Algorithms for Optimal Project Scheduling

by

Thomas S. Kyriakidis

A Dissertation

Submitted in partial fulfilment of the Requirements

For the degree of Doctor of Philosophy

Department of Engineering Informatics and Telecommunications

University of Western Macedonia

Karamanli & Lygeris Street

Kozani 50100, Greece

January 2013

Algorithms for Optimal Project Scheduling

by

Thomas S. Kyriakidis

A Dissertation submitted in partial fulfilment of the Requirements for the degree of Doctor of Philosophy

Advisory Committee

Supervisor : Assoc. Prof. Michael C. Georgiadis

Members : Prof. Andreas Georgiou

Assist. Prof. Konstantinos Stergiou

Department of Engineering Informatics and Telecommunications

University of Western Macedonia

Karamanli & Lygeris Street

Kozani 50100, Greece

January 2013

Approved by the Examination Committee Members:

First Member (Supervisor)	Assoc. Prof. Michael Georgiadis Department of Informatics and Telecommunications Engineering University of Western Macedonia
Second Member	Prof. Andreas Georgiou Department of Business Administration University of Macedonia
Third Member	Assist. Prof. Konstantinos Stergiou Department of Informatics and Telecommunications Engineering University of Western Macedonia
Fourth Member	Prof. Theodoros Chadjipadelis Department of Political Sciences Aristotle University of Thessaloniki
Fifth Member	Prof. Patroklos Georgiadis Department of Mechanical Engineering Aristotle University of Thessaloniki
Sixth Member	Assoc. Prof. Nikolaos Samaras Department of Applied Informatics University of Macedonia
Seventh Member	Assist. Prof. Georgios Kozanidis Department of Mechanical Engineering University of Thessalu

Dedicated to my wife Eleni and my parents...

ABSTRACT

Project scheduling plays a vital role in project management, and constitutes one of the most important directions in both research and practice in the Operational Research (OR) field. During the last decades, the Resource-Constrained Project Scheduling Problem (RCPSP) has become the most challenging standard problem of project scheduling in the OR literature. The RCPSP involves the construction of a *precedence* and *resource* feasible time schedule which identifies the starting and completion times of *activities*, under a specific *objective*. Several variations of the RCPSP exist that represent different practical problems with different objectives, resource types, more than one way (mode) to execute an activity, generalised precedence relations for activities, etc. The RCPSP and its variants belong to the class of strongly NP-hard problems and a number of solution methods, both *exact* and *approximate* have been proposed in the literature.

Scheduling is also a critical issue in process operations. The process scheduling problem consists of determining the most efficient way to produce a set of products in a time horizon given a set of processing recipes and limited resources. The activities to be scheduled usually take place in multiproduct and multipurpose plants, in which a wide variety of different products can be manufactured via the same recipe or different recipes by sharing limited resources, such as equipment, material, time, and utilities. The common problem features, such as required resource types, precedence relations and initial/target inventories, suggest that exchanging solution techniques between the two research fields is both possible and useful.

The process scheduling industry is driven by the substantial advances of related modelling and solution techniques, as well as the rapidly growing computational power. On the other hand, project scheduling research effort has mostly focused on developing approximate solution techniques. However,

recent project scheduling research papers show a renewed interest for mathematical programming-based solution strategies. Moreover, the best lower bounds ever found on broadly-studied RCPSP test instances, were obtained by a hybrid method involving constraint propagation and a MILP formulation. Additionally, mathematical programming solvers are often the only software available to industrial practitioners. Therefore, the study of exact methods, and especially mathematical programming techniques for solving the RCPSP is of particular theoretical and practical interest. The main objective of this work is to develop new optimal project scheduling techniques inspired by the process scheduling literature.

This thesis consists of a literature review and state-of-the-art, three chapters with novel mathematical programming solution methods for the RCPSP and its variants under the objective of minimising the makespan and finally some concluding remarks. The first part presents new mixed-integer linear programming models for the deterministic single- and multi-mode RCPSP with renewable and non-renewable resources. The modelling approach relies on the Resource-Task Network (RTN) representation, a network representation technique used in process scheduling problems, based on continuous time models. Next, two new binary integer programming discrete-time models and two novel precedence-based mixed integer continuous-time formulations are developed. These four novel mathematical formulations are compared with four state-of-the-art models from the open literature using a total number of 2760 well-known open-accessed benchmark problem instances. The computational comparison demonstrates that the proposed mathematical formulations feature the best overall performance. Finally, a new precedence-based continuous-time formulation is proposed for a challenging extension of the standard single-mode resource-constrained project scheduling problem that also considers minimum and maximum time lags (RCPSP/max). The new formulation is then used to conduct an extensive computational study on a total of 2,250 benchmark problems, which illustrates its efficient performance.

