams, CommonVars.adaptiveChromosome[] UnionPopulation)
    {
        Functions.AuxiliaryFunctions AuxFuns = new AuxiliaryFunctions();
        CommonVars.adaptiveChromosome[] CurrentPopulation = new
CommonVars.adaptiveChromosome[probParams.POP];
        try{
            int[] Fitness = CalcFitness(inputData, probParams, ref UnionPopulation);

            //Extension.Shuffle(UnionPopulation);
            int POP = probParams.POP;
            int newPopSize = 0;
            int[] FitnessFinal = new int[POP];
            Objectives obj = new Objectives();

            for (int i = 0; i < UnionPopulation.Length; i++)
                obj.calcRLI(ref inputData, probParams, ref UnionPopulation[i]);

            // for (int i = 0; i < UnionPopulation.Length; i++)
            //     Console.WriteLine("{0} -- {1}", UnionPopulation[i].fit.makespan,
UnionPopulation[i].Algo);

            Array.Sort(UnionPopulation,
                delegate(CommonVars.adaptiveChromosome x, CommonVars.adaptiveChromosome y) { return
x.fit.makespan.CompareTo(y.fit.makespan); });
            Array.Sort(Fitness);
            //do
            //{

                // int minFit = Fitness.Min();
                // int individual = Array.IndexOf(Fitness, minFit);
                // CurrentPopulation[newPopSize] = new
CommonVars.adaptiveChromosome();
                // CurrentPopulation[newPopSize].activityList = new
int[inputData.jobNum];
                //
                UnionPopulation[individual].activityList.CopyTo(CurrentPopulation[newPopSize].activityLis
t, 0);
                // FitnessFinal[newPopSize] = Fitness[individual];
                // CopyChromo(inputData, CurrentPopulation[newPopSize].activityList,
ref CurrentPopulation[newPopSize], UnionPopulation[individual]);
                // Fitness[individual] = Fitness.Max();
                // newPopSize++;

            //} while (newPopSize < POP);
            int individual = 0;
            do
            {
                CurrentPopulation[newPopSize] = new CommonVars.adaptiveChromosome();
                CurrentPopulation[newPopSize].activityList = new
int[inputData.jobNum];

                UnionPopulation[individual].activityList.CopyTo(CurrentPopulation[newPopSize].activityLis
t, 0);

                FitnessFinal[newPopSize] = Fitness[individual];

```

```

        CopyChromo(inputData, CurrentPopulation[newPopSize].activityList, ref
CurrentPopulation[newPopSize], UnionPopulation[individual]);
        newPopSize++;
        individual++;

    } while (newPopSize < POP);
    E_MO_RCPSP.OthersAlgos.RCPSP.SerialSGS_unscheduling schedule = new
OthersAlgos.RCPSP.SerialSGS_unscheduling();

    //int min = FitnessFinal.Min();

    //calc other objectives just to know...
}
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return CurrentPopulation;
}
public class SortByMakespan : IComparer<CommonVars.adaptiveChromosome>
{
    public int Compare(CommonVars.adaptiveChromosome x,
CommonVars.adaptiveChromosome y)
    {
        return x.fit.makespan.CompareTo(y.fit.makespan);
    }
}
public int[] CalcFitness(CommonVars.EmoData inputData, CommonVars.problemParams
probParams, ref CommonVars.adaptiveChromosome[] Population)
{
    int[] Fitness = new int[Population.Length];
    try
    {
        Functions.AuxiliaryFunctions AuxFuns = new AuxiliaryFunctions();
        List<CommonVars.adaptiveChromosome> PSOPopulationList = new
List<CommonVars.adaptiveChromosome>();
        List<CommonVars.adaptiveChromosome> GAPopulationList = new
List<CommonVars.adaptiveChromosome>();
        for (int i = 0; i < Population.Length; i++)
        {
            //calculate ES,LS mode specific therefore is executed after mode
assignment

            //for (int q = 0; q < inputData.jobNum; q++)
            //    for (int j = 0; j <
inputData.activities[q].relationships.Count(); j++)
            //        Console.WriteLine(q + "--"+
inputData.activities[q].relationships[j].ID + " -- "+
inputData.activities[q].relationships[j].lag[0].lag);
            AuxFuns.FloydWarshall(inputData, ref Population[i]);

            switch (Population[i].Algo)
            {
                case 0:
                {

```

```

        Fitness[i] = SA4Adaptive(inputData, probParams, ref
Population[i]);
        break;
    }
    case 1: PSOPopulationList.Add(Population[i]);
        break;
    case 2: GAPopulationList.Add(Population[i]);
        break;
    case 3:
    {
        Fitness[i] = SA4Adaptive(inputData, probParams, ref
Population[i]); // TS
        break;
    }
    default: Fitness[i] = -1;
        break;
    }
}
}
CommonVars.adaptiveChromosome[] PSOPopulation =
PSOPopulationList.ToArray();
CommonVars.adaptiveChromosome[] GAPopulation =
GAPopulationList.ToArray();
int[] PSOFitness = new int[PSOPopulation.Length];
if (PSOPopulation.Length >= 2)
    PSOFitness = PSOAdaptive(ref PSOPopulation, inputData, probParams);
else
{
    for (int count = 0; count < PSOPopulation.Count(); count++)
        PSOFitness[count] = SA4Adaptive(inputData, probParams, ref
PSOPopulation[count]);
}
int[] GAFitness = new int[GAPopulation.Length];
if (GAPopulation.Length >= 2)
    GAFitness = GA4Adaptive(ref GAPopulation, inputData, probParams);
else
{
    for (int count = 0; count < GAPopulation.Count(); count++)
        GAFitness[count] = SA4Adaptive(inputData, probParams, ref
GAPopulation[count]);
}
int PSOcount = 0;
int GAcount = 0;
for (int i = 0; i < Population.Length; i++)
{
    if (Population[i].Algo == 1)
    {
        Fitness[i] = PSOFitness[PSOcount];
        Array.Copy(PSOPopulation[PSOcount].activityList,
Population[i].activityList, Population[i].activityList.Length);
        Population[i].fit = new CommonVars.Fitness();
        Population[i].fit.makespan = Fitness[i];
        PSOcount++;
    }
    if (Population[i].Algo == 2)
    {
        Fitness[i] = GAFitness[GAcount];
    }
}

```

```

        Array.Copy(GAPopulation[GAccount].activityList,
Population[i].activityList, Population[i].activityList.Length);
        Array.Copy(GAPopulation[GAccount].modeList,
Population[i].modeList, Population[i].activityList.Length);
        CopyChromo(inputData, GAPopulation[GAccount].activityList, ref
Population[i], GAPopulation[GAccount]);
        GAccount++;
    }
}
}
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
return Fitness;
}

#region SA 4 Adaptive

public int SA4Adaptive(CommonVars.EmoData inputData, CommonVars.problemParams
probParams, ref CommonVars.adaptiveChromosome chromo)
{
    int Fitness_xbest = -1;
    try
    {
        Console.WriteLine("SA");
        int jobNum = inputData.jobNum;
        Functions.AuxiliaryFunctions AuxFuns = new
Functions.AuxiliaryFunctions();