ΠΕΡΙΛΗΨΗ

Ο Χρονοπρογρογραμματισμός Έργων (ΧΕ) παίζει ζωτικό ρόλο στη Διαχείριση Έργων (Project Management), και αποτελεί μία από τις πιο σημαντικές κατευθύνσεις τόσο στην έρευνα όσο και την πρακτική στο πεδίο της Επιχειρησιακής Έρευνας (ΕΕ). Τις τελευταίες δεκαετίες, το Πρόβλημα Χρονοπρογραμματισμού Έργων με Περιορισμένους Πόρους (Resource-Constrained Project Scheduling Problem - RCPSP) έχει τυποποιηθεί και αποτελεί μία από τις μεγαλύτερες προκλήσεις στην βιβλιογραφία της ΕΕ. Το RCPSP περιλαμβάνει τη δημιουργία ενός χρονοπρογράμματος που ικανοποιεί τις συνθήκες *προτεραιότητας* και τους *περιορισμούς πόρων* και υπολογίζει τους χρόνους έναρξης και ολοκλήρωσης των *εργασιών*, έχοντας θέσει κάποιο συγκεκριμένο *στόχο*. Υπάρχουν αρκετές παραλλαγές του RCPSP οι οποίες αναπαριστούν διάφορα πρακτικά προβλήματα με διαφορετικούς στόχους, τύπους πόρων, περισσότερους από έναν τρόπο εκτέλεσης μίας εργασίας (mode), γενικευμένες σχέσεις προτεραιοτήτων μεταξύ των εργασιών, κ.α. Το RCPSP και οι παραλλαγές του ανήκουν στην κατηγορία των ισχυρά NP-hard προβλημάτων και έχουν αναπτυχθεί διάφορες ακριβείς και προσεγγιστικές μεθοδολογίες επίλυσής τους στη βιβλιογραφία.

Ο χρονοπρογραμματισμός αποτελεί σημαντικό πεδίο έρευνας και στον τομέα Λειτουργίας Διεργασιών (Process Operations). Το πρόβλημα χρονοπρογραμματισμού διεργασιών περιλαμβάνει τον υπολογισμό του πιο αποδοτικού τρόπου παραγωγής ενός συνόλου προϊόντων σε συγκριμένο χρονικό ορίζοντα, δεδομένου ενός συνόλου συνταγών επεξεργασίας και περιορισμένους πόρους. Οι εργασίες πρέπει να χρονοπρογραμματιστούν σε ένα βιομηχανικό περιβάλλον, όπου μπορεί να παραχθεί μία πληθώρα διαφορετικών προϊόντων με την ίδια ή διαφορετικές συνταγές, χρησιμοποιώντας κοινόχρηστους περιορισμένους πόρους, όπως εξοπλισμό, υλικά, χρόνο και αναλώσιμα. Τα κοινά χαρακτηριστικά των δύο προβλημάτων, όπως οι απαιτούμενοι τύποι πόρων, οι σχέσεις προτεραιοτήτων μεταξύ

εργασιών, και τα αρχικά/τελικά αποθέματα, υποδηλώνουν ότι η ανταλλαγή τεχνικών επίλυσης μεταξύ των δύο ερευνητικών πεδίων είναι δυνατή και χρήσιμη.

Η βιομηχανία χρονοπρογραμματισμού διεργασιών επωφελείται από τη σημαντική πρόοδο τεχνικών μοντελοποίησης και επίλυσης, καθώς και την ταχύτατα αυξανόμενη υπολογιστική ισχύ. Από την άλλη, η έρευνα στο ΧΕ έχει εστιάσει κυρίως στην ανάπτυξη προσεγγιστικών τεχνικών επίλυσης. Ωστόσο, πρόσφατες ερευνητικές εργασίες στο ΧΕ δείχνουν ότι παρουσιάζεται ανανεωμένο ενδιαφέρον για στρατηγικές επίλυσης που βασίζονται στο μαθηματικό προγραμματισμό. Επιπλέον, τα καλύτερα κατώτατα όρια που έχουν υπολογιστεί σε ευρέως μελετημένα στιγμιότυπα RCPSP, βρέθηκαν με μία υβριδική μέθοδο που χρησιμοποιεί διάδοση περιορισμών (constraint propagation) και ένα μαθηματικό μοντέλο μικτού-ακέραιου γραμμικού προγραμματισμού. Επιπρόσθετα, το μόνο λογισμικό που είναι συνήθως διαθέσιμο σε βιομηχανικό περιβάλλον είναι λογισμικό επίλυσης μαθηματικών προγραμμάτων. Συνεπώς, η μελέτη ακριβών μεθόδων και ειδικά τεχνικών μαθηματικού προγραμματισμού, για την επίλυση RCPSP έχει ιδιαίτερο θεωρητικό και πρακτικό ενδιαφέρον. Ο κύριος στόχος αυτής της εργασίας είναι η ανάπτυξη νέων βέλτιστων μεθόδων ΧΕ, εμπνευσμένες από την βιβλιογραφία του χρονοπρογραμματισμού διεργασιών.