        #region setup parameters
        int Neighborhoods = probParams.SAparams.Neighborhoods;
        int Chains = probParams.SAparams.Chains;
        int Steps = probParams.SAparams.Steps;
        int N0 = probParams.SAparams.N0;
        int h = probParams.SAparams.h;
        double T0max = probParams.SAparams.T0max;
        double a = probParams.SAparams.a;

        #endregion

        #region step 1: compute cp finish time

        int CP_FinishTime = inputData.activities[jobNum - 1].LS;

        //help print
        // Console.WriteLine("CP Finish Time: " + CP_FinishTime);
        #endregion

        #region step3 and 4 initialize variables - set best and current solutions

        int[] actList = new int[chromo.activityList.Length];
        Array.Copy(chromo.activityList, actList, chromo.activityList.Length);
        CommonVars.adaptiveChromosome actListChromo = new
CommonVars.adaptiveChromosome();
        CopyChromo(inputData, actList, ref actListChromo, chromo);

```

```

SerialSGS_unscheduling schedule = new SerialSGS_unscheduling();
int FinishTimeSerial = schedule.serialSGSu(inputData, probParams, ref
actListChromo);

int[] xbest = new int[jobNum];
Array.Copy(actList, xbest, jobNum);
CommonVars.adaptiveChromosome xbestChromo = new
CommonVars.adaptiveChromosome();
CopyChromo(inputData, xbest, ref xbestChromo, actListChromo);
////////////////////////////////////
int[] xcurrent = new int[jobNum];
Array.Copy(actList, xcurrent, jobNum);
CommonVars.adaptiveChromosome xcurrentChromo = new
CommonVars.adaptiveChromosome();
CopyChromo(inputData, xcurrent, ref xcurrentChromo, actListChromo);

Fitness_xbest = FinishTimeSerial; //fitness of initial chromo

int Fitness_xcurrent = FinishTimeSerial; //fitness of initial chromo

#endregion

#region step 6 ...main component
int Ntotal = 0;
for (int i = 0; i < Chains; i++)
{
    //help print
    //Console.WriteLine("Chain: " + i);
    //proposed value of param ToMax
    T0max = 0.2 * Fitness_xcurrent;
    double T = T0max;
    int Ns = N0;
    // int Ntot=0;
    for (int j = 0; j < Steps; j++)
    {
        //Console.WriteLine("Step: " + j);
        Ns = Ns * (1 + h * j); //h*s where s the step in the algo
        //Ntot += Ns; // dont look for more than Neighborhoods total

solutions

generated neighborhoods

        if (Ntotal < Neighborhoods) //stopping criterion max number of
        {
            for (int k = 0; k < Ns; k++)
            {
                Ntotal++;
                //Console.WriteLine("Neighborhood: " + k);
                //generate neighborhood of x0

                AuxFuns.GenerateNeighborhood(inputData, ref
actListChromo);

                //help print
                //for (int m = 0; m < jobNum; m++)
                // Console.WriteLine((actList[m]));
            }
        }
    }
}

```

```

//Console.WriteLine("");
//calculate fitness of generated act List

int Fitness_xnew = schedule.serialSGSu(inputData,
probParams, ref actListChromo);
//help print
//Console.WriteLine("Finish time: " + Fitness_xnew);
//calculate D= fitness of new solution - fitness of
current

int D = Fitness_xnew - Fitness_xcurrent;
if (D < 0)
{
    Array.Copy(actList, xcurrent, jobNum);

    CopyChromo(inputData, xcurrent, ref xcurrentChromo,
actListChromo);

    Fitness_xcurrent = Fitness_xnew;

    if (Fitness_xnew < Fitness_xbest)
    {
        Array.Copy(actList, xbest, jobNum);

        CopyChromo(inputData, xbest, ref xbestChromo,
actListChromo);

        Fitness_xbest = Fitness_xnew;
        if (Fitness_xbest == CP_FinishTime)
            break;
    }
}
else
{
    double P = Math.Pow(Math.E, -D / T);
    double Xrandom = GlobalVars.rand.NextDouble();
    //help print
    //Console.WriteLine("Value of P: " + P + " Value of
Xrandom: " + Xrandom);

    if (P > Xrandom)
    {
        Array.Copy(actList, xcurrent, jobNum);
        CopyChromo(inputData, xcurrent, ref
xcurrentChromo, actListChromo);

        Fitness_xcurrent = Fitness_xnew;
    }
}
}
}

T = a * T;
}
}

#endregion

#region step 7 explore neighborhood of x best
// Console.WriteLine("Exploring neighborhood of currently best
solution");

```

```

int Naverage = 10;
Array.Copy(xbest, actList, jobNum);
CopyChromo(inputData, actList, ref actListChromo, xbestChromo);

Array.Copy(actList, xcurrent, jobNum);
CopyChromo(inputData, xcurrent, ref xcurrentChromo, actListChromo);

Fitness_xcurrent = Fitness_xbest;

for (int k = 0; k < Naverage; k++)
{
    // Console.WriteLine("Neighborhood: " + k);
    //generate neighborhood of x0

    AuxFuns.GenerateNeighborhood(inputData, ref actListChromo);
    //help print
    //for (int m = 0; m < jobNum; m++)
    // Console.WriteLine((activityList[m]));
    //Console.WriteLine("");
    //calculate fitness of generated act List
    int Fitness_xnew = schedule.serialSGSu(inputData, probParams, ref
actListChromo);
    //help print
    //Console.WriteLine("Finish time: " + Fitness_xnew);
    //calculate D= fitness of new solution - fitness of current
    int D = Fitness_xnew - Fitness_xcurrent;
    if (D < 0)
    {
        Array.Copy(actList, xcurrent, jobNum);
        CopyChromo(inputData, xcurrent, ref xcurrentChromo,
actListChromo);
        Fitness_xcurrent = Fitness_xnew;

        if (Fitness_xnew < Fitness_xbest)
        {
            Array.Copy(actList, xbest, jobNum);
            CopyChromo(inputData, xbest, ref xbestChromo, actListChromo);
            Fitness_xbest = Fitness_xnew;
            if (Fitness_xbest == CP_FinishTime)
                break;
        }
    }
}

}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}

//help print
//Console.WriteLine("Best Solution");
//for (int j = 0; j < jobNum; j++)
// Console.WriteLine((xbest[j]));
//Console.WriteLine("");

```

```

// Console.WriteLine("Finish time: " + Fitness_xbest);
chromo.fit = new CommonVars.Fitness();
chromo.fit.makespan = Fitness_xbest;
return Fitness_xbest;
#endregion
}
//copy all data from chromo except actList that is given using the xbest array
public void CopyChromo(CommonVars.EmoData inputData, int[] xbest, ref
CommonVars.adaptiveChromosome xbestChromo, CommonVars.adaptiveChromosome chromo)
{
    try
    {
        xbestChromo.activityList = new int[inputData.jobNum];
        Array.Copy(xbest, xbestChromo.activityList, xbest.Length);

        xbestChromo.modeList = new int[chromo.modeList.Length];
        Array.Copy(chromo.modeList, xbestChromo.modeList,
chromo.modeList.Length);

        xbestChromo.Algo = chromo.Algo;
        xbestChromo.SGS = chromo.SGS;

        xbestChromo.SGSstat = new CommonVars.SGSstatus();
        xbestChromo.SGSstat.uStep = chromo.SGSstat.uStep;

        xbestChromo.SGSstat.startTimes = new int[inputData.jobNum];
        if (chromo.SGSstat.startTimes != null)
            Array.Copy(chromo.SGSstat.startTimes, xbestChromo.SGSstat.startTimes,
chromo.SGSstat.startTimes.Length);

        xbestChromo.SGSstat.finishTimes = new int[inputData.jobNum];
        if (chromo.SGSstat.finishTimes != null)
            Array.Copy(chromo.SGSstat.finishTimes,
xbestChromo.SGSstat.finishTimes, chromo.SGSstat.finishTimes.Length);
        if (chromo.SGSstat.scheduledSet != null)
        {
            List<int> scheduledSet = new List<int>(chromo.SGSstat.scheduledSet);
            xbestChromo.SGSstat.scheduledSet = scheduledSet;
        }
        xbestChromo.SGSstat.resAvailDynamic = new
Commons.CommonVars.resourceAvail();
        xbestChromo.SGSstat.resAvailDynamic.renResAvail = new
System.Collections.Generic.List<Commons.CommonVars.resource>();

        if (chromo.SGSstat.resAvailDynamic != null)
        {
            if (chromo.SGSstat.resAvailDynamic.renResAvail != null)
                foreach (Commons.CommonVars.resource res in
chromo.SGSstat.resAvailDynamic.renResAvail)
                {
                    Commons.CommonVars.resource copyres = new
Commons.CommonVars.resource();
                    copyres.resID = res.resID;
                    copyres.resCalMatrix = new int[res.resCalMatrix.Length];
                    if (res.resCalMatrix != null)
                        Array.Copy(res.resCalMatrix, copyres.resCalMatrix,
res.resCalMatrix.Length);
                    xbestChromo.SGSstat.resAvailDynamic.renResAvail.Add(copyres);

```



```

        }

        xbestChromo.SGSstat.resAvailDynamic.nonrenResAvail = new
System.Collections.Generic.List<Commons.CommonVars.resource>();
        if (chromo.SGSstat.resAvailDynamic.nonrenResAvail != null)
            foreach (Commons.CommonVars.resource res in
chromo.SGSstat.resAvailDynamic.nonrenResAvail)
            {
                Commons.CommonVars.resource copyres = new
Commons.CommonVars.resource();
                copyres.resID = res.resID;
                copyres.resCalMatrix = new int[res.resCalMatrix.Length];
                Array.Copy(res.resCalMatrix, copyres.resCalMatrix,
res.resCalMatrix.Length);

                xbestChromo.SGSstat.resAvailDynamic.nonrenResAvail.Add(copyres);
            }
        xbestChromo.SGSstat.resAvailDynamic.dbResAvail = new
System.Collections.Generic.List<Commons.CommonVars.resource>();
        if (chromo.SGSstat.resAvailDynamic.dbResAvail != null)
            foreach (Commons.CommonVars.resource res in
chromo.SGSstat.resAvailDynamic.dbResAvail)
            {
                Commons.CommonVars.resource copyres = new
Commons.CommonVars.resource();
                copyres.resID = res.resID;
                copyres.resCalMatrix = new int[res.resCalMatrix.Length];
                if (res.resCalMatrix != null)
                    Array.Copy(res.resCalMatrix, copyres.resCalMatrix,
res.resCalMatrix.Length);
                xbestChromo.SGSstat.resAvailDynamic.dbResAvail.Add(copyres);
            }
    }

    xbestChromo.SGSstat.ES = new int[inputData.jobNum];
    if (chromo.SGSstat.ES != null)
        Array.Copy(chromo.SGSstat.ES, xbestChromo.SGSstat.ES,
chromo.SGSstat.ES.Length);
    xbestChromo.SGSstat.LS = new int[inputData.jobNum];
    if (chromo.SGSstat.LS != null)
        Array.Copy(chromo.SGSstat.LS, xbestChromo.SGSstat.LS,
chromo.SGSstat.LS.Length);

    if (chromo.SGSstat.strongPredsSet != null)
    {
        System.Collections.Generic.List<List<int>> strongPreds = new
List<List<int>>(chromo.SGSstat.strongPredsSet);
        xbestChromo.SGSstat.strongPredsSet = strongPreds;
    }
    if (chromo.SGSstat.strongSucsSet != null)
    {
        System.Collections.Generic.List<List<int>> strongSucs = new
List<List<int>>(chromo.SGSstat.strongSucsSet);
        xbestChromo.SGSstat.strongSucsSet = strongSucs;
    }

    xbestChromo.SGSstat.jSelected = chromo.SGSstat.jSelected;

```

```

        xbestChromo.SGSstat.tSelected = chromo.SGSstat.tSelected;

        xbestChromo.SGSstat.initialDistanceMatrix = new int[inputData.jobNum,
inputData.jobNum];
        if (chromo.SGSstat.initialDistanceMatrix != null)
            Array.Copy(chromo.SGSstat.initialDistanceMatrix,
xbestChromo.SGSstat.initialDistanceMatrix, chromo.SGSstat.initialDistanceMatrix.Length);

        xbestChromo.SGSstat.distanceMatrix = new int[inputData.jobNum,
inputData.jobNum];
        if (chromo.SGSstat.distanceMatrix != null)
            Array.Copy(chromo.SGSstat.distanceMatrix,
xbestChromo.SGSstat.distanceMatrix, chromo.SGSstat.distanceMatrix.Length);

        //added for fitness
        xbestChromo.fit = new CommonVars.Fitness();
        if (chromo.fit != null)
        {
            xbestChromo.fit.rank = chromo.fit.rank;
            xbestChromo.fit.makespan = chromo.fit.makespan;
            xbestChromo.fit.RLI = chromo.fit.RLI;
            xbestChromo.fit.Cost = chromo.fit.Cost;
            xbestChromo.fit.Robustness = chromo.fit.Robustness;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
#endregion
#region PSO 4 Adaptive
private int[] PSOAdaptive(ref CommonVars.adaptiveChromosome[] PSO_Swarm,
CommonVars.EmoData inputData, CommonVars.problemParams probParams)
{
    Console.WriteLine("PSO");

    int jobNum = PSO_Swarm[0].activityList.Length;

    SerialSGS_unscheduling schedule = new SerialSGS_unscheduling();

    Functions.AuxiliaryFunctions AuxFuns = new Functions.AuxiliaryFunctions();

    #region parameters
    double[][] bestOfparticle = new double[PSO_Swarm.Length][]; //Swarm Size
Lines x JobNum cols
    double[] bestOfswarm = new double[jobNum]; // the best chromo converted to
double in [0,1]

    int[] FitnessSwarm = new int[PSO_Swarm.Length];
    double weight = probParams.PSOparams.weight;
    double c1 = probParams.PSOparams.c1;
    double c2 = probParams.PSOparams.c2;

    double r1 = -1; //random value uniformly drawn from [0,1] --> already uniform
in C# Random
    double r2 = -1; //random value uniformly drawn from [0,1]

```

```

double[][] velocity = new double[PSO_Swarm.Length][];
double[][] x = new double[PSO_Swarm.Length][]; //particles corresponding to
initial chromos
int[] FitBestOfParticle = new int[PSO_Swarm.Length];
int FitBestOfSwarm = -1;

BestOfPSO_InitialValues( PSO_Swarm, ref bestOfparticle, ref bestOfswarm, ref
FitBestOfParticle, ref FitBestOfSwarm, inputData, probParams);