Ο κορμός αυτής της διατριβής αποτελείται από την ανασκόπηση της βιβλιογραφίας και των έως σήμερα εξελίξεων, τρία κεφάλια με καινοτόμες μεθόδους επίλυσης μαθηματικού προγραμματισμού για το RCPSP και παραλλαγές του, θέτοντας ως στόχο την ελαχιστοποίηση του χρόνου ολοκλήρωσης του έργου και τέλος, κάποια συμπεράσματα. Στο πρώτο μέρος, παρουσιάζονται νέα μαθηματικά μοντέλα μικτού-ακέραιου γραμμικού προγραμματισμού για το ντετερμινιστικό RCPSP με έναν (single-mode) και πολλαπλούς (multi-mode) τρόπους εκτέλεσης των εργασιών, που χρησιμοποιεί ανανεώσιμους και αναλώσιμους πόρους. Η νέα προσέγγιση στηρίζεται στην αναπαράσταση Resource-Task Network (RTN), μία τεχνική μοντελοποίησης

που χρησιμοποιείται σε προβλήματα χρονοπρογραμματισμού διεργασιών και βασίζεται σε μοντέλα συνεχούς-χρόνου. Στο επόμενο κεφάλαιο παρουσιάζονται 2 νέα μοντέλα δυαδικού-ακέραιου προγραμματισμού διακριτού-χρόνου και 2 νέα μικτού-ακέραιου προγραμματισμού συνεχούς-χρόνου, που βασίζονται στη διαδοχή εργασιών. Αυτά τα τέσσερα μοντέλα συγκρίνονται με 4 από τα κορυφαία μοντέλα που παρουσιάζονται στη βιβλιογραφία, σε ένα σύνολο 2760 ευρέως χρησιμοποιημένων προβλημάτων, που είναι διαθέσιμα στο διαδίκτυο. Από την υπολογιστική μελέτη αποδεικνύεται ότι τα προτεινόμενα μοντέλα έχουν συνολικά την καλύτερη απόδοση. Τέλος, αναπτύσσεται ένα νέο μαθηματικό μοντέλο συνεχούς-χρόνου βασισμένο στη διαδοχή εργασιών, για μία δύσκολη επέκταση του κλασικού RCPSP που συμπεριλαμβάνει ελάχιστες και μέγιστες χρονικές υστερήσεις μεταξύ των εργασιών (RCPSP/max). Η εκτενής υπολογιστική μελέτη σε 2250 προβλήματα αποδεικνύει την αποτελεσματικότητα του νέου μοντέλου.

ACKNOWLEDGEMENTS

Foremost, I would like to express my gratitude to my advisor, Prof. Michael C. Georgiadis for his continuous support and mentorship, from first contact and initial advice in the early stages, through ongoing guidance and encouragement, and up to this day. I greatly appreciate the opportunity I had to work on my PhD under his supervision.

This dissertation would have not been possible without the numerous brainstorming sessions with Dr. Georgios Kopanos. His experience, assistance and countless practical advices on many aspects of this thesis were invaluable. Our fruitful collaboration has led to a series of novel publications.

I would also like to extend my thanks to my co-advisors Professors Andreas Georgiou and Konstantinos Stergiou for sharing their expertise with sound advice and helpful comments.

Last but not least, I would like to thank my wife Eleni for her love, encouragement and great patience at all times. My parents, brothers and friends have given me their indisputable support through all these years and for which my mere expression of gratitude does not suffice.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Project Scheduling and Management

Project scheduling plays a vital role in project management, and constitutes one of the most important directions in both research and practice in the Operational Research (OR) field. The term *project* means different things to different people and according to ISO 10006 (2003) *Guideline for Quality in Project Management* (Section 3.5), it is used to describe a:

Unique process, consisting of a set of co-ordinated and controlled activities with start and finish dates, undertaken to achieve an objective conforming to specific requirements including constraints of time, cost and resources.