//initialize particles with current chromos converted to particles by
normalization
for (int i = 0; i < PSO_Swarm.Length; i++)
{
    x[i] = new double[jobNum];
    Array.Copy(bestOfparticle[i], x[i], jobNum);
}
//initialize velocity to ???
for (int i = 0; i < PSO_Swarm.Length; i++)
{
    velocity[i] = new double[jobNum];
    for (int j = 0; j < jobNum; j++)
        velocity[i][j] = GlobalVars.rand.NextDouble();
}

#endregion

#region PSO repetitive process
int GlobalCount = 0;
int TotalCount = 0;
do
{
    for (int part = 0; part < PSO_Swarm.Length; part++)
    {
        int[] actList = PSO_Swarm[part].activityList;
        // velocity=w * vi + c1 * U(0 , 1) * (pi - xi) + c2 * U(0 , 1) * (g -
xi) where pi is the best previous position of the particle and g is the best found
position within the swarm so far
        for (int i = 0; i < jobNum; i++)
        {
            r1 = GlobalVars.rand.NextDouble();
            r2 = GlobalVars.rand.NextDouble();

            velocity[part][i] = weight * velocity[part][i] + c1 * r1 *
(bestOfparticle[part][i] - x[part][i]) + c2 * r2 * (bestOfswarm[i] - x[part][i]);
            x[part][i] = x[part][i] + velocity[part][i];
        }

        #region mapping
        //x vector is ordered and KEYS are ordered accordingly this way we
get the new actList
        int[] chromoList = new int[jobNum];
        double[] xTemp = new double[jobNum];
        Array.Copy(x[part], xTemp, jobNum);

        for (int i = 0; i < jobNum; i++)
            chromoList[i] = i;

        Array.Sort(xTemp, chromoList);
    }
}

```

```

        int temp = chromoList[0];
        int pos = Array.IndexOf(chromoList, 0);
        chromoList[pos] = temp;
        chromoList[0] = 0;
        temp = chromoList[jobNum - 1];
        pos = Array.IndexOf(chromoList, jobNum - 1);
        chromoList[pos] = temp;
        chromoList[jobNum - 1] = jobNum - 1; ;
        //addition for modes

        //send back updated chromo
        Array.Copy(chromoList, PSO_Swarm[part].activityList, jobNum);
        CopyChromo(inputData, PSO_Swarm[part].activityList, ref
PSO_Swarm[part], PSO_Swarm[part]);
        //addition for modes

        //calculate fitness
        int Fitness_xnew = schedule.serialSGSu(inputData, probParams, ref
PSO_Swarm[part]);

        //update Local Best
        if (FitBestOfParticle[part] > Fitness_xnew)
        {
            Array.Copy(x[part], bestOfparticle[part], jobNum);

            FitBestOfParticle[part] = Fitness_xnew;
        }
        //update Global Best
        if (FitBestOfSwarm > Fitness_xnew)
        {
            Array.Copy(x[part], bestOfswarm, jobNum);
            FitBestOfSwarm = Fitness_xnew;
        }
        else GlobalCount++;
    }
    TotalCount++;
    #endregion
#endregion
} while (GlobalCount < 100 * PSO_Swarm.Length && TotalCount < 1000);

//help print
// Console.WriteLine("Best of Swarm:{0}", FitBestOfSwarm);
return (FitBestOfParticle);
}

public void BestOfPSO_InitialValues( CommonVars.adaptiveChromosome[] PSO_Swarm,
ref double[][] BestOfParticle,ref double[] BestOfSwarm, ref int[] FitBestOfParticle, ref
int FitBestOfSwarm, CommonVars.EmoData inputData, CommonVars.problemParams probParams)
{
    int jobNum = PSO_Swarm[0].activityList.Length;
    SerialSGS_unscheduling schedule = new SerialSGS_unscheduling();
    Functions.AuxiliaryFunctions AuxFuns = new Functions.AuxiliaryFunctions();

    for (int i = 0; i < PSO_Swarm.Length; i++) //for each particle

```

```

    {
        int[] chromo = new int[jobNum];
        chromo = PSO_Swarm[i].activityList;
        CommonVars.adaptiveChromosome actListChromo = new
CommonVars.adaptiveChromosome();
        CopyChromo(inputData, chromo, ref actListChromo, PSO_Swarm[i]);

        //mapping to particle
        BestOfParticle[i] = new double[jobNum];
        for (int j = 0; j < jobNum; j++) //convert chromo to particle by
normalizing in [0,1]
            BestOfParticle[i][j] = Convert.ToDouble(chromo[j]) /
Convert.ToDouble((jobNum - 1));

        FitBestOfParticle[i] = schedule.serialSGSu(inputData, probParams, ref
actListChromo);

        if (i == 0)
        {
            FitBestOfSwarm = FitBestOfParticle[i];
            Array.Copy(BestOfParticle[i], BestOfSwarm, jobNum);
        }
        if (FitBestOfParticle[i] <= FitBestOfSwarm)
        {
            Array.Copy(BestOfParticle[i], BestOfSwarm, jobNum);
            FitBestOfSwarm = FitBestOfParticle[i];
        }
    }

}

#endregion 4 Adaptive
#region GA 4 Adaptive
private int[] GA4Adaptive(ref CommonVars.adaptiveChromosome[]
GAPopulation,CommonVars.EmoData inputData, CommonVars.problemParams probParams)
{
    Console.WriteLine("GA");
    int[] Fitness = new int[GAPopulation.Length];

    Fitness = ActList_M_GA(ref GAPopulation, inputData, probParams);

    return (Fitness);
}
private int[] ActList_M_GA(ref CommonVars.adaptiveChromosome[] GAPopulation,
CommonVars.EmoData inputData, CommonVars.problemParams probParams)
{
    int POP = 0;
    if (GAPopulation.Length % 2 == 0)
        POP = GAPopulation.Length;
    else POP = GAPopulation.Length - 1;
    probParams.sGParams.POP = POP;
    int numOfRepetitions = probParams.sGParams.numOfRepetitions;

    Functions.AuxiliaryFunctions AuxFuns = new Functions.AuxiliaryFunctions();

```

```

SerialSGS_unscheduling schedule = new SerialSGS_unscheduling();

int[] Fitness = new int[GAPopulation.Length];
try
{
    // Console.WriteLine("GA");
    for (int repetitions = 0; repetitions < numOfRepetitions; repetitions++)
    {
        CommonVars.adaptiveChromosome[] InPopulation = new
CommonVars.adaptiveChromosome[GAPopulation.Length];
        Array.Copy(GAPopulation, InPopulation, GAPopulation.Length);

        //calculate fitness

        int i = 0;
        for (int pop = 0; pop < GAPopulation.Length; pop++)
        {
            Fitness[i] = schedule.serialSGSu(inputData, probParams, ref
GAPopulation[pop]);
            i++;
        }

        //////////////////////////////////////

        int NumOfGenerations = 0;
        int generMin = 0;
        int jobNum = inputData.jobNum;
        while (NumOfGenerations < probParams.GEN)
        {
            NumOfGenerations++;

            //Crossover
            CommonVars.adaptiveChromosome[] ChildrenPopulation = new
CommonVars.adaptiveChromosome[POP];
            ChildrenPopulation = AuxFuns.CrossoverM_act4Ad(InPopulation,
probParams, inputData);
            //Mutation
            ChildrenPopulation = AuxFuns.MutationM4Ad(ChildrenPopulation,
probParams, inputData);

            //Fitness of children
            int[] FitnessChildren = new int[probParams.sGParams.POP];
            int j = 0;
            foreach (CommonVars.adaptiveChromosome chromo in
ChildrenPopulation)
            {
                if (ChildrenPopulation[j] != null)
                {
                    FitnessChildren[j] = schedule.serialSGSu(inputData,
probParams, ref ChildrenPopulation[j]);
                }
                else FitnessChildren[j] = FitnessChildren.Max() + 1000;
                j++;
            }

            //Selection

```

```

        CommonVars.adaptiveChromosome[] NewGeneration = new
CommonVars.adaptiveChromosome[POP];
        int[] FitnessFinal = new int[POP];
        AuxFuns.SelectionM(InPopulation, ChildrenPopulation, Fitness,
FitnessChildren, ref NewGeneration, ref FitnessFinal, probParams, inputData);

        //Set new generation as current population and repeat
NewGeneration.CopyTo(InPopulation, 0);
NewGeneration.CopyTo(GAPopulation, 0);
if (GAPopulation.Length % 2 != 0)
    GAPopulation[GAPopulation.Length - 1] = GAPopulation[0];

FitnessFinal.CopyTo(Fitness, 0);

//help step

int min = FitnessFinal.Min();

if (NumOfGenerations == 1) generMin = min;
if (generMin > min) generMin = min;

}
int minSchDuration = generMin;

////////////////////////////////////

    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}

return (Fitness);

}

#endregion
}

```

```

class Pareto
{
    public CommonVars.adaptiveChromosome[] ParetoGA(CommonVars.EmoData inputData,
CommonVars.problemParams probParams, CommonVars.adaptiveChromosome[] UnionPopulation)
    {
        Functions.AuxiliaryFunctions AuxFuns = new AuxiliaryFunctions();
        E_MO_RCPSP.OthersAlgos.RCPSP.SerialSGS_unscheduling schedule = new
OthersAlgos.RCPSP.SerialSGS_unscheduling();
        E_MO_RCPSP.OurAlgos.SingleObjectiveGA single = new SingleObjectiveGA();
        Objectives obj = new Objectives();
        NodDominatedSort nonDomSort =new NodDominatedSort();

        CommonVars.adaptiveChromosome[] CurrentPopulation = new
CommonVars.adaptiveChromosome[probParams.POP];

        int K = probParams.objectives.Count();

        //CALCULATE FITNESS VECTOR
        for (int i = 0; i < UnionPopulation.Length; i++)
        {
            CommonVars.adaptiveChromosome tempChromo = new
CommonVars.adaptiveChromosome();
            tempChromo = UnionPopulation[i];
            obj.calcRLI(ref inputData, probParams, ref tempChromo);
            UnionPopulation[i] = tempChromo;
            //add the rest of the objectives calculations HERE
        }

        //NON DOMINATED SORT

        List<CommonVars.adaptiveChromosome>[]
tempDomLevel=nonDomSort.calcNonDominatedSort(inputData, probParams, ref UnionPopulation);
        List<CommonVars.adaptiveChromosome>[] DomLevel = new
List<CommonVars.adaptiveChromosome>[tempDomLevel.Length];
        Array.Copy(tempDomLevel, DomLevel, tempDomLevel.Length);

        int emptySlots = CurrentPopulation.Length;
        int cur = 0;
        int dom = 0;
        do
        {
            for (int k = 0 + cur; k < cur + DomLevel[dom].Count; k++)
            {
                CurrentPopulation[k] = new CommonVars.adaptiveChromosome();

                CurrentPopulation[k].activityList = new int[DomLevel[dom][k -
cur].activityList.Length];
                Array.Copy(DomLevel[dom][k - cur].activityList,
CurrentPopulation[k].activityList, DomLevel[dom][k - cur].activityList.Length);
                single.CopyChromo(inputData, CurrentPopulation[k].activityList,
ref CurrentPopulation[k], DomLevel[dom][k - cur]);
            }
            cur += DomLevel[dom].Count;
            emptySlots -= DomLevel[dom].Count;

            dom++;
        }
    }
}

```



```

    } while (DomLevel[dom].Count <= emptySlots && dom < DomLevel.Length) ;

    //CLOSENESS
    //get the domination level that didnt fit in Current Population and sort it
    using Closeness function then add it to current population
    if (emptySlots > 0)
    {
        CommonVars.adaptiveChromosome[] BigCurPopulation = new
CommonVars.adaptiveChromosome[DomLevel[dom].Count];
        for (int k = 0; k < DomLevel[dom].Count; k++)
        {
            BigCurPopulation[k] = new CommonVars.adaptiveChromosome();
            BigCurPopulation[k].activityList = new
int[DomLevel[dom][k].activityList.Length];
            Array.Copy(DomLevel[dom][k].activityList,
BigCurPopulation[k].activityList, DomLevel[dom][k].activityList.Length);
            single.CopyChromo(inputData, BigCurPopulation[k].activityList, ref
BigCurPopulation[k], DomLevel[dom][k]);
        }
        CommonVars.adaptiveChromosome[] toBeAddedPop =
nonDomSort.SortByCloseness(BigCurPopulation, probParams, emptySlots);

        for (int k = 0; k < emptySlots; k++)
        {
            CurrentPopulation[k + cur] = new CommonVars.adaptiveChromosome();