The same ISO, states some of the characteristics a project must have:

1. Unique, non-repetitive phases consisting of processes and activities.
2. Expected to deliver specified (minimum) quality results within pre-determined parameters.
3. Have planned start and finish dates, within clearly specified cost and resource constraints.

A project is a one-time endeavour with a specific objective that must be achieved, under cost, resource and time constraints. The relationships between the various tasks that have to be performed to achieve the project's objectives can be very complex.

The process of project management involves three phases, *planning*, *scheduling* and *controlling*. In the *planning phase* we define the activities that

must be carried out to achieve the project objective and their characteristics (i.e., duration, resource requirements, relationships, constraints, etc). During the *scheduling phase*, the actual project schedule is produced, containing activity starting and/or finishing times. Finally, the *control phase* focuses on examining and determining solutions when variations from the original schedule occur.

1.2 The Resource-Constrained Project Scheduling Problem

Quantitative approaches to project management date back to the 1950s. Early solution procedures like the Critical Path Method (CPM) by Kelley and Walker (1959) and Project Evaluation and Review Technique (PERT) by Malcolm et al. (1959), only took into account activity durations (deterministic or probabilistic) and assumed resources to be available in unlimited quantities. However, in most practical situations this assumption is not realistic, since the required resources are limited and to produce a functional schedule the solution method should take this into account. The additional constraints imposed by the limited resources significantly increase the problem hardness. According to Blazewicz et al. (1983) the Resource-Constrained Project Scheduling Problem (RCPSP) belongs to the class of strongly NP-hard problems.

During the last decades, the RCPSP has become a standard problem for project scheduling in the OR literature. The RCPSP involves the construction of a *precedence* and *resource* feasible time schedule which identifies the starting and finishing times of *activities*, under a specific *objective*. A project consists of a set of interconnected activities and resources, logically linked. These activities usually have to be performed for a successful project completion. Several variations of the RCPSP exist that represent different practical problems with different objectives, resource types, more than one way (mode) to execute an activity, generalised precedence relations for activities, e.t.c.

1.3 Challenges and Motivation

OR uses scientific techniques and tools from various disciplines such as informatics, mathematics, economics, chemistry, even biology to assist decision making or provide a solution to a given problem (preferably optimal). Over the years, the methodology of project scheduling has been developing constantly, trying, from one side to model adequately new practical problems, and, from the other side, to efficiently solve the resulting optimisation problems. The methodology benefited from the development of both: optimisation (especially combinatorial) and computational possibilities.

A number of solution methods for the RCPSP, both *exact* and *approximate* have been proposed in the OR literature. *Exact* techniques usually include mathematical programming formulations and specialised branch-and-bound algorithms. Due to the high degree of complexity of RCPSPs, an even larger number of *approximate* methods such as heuristics and metaheuristics have also been proposed. Roughly speaking, a *heuristic* is a technique designed to solve a problem, or find an approximate solution with low computational requirements, when classic methods fail to find the exact solution. By trading optimality, completeness, accuracy, and/or precision for speed, a heuristic can quickly produce a solution that is good enough for solving the problem at hand.

Scheduling is a critical issue both in project management and process operations. Process and project scheduling problems share common features such as required resource types, precedence relations and initial/target inventories. The process scheduling problem consists of determining the most efficient way to produce a set of products in a time horizon given a set of processing recipes and limited resources. The activities to be scheduled usually take place in multiproduct and multipurpose plants, in which a wide variety of different products can be manufactured via the same recipe or different recipes by sharing limited resources, such as equipment, material,

time, and utilities. The common problem features, such as required resource types, precedence relations and initial/target inventories, suggest that exchanging solution techniques between the two research fields is both possible and useful.

The process scheduling industry is driven by the substantial advances of related modelling and solution techniques, as well as the rapidly growing computational power. Mathematical programming, especially Mixed Integer Linear Programming (MILP), because of its rigorousness, flexibility and extensive modelling capability, has become one of the most widely explored methods for process scheduling problems.

On the other hand, project scheduling research effort has mostly focused on developing approximate solution techniques. However, recent project scheduling research papers (Koné et al. 2011, Bianco and Caramia 2012a, 2012b and Rieck et al. 2012) show a renewed interest for mathematical programming-based solution strategies. The study of exact methods, and especially mathematical programming techniques, for solving the RCPSP is of particular theoretical and practical interest. Indeed, mathematical programming solvers are often the only software available to industrial practitioners. Moreover, the best lower bounds ever found on broadly-studied RCPSP test instances, were obtained by a hybrid method (Demassey et al., 2005) involving constraint propagation and the MILP formulation of Christofides et al. (1987). Also, a branch-and-cut method based on the latter formulation was developed by Zhu et al. (2006), to solve the multimode RCPSP and yielded very competitive results on benchmark problems.

Taking advantage of the continuous commercial software and hardware advances, the size and difficulty of the combinatorial problems that can be solved are constantly growing. The main objective of this thesis is to develop new project scheduling techniques inspired by the process scheduling literature, similar to the paper of Koné et al (2011), which is based on the

work of Pinto and Grossmann on batch process problems (1995).

1.4 Thesis Structure

The rest of this thesis consists of a literature review and state-of-the-art, three novel research chapters and finally some concluding remarks.

Chapter 2 is an introduction to the state-of-the-art in RCPSP. We discuss the resource constrained project scheduling problem, its components, variants and network representation techniques. Afterwards, commonly used objective functions and a classification scheme are described. We then present the test instance sets available for benchmarking new solution techniques, followed by a discussion of mathematical programming concepts, which is the main optimisation method used in this thesis. Next a thorough literature review of both exact and approximate solution techniques is presented. Finally, we describe the commercial software used to solve RCPSP test problem instances and measure the performance of the proposed solution procedures.

Chapter 3 presents new mixed-integer linear programming models for the deterministic single- and multi-mode resource constrained project scheduling problem with renewable and non-renewable resources. The modelling approach relies on the Resource-Task Network (RTN) representation, a network representation technique used in process scheduling problems, based on continuous-time models. First, we propose new RTN-based network representation methods, and then we efficiently transform them into mathematical formulations including a set of constraints describing precedence relations, different types of resources and multiple objectives. Finally, the applicability of the proposed formulations is illustrated using several example problems under the most commonly addressed objective, the makespan minimization.

Chapter 4 introduces two new binary integer programming discrete-time models and two novel precedence-based mixed integer continuous-time formulations for the solution of standard resource-constrained project scheduling problems. The proposed discrete-time models are based on the definition of binary variables that describe the processing state of every activity between two consecutive time points, while the proposed continuous-time models are based on the concept of overlapping of activities, and the definition of a number of newly introduced sets. These four novel mathematical formulations are compared with four representative literature models using a total number of 2760 well-known open-accessed benchmark problem instances involving 30 and 60 activities. A detailed computational comparison study demonstrates the salient performance of the proposed mathematical formulations that feature the best overall performance.

Chapter 5 presents a new precedence-based continuous-time formulation for a challenging extension of the standard single-mode resource-constrained project scheduling problem that also considers minimum and maximum time lags (RCPSP/max), under the objective of minimizing the project makespan. The proposed linear mixed integer programming model is an extension of the continuous-time formulations proposed in Chapter 4 and is used to conduct an extensive computational study on a total of 2,250 well-known and open-accessed benchmark problem instances from the literature. Various problem sizes are considered in the test sets involving 10, 20, 30, 50 and 100 activities. Computational results illustrate the efficient performance of the proposed mathematical formulation.

Finally, concluding remarks and future research directions are drawn in *Chapter 6*.

Chapter 2

Review of the State-of-the-Art

In this chapter we first discuss the resource constrained project scheduling problem, its components, variants and network representation techniques. Afterwards, commonly used objective functions and a classification scheme are described. We then present the test instance sets available for benchmarking new solution techniques, followed by a discussion of mathematical programming concepts, which is the main optimisation method used in this thesis. Next a thorough literature review of modelling and solution techniques is presented. Finally, we describe the commercial software used to solve RCPSP test problem instances and measure the performance of the proposed solution procedures.

2.1 Resource-constrained project scheduling problem (RCPSP)

A project has a finite number of *activities* with specific *durations*. *Precedence relations* between some activities are present and each activity requires certain amounts of *resources* with limited availability, to be processed. For modelling purposes, two dummy activities are added: (i) a start dummy activity to represent the beginning of the project, and (ii) an end dummy activity corresponding to the completion of the project. Dummy activities have zero duration and zero resource requirements. The typical objective of the RCPSP is to find an optimal (or at least feasible) schedule, while satisfying time, precedence and resource constraints, such that a specific objective is optimised (i.e., minimisation of the project makespan). An illustrative example of a simple RCPSP and a feasible solution are displayed in Fig. 2.1. In the standard RCPSP, all information data are deterministic. The resource

type is renewable (i.e., they are not consumed, instead after the completion of an activity, the bound quantities are released and become available again).