            CurrentPopulation[k + cur].activityList = new
int[toBeAddedPop[k].activityList.Length];
            Array.Copy(toBeAddedPop[k].activityList, CurrentPopulation[k +
cur].activityList, toBeAddedPop[k].activityList.Length);
            single.CopyChromo(inputData, CurrentPopulation[k + cur].activityList,
ref CurrentPopulation[k + cur], toBeAddedPop[k]);
        }
    }
    return CurrentPopulation;
}
public class SortByRank : IComparer<CommonVars.adaptiveChromosome>
{
    public int Compare(CommonVars.adaptiveChromosome x,
CommonVars.adaptiveChromosome y)
    {
        return x.fit.rank.CompareTo(y.fit.rank);
    }
}
}

```



# Appendix C

## Ms Project 2013 – Add In for Multi-Objective Resource Constrained Project Scheduling

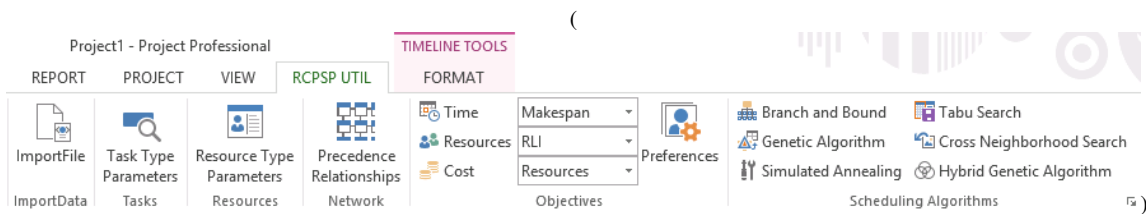


Fig. C.1 Ribbon styled toolbox for EMO- RCPSP

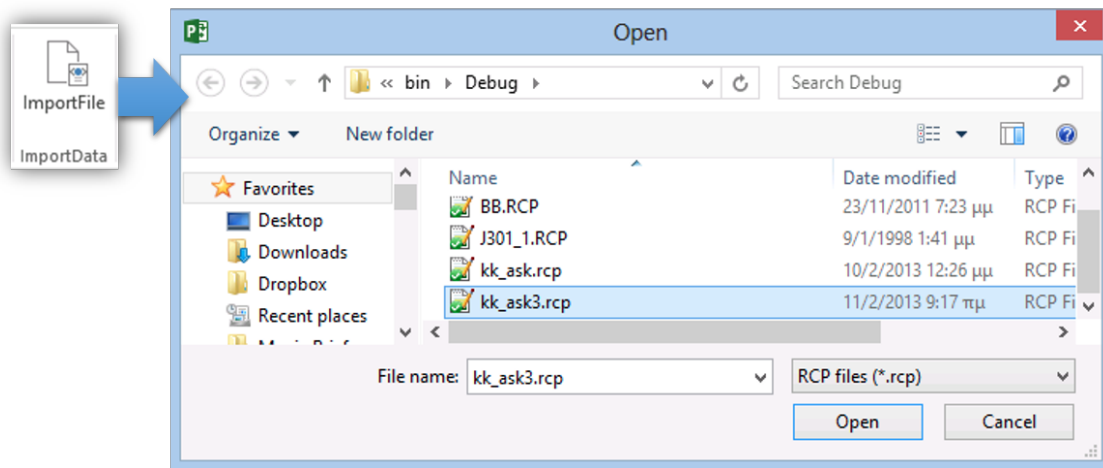
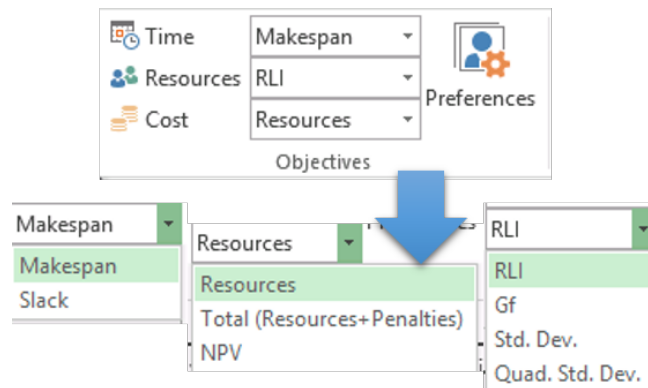


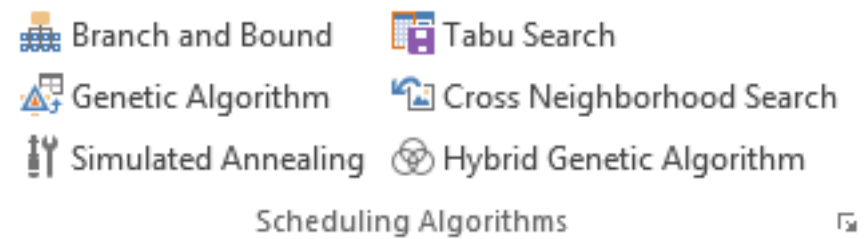
Fig. C.2 Import text files formatted as in PSPLib to run experiments



**Fig. C.3** Set up additional properties to handle multiple modes of execution, non renewable resource type and maximal time lags



**Fig. C.4** Define one or more optimisation objectives



**Fig. C.5** Select execution algorithm to generate solutions - schedules

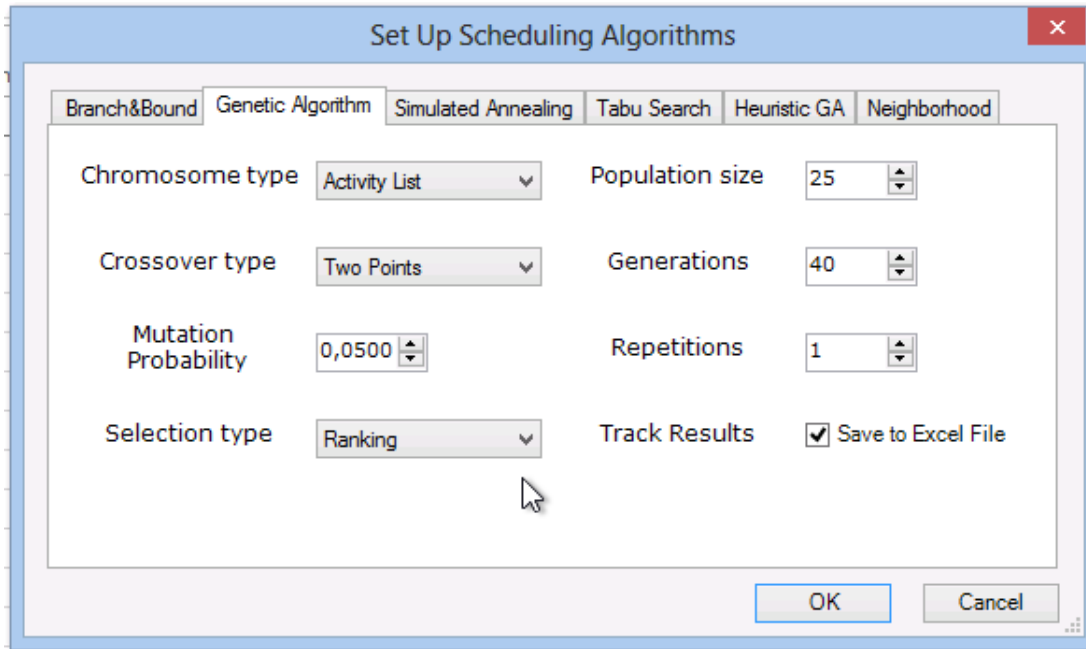


Fig. C.6 Set up parameters for the genetic algorithm (auxiliary algorithm)

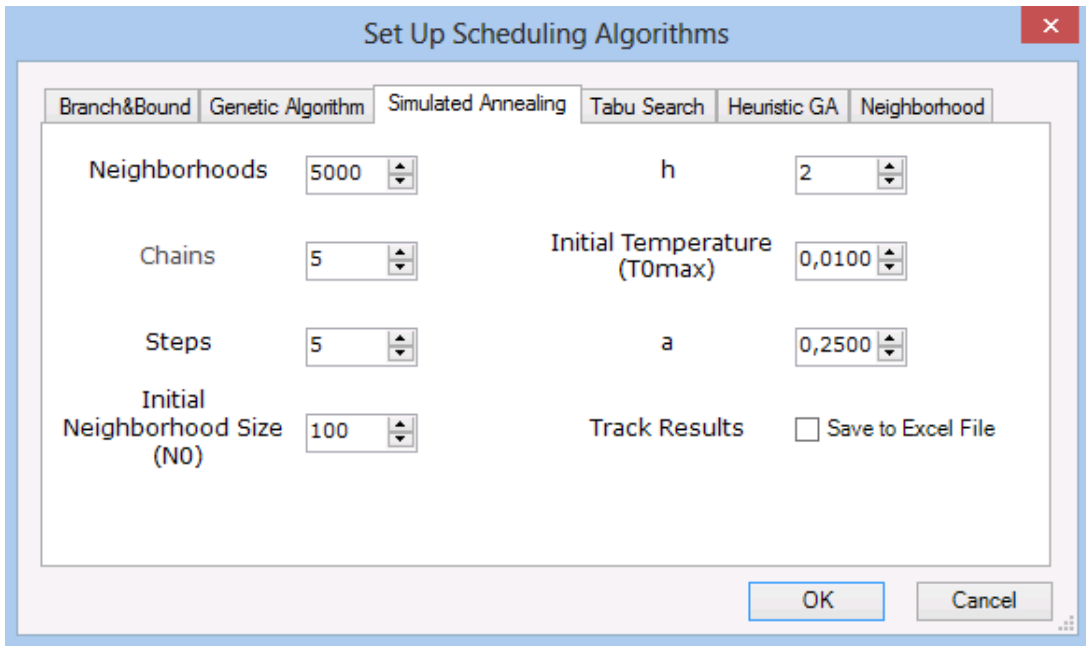


Fig. C.7 Set up parameters for the simulated annealing algorithm (auxiliary algorithm)

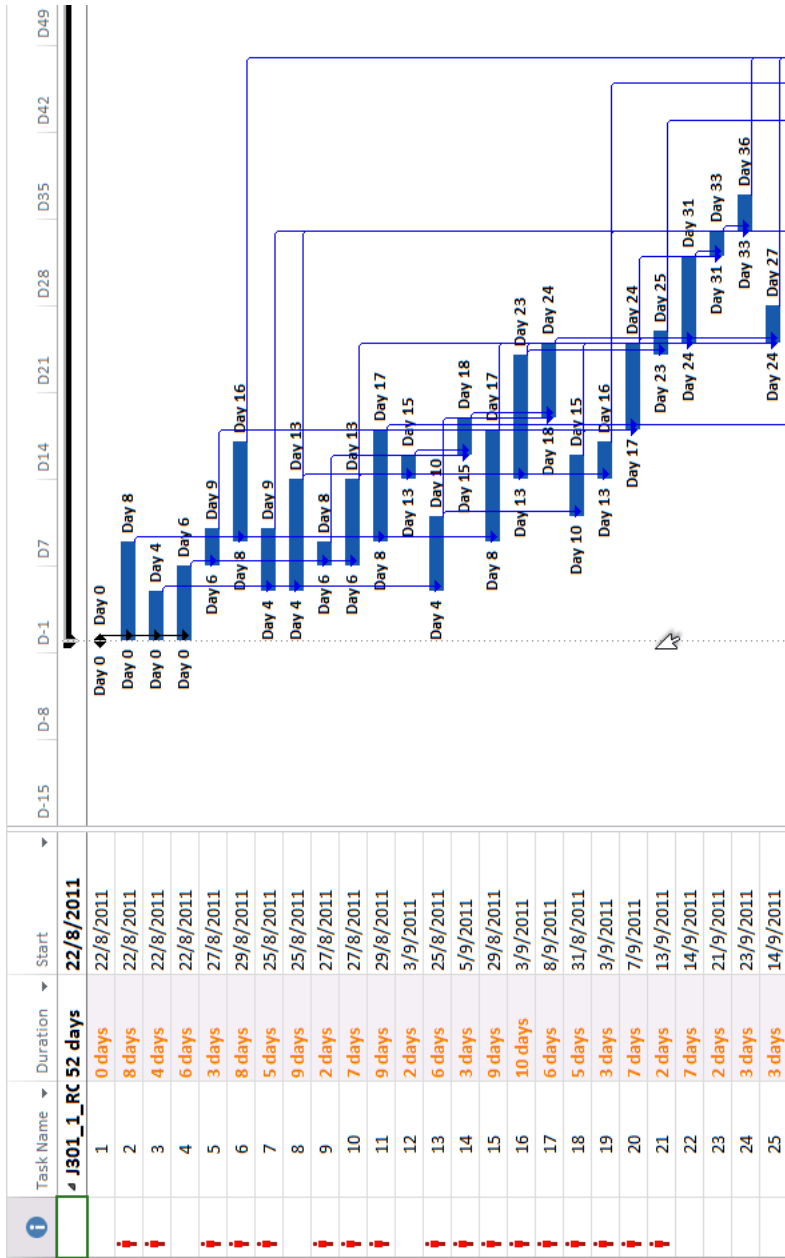


Fig. C.8 The given scheduled as was imported from J301\_1.rcp file from PSPLib

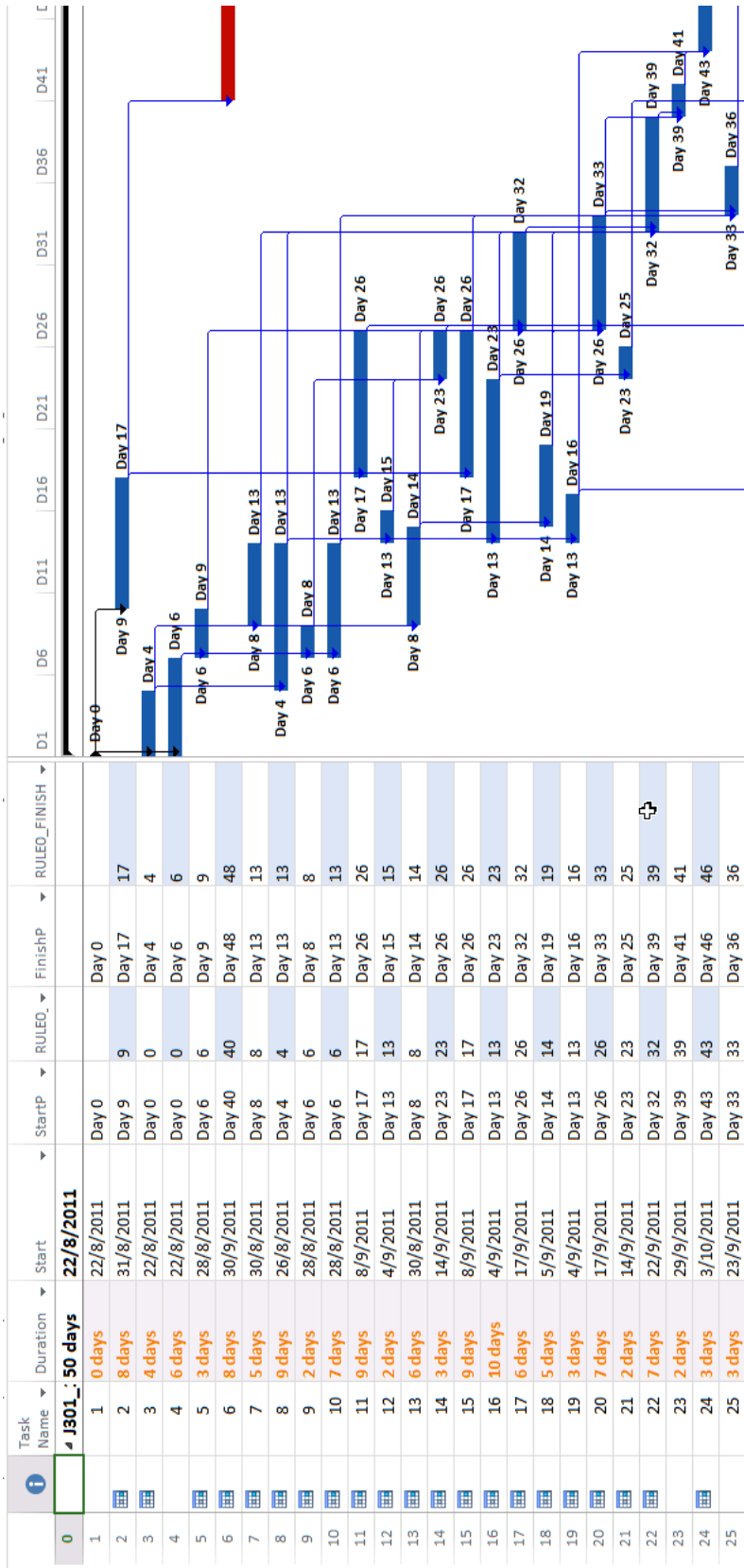
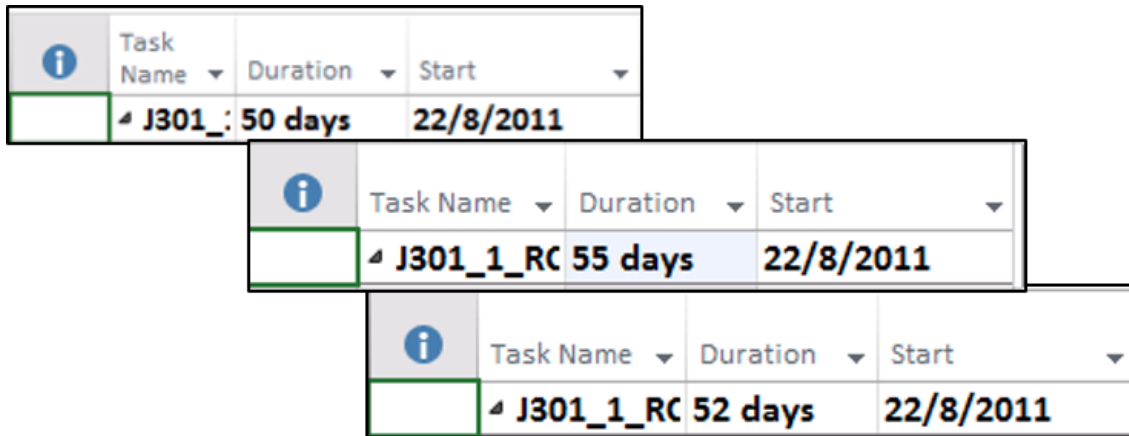


Fig. C.9 Schedule generated by the proposed algorithm - note that the "As Soon As Possible" constraint has been changed to "Must Start On"

Resource Name	Type	Material	Initials	Group	Max.	Std. Rate	Ovt. Rate	Cost/Use	Accrue	Base
0	Work		0		1.200%	\$0,00/hr	\$0,00/hr	\$0,00	Prorated	GENERIC RCPSP
1	Work		1		1.300%	\$0,00/hr	\$0,00/hr	\$0,00	Prorated	GENERIC RCPSP
2	Work		2		400%	\$0,00/hr	\$0,00/hr	\$0,00	Prorated	GENERIC RCPSP
3	Work		3		1.200%	\$0,00/hr	\$0,00/hr	\$0,00	Prorated	GENERIC RCPSP

Fig. C.10 Resource availabilities - the overallocated resources are marked with red



The figure displays three overlapping screenshots of a task list table. Each screenshot shows a table with three columns: Task Name, Duration, and Start. The first screenshot shows a task named 'J301\_' with a duration of 50 days and a start date of 22/8/2011. The second screenshot shows a task named 'J301\_1\_RC' with a duration of 55 days and a start date of 22/8/2011. The third screenshot shows a task named 'J301\_1\_RC' with a duration of 52 days and a start date of 22/8/2011.

Task Name	Duration	Start
J301_	50 days	22/8/2011

Task Name	Duration	Start
J301_1_RC	55 days	22/8/2011

Task Name	Duration	Start
J301_1_RC	52 days	22/8/2011

Fig. C.11 Duration of proposed scheduled compared to the results of MS Project's levelling option



## Glossary

**Feasible Schedule** A schedule that satisfies all the given constraints, like precedence relationships and resource availabilities.

**Regular Objective** The objective functions is monotone non-decreasing

**Non regular Objective** The objective functions is not monotone non-decreasing.

**Schedule Generation Scheme (SGS)** Constructive heuristics consisting of two major components, the scheduling scheme and the priority rule. The scheduling scheme determines the way in which a feasible schedule is constructed by assigning starting times to the different activities.

**Serial Schedule Generation Scheme (s-SGS)** A SGS that sequentially adds activities to the schedule until a feasible complete schedule is obtained. In each iteration, the next activity in the priority list is chosen and for that activity the first possible starting time is assigned such that no precedence or resource constraint is violated.

**Parallel Schedule Generation Scheme (p-SGS)** The parallel scheduling scheme iterates over the different decision points at which activities can be added to the schedule, thus it does time incrementation. These decision points correspond with the completion times of already scheduled activities and thus at most  $n$  decision points need to be considered in the parallel scheduling scheme. At each decision point, the unscheduled activities whose predecessors have completed are considered in the order of the priority list and are scheduled on the condition that no resource conflict originates at that time instant.

**Baseline schedule** The baseline schedule specifies for each activity the precedence and resource feasible start and completion dates, the amounts of the various resource types that will be needed during each time period and as a result the corresponding budget required for the execution of the project. It is a snapshot of how the project should be executed.