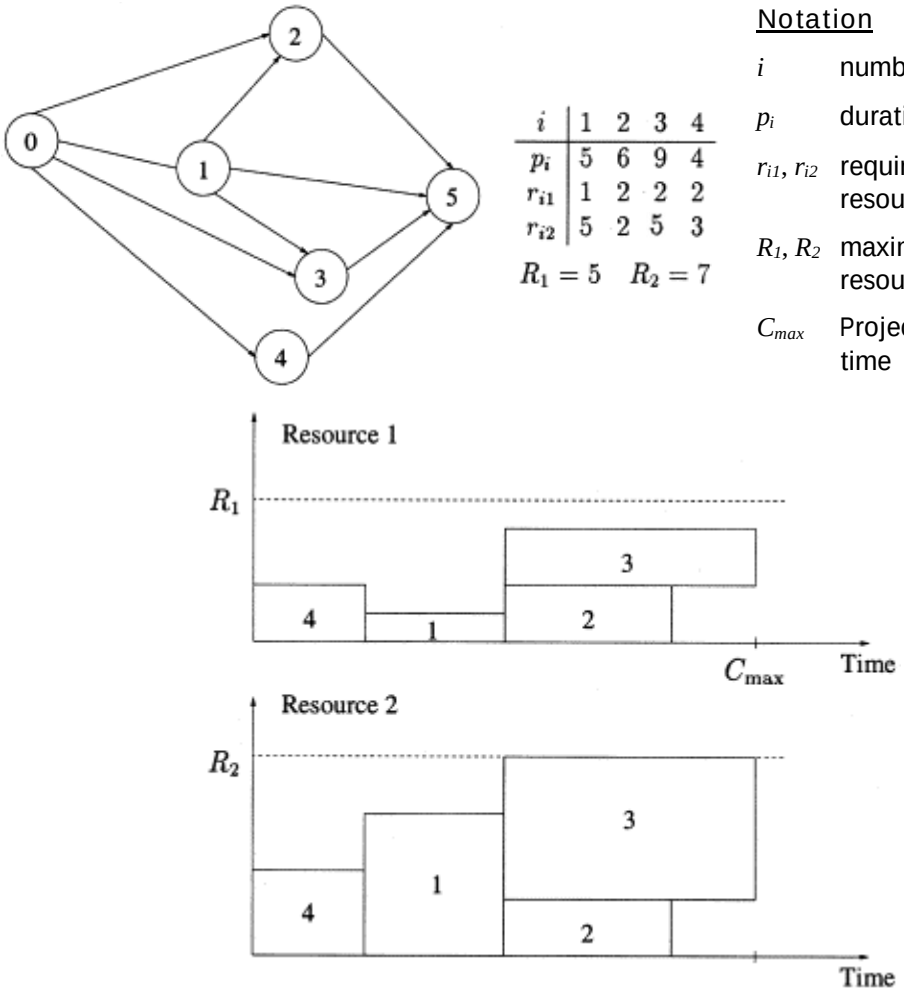


Figure 2.1. Illustrative example of a simple RCPSP and a feasible solution

The standard RCPSP is denoted by $PS|prec|C_{max}$ in accordance with the notation proposed by Brucker et al. (1999), which follows the well-known three-field notation for machine scheduling problems introduced by Graham et al. (1979). More specifically, $PS|prec|C_{max}$ notation specifies the single-mode project scheduling (PS) problem under precedence constraints between activities ($prec$) while minimizing the makespan of the project (C_{max}).

2.2 Project Activities

A project consists of activities, also known as jobs, operations, or tasks. In order to complete the project successfully, all or some of the activities have to be performed. Project activities have various characteristics, depending on the tasks involved.

In some projects the processing of activities may be preempted (interrupted) and recommenced at a later time (*preempt-resume*). In other cases, stopping an activity is allowed, but resuming is not and it has to be restarted (*preempt-repeat*). Finally, for certain activities preemption is not allowed at all and once execution has started, it must be carried out to completion.

Another characteristic regards the order and timing in which activities are executed. These *precedence relations* encountered in project scheduling problems are presented in detail in the following section.

Activity ready times may need to be taken into account, *durations* can be integer or continuous and *deadlines* may be imposed on each activity or on the maximal project duration. The *resource consumption* can occur in constant or variable amounts over their periods of execution.

Most problems assume a single execution mode per activity, while others assume time/cost, time/resource and/or resource/resource trade-offs and give rise to various possible execution modes. While the classical RCPSP is a popular 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. Such an example is the *Multi-mode Resource-Constrained Project Scheduling Problem* (MRCPSP) or $MPS|prec|C_{max}$ according to the notation of Brucker et al. (1999). In this problem, the mode determines the duration of the activity and the requirements for resources of various categories. Another extension studied

by Salewski et al. (1997) are project scheduling problems which generalize multiple activity modes to so-called mode identity constraints in which the set of activities is partitioned into disjoint subsets. All activities in a subset must then be executed in the same mode. Both the time and cost incurred by processing a subset of activities depend on the resources assigned to it.

Finally, activities may require *changeover times*. When these times are sequence-independent, we can include them in the activity durations. However, sometimes changeovers are sequence-dependent (e.g. equipping an excavator with different scoops and workers travelling between sites) and must be taken into account separately in project settings.

2.3 Precedence Relations

Project activities are usually subject to precedence relations. In the rare case where project activities can be performed in any order, no sophisticated project scheduling solution procedures are required.

The traditional *PERT/CPM* methodology uses finish-start (FS) precedence relations with zero time-lag, that is, an activity can only start as soon as all its predecessor activities have finished.

Precedence relations with zero time-lag between two activities are not always sufficient. Elmaghraby and Kamburowski (1992) defined four types of *Generalized Precedence Relations* (GPR): start-start (SS), start-finish (SF), finish-start (FS) and finish-finish (FF) to model minimum and maximum time-lags.

The minimal time-lag ($SS_{ij}^{\min}(x)$, $SF_{ij}^{\min}(x)$, $FF_{ij}^{\min}(x)$, $FS_{ij}^{\min}(x)$) specifies that activity j can only start/finish when its predecessor i has already started/finished for a certain time period (x time units). A maximal time-lag

($SS_{ij}^{\max}(x)$, $SF_{ij}^{\max}(x)$, $FF_{ij}^{\max}(x)$, $FS_{ij}^{\max}(x)$) specifies that activity j should be started/finished at the latest x time periods after the start/finish of activity i.

In the *minimal Finish-Start relation* $FS_{ij}^{\min}(0)$, an activity j (for example the installation of a crane on a site) can start immediately after activity i (for example the preparation of the site) has finished. This strict finish-start relation is the traditional PERT/CPM precedence relation mentioned above. If a certain number of time units must elapse between the end of activity i and the start of activity j (to allow for a lead time for example), the finish-start relation receives a positive lead-lag factor. As such, $FS_{ij}^{\min}(6)$ means that the start of activity j cannot be sooner than 6 time units after activity i finishes.

Minimal Start-Start relations denote that a certain time-lag must occur between the start of two activities. The relationship $SS_{ij}^{\min}(2)$ for example, denotes that the start of activity j (for example place pipe) must lag 2 time units behind the start of activity i (level ground). $SS_{ij}^{\min}(0)$ denotes that activity j (levelling concrete) can start as soon as activity i (pouring concrete) has started. *Ready time* for activity i can be modelled by imposing a minimal start-start relation between the start node in the project network and activity i.

Minimal Finish-Finish relations are used quite often. $FF_{ij}^{\min}(x)$ represents the requirement that the finish of an activity j (for example finish walls) must lag the finish of activity i (for example install electricity) by x time units.

Start-finish relations occur very rarely in practice.

Combinations of the various types of generalized precedence relations can be used. Consider the example of activity i (erect wall frames) and activity j (install electricity). Both activities have a $SS_{ij}^{\min}(x)$ relationship (the

electricians can only start installing electricity when sufficient wall frame surface is in place), but since the electricians need some time to cope with the output of the carpenters who are responsible for erecting the walls, both activities also have a $FF_{ij}^{\min}(x)$ relation.

Maximal time-lags respectively impose a maximum number of time units between the start/finish times of activities. An interesting usage of such time-lags is a $SF_{ij}^{\max}(x)$ between the first and last activity in the project, which in effect sets an upper bound to the project completion time.

The various types of GPRs can be represented in a standardized form by reducing them to minimum SS precedence relationships, through the transformations proposed by Bartusch et al. (1988). This extension of the RCPSP is denoted as *RCPSP/max* or $PS \mid temp \mid C_{max}$, using the notation of Brucker et al. (1999). More specifically, $PS \mid temp \mid C_{max}$ notation specifies the single-mode project scheduling problem (PS) under general temporal constraints given by minimum and maximum start-start time lags between activities ($temp$) while minimizing the makespan of the project (C_{max}).

2.4 Resource Types

Each project activity (besides dummy ones) requires some resources for its processing, which are available in limited amounts. Examples of resources are raw materials, intermediate products, tools, machinery, manpower, financial, energy, etc. Węglarz (1979) and Blazewicz et al. (1983) categorize resources used by project activities as *renewable*, *non-renewable* and *doubly constrained*.