**Proactive schedule** It is a baseline schedule developed before starting the project's execution. It is also called preschedule, predictive schedule, etc.

**Reactive schedule** Reactive scheduling is about the revision and re-optimisation of the baseline schedule after one or more unexpected events have occurred. The goal is generate a new optimal schedule that will be as close as possible to the baseline.

**Semi-active schedule** Feasible schedules obtained by sequencing activities as early as possible. In a semi-active schedule no activity can be started earlier without altering the precedences.

**Active schedule** Feasible schedules in which no activity could be started earlier without delaying some other activity or breaking a precedence constraint.

**Non-delay schedules** Feasible schedules in which no resource is kept idle when it could start processing some activity.

**Makespan** The project's duration calculated as the finish time of the dummy sink activity that represents the project's finish.

**Preemption** Activity splitting, implies that the processing of an activity may be interrupted and resumed at a later time (preempt-resume). are available on a period-by-period basis. Only the total amount of resource used within each period is constrained.

**Nonrenewable resources** are available on a total project basis, with a limited consumption availability for the entire project.

**Doubly-constrained resources** are constrained per period as well as for the whole project.

**Partially (non)renewable resources** are resources whose availability is defined for a specific time interval (subset of periods). It is a generalisation of the above resource types and can be used to define both renewable and non renewable resources using a single resource type.

**Minimal and maximal lags** Minimal time lags in a *FS* relation introduce a time period  $t$  between the finish time of activity  $i$  and the start time of activity  $j$ . Allowing negative minimal time lags implies that the corresponding activities may overlap. Similarly maximal time lags in a *FS* relation, introduce a maximum time period  $t$  between the finish time of activity  $i$  and the starting time of activity  $j$ . A release date is a minimal finish to start time lag between the dummy source and the under question activity  $j$  and a deadline is a maximal finish to finish time lag between the dummy source activity and activity  $j$ .