Renewable resources are periodically renewed, but their quantity is limited over each time period and may differ from one period to the next. Some examples are manpower, machines, equipment, power and fuel flow.

For *non-renewable resources*, constraints on availability only concern total consumption over the whole period of project duration and not each time period. Raw materials are a typical example of non-renewable resources, since they are available at a specific quantity for a project.

Doubly constrained resource quantities are both per period and per project constrained. Money is an example of such resource, since there is usually a specific total budget for the entire project, as well as a limited cash flow per period, according to progress. As formally shown by Talbot (1982), each doubly constrained resource can be represented by one renewable and one non-renewable resource, respectively.

Partially (non)renewable resources, introduced by Böttcher et al. (1996), Schirmer and Drexl (1996) and Drexl (1997) limit utilization of resources within a subset of the planning horizon. Essentially, partially (non)renewable resources can be viewed as a generic resource concept in project scheduling, as they include both renewable and non-renewable (and, hence doubly constrained) resources. An example is that of a planning horizon of a month with workers whose weekly working time, not the daily time, is limited by their working contract.

It has been shown by Böttcher et al. (1999) that both renewable and non-renewable resource categories can be depicted by partially renewable resources. A partially renewable resource, with a specified availability for a time interval equal to a unit duration period, is essentially a renewable resource. A partially renewable resource, with a specified availability for a time interval equal to the project horizon, is essentially a non-renewable resource. Partially renewable resources with a specified availability on both a unit duration period and a total project horizon basis can be interpreted as doubly constrained resources.

2.5 Project Network Representations

Two representations have been commonly used to capture project networks, the *Activity-on-Arc* (AoA) which is event-based and *Activity-on-Node* (AoN) which is activity based. The latter represents activity interdependencies in a more natural way, without the need for dummy activities. Understanding an AoN representation is easier, even for inexperienced users. Finally, reviewing an AoN network is easier when a change occurs in the network.

2.5.1 Activity-on-Arc (AoA)

An *Activity-on-Arc* (AoA) diagram is based on the idea that each activity is a transition between two events, its start and its end. Each activity is represented as an arc, which starts and finishes at a *node* (drawn as a circle). Each node represents an *event*, a point of zero time duration, which signifies the completion of all the activities leading into it and the start of all activities pointing out.

An AoA network can contain no cycles, because if it did, the transitivity property of precedence would lead to the conclusion that an activity would have to precede itself, which is impossible.

In AoA networks we use two *dummy nodes* to represent the start and completion of the project. The *initial event* is the starting node of all activities and has no predecessor(s), while the *terminal event* is the ending node of all activities and has no successor(s).

Any two nodes may be connected by only one activity. So, for several activities to be executed simultaneously, we need to introduce *dummy activities*. Dummy activities are drawn as dotted arcs, consume no resources and have zero time duration. An example of an AoA diagram is shown in Fig. 2.2 below.

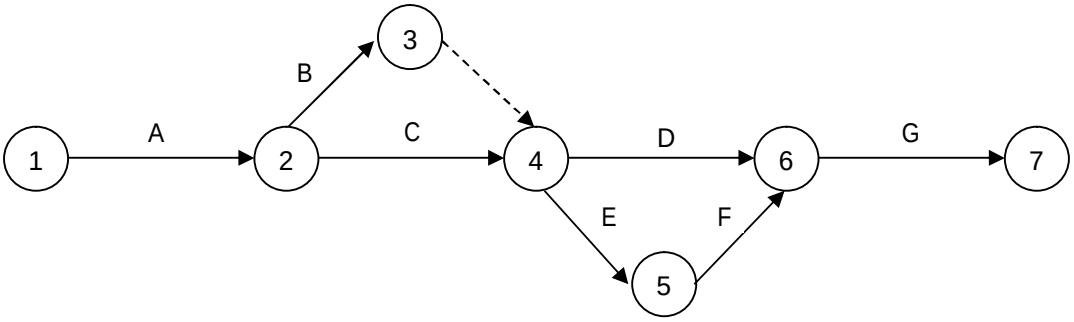


Figure 2.2. Example of an Activity on Arc network

2.5.2 Activity-on-Node (AoN)

Activity-on-Node (AoN) is a network representation for activity sequencing, also known as Precedence Diagramming Method (PDM). Activity sequence diagrams use boxes or rectangles to represent the activities which are called nodes. The nodes are connected with other nodes by arrows, which show the dependencies between the connected activities. To construct an AoN network, we must draw one node for each activity and an arrow from all nodes i to nodes j, if activity i precedes activity j. The respective AoN network for the AoA represented in Fig. 2.2 is displayed in Fig. 2.3 below.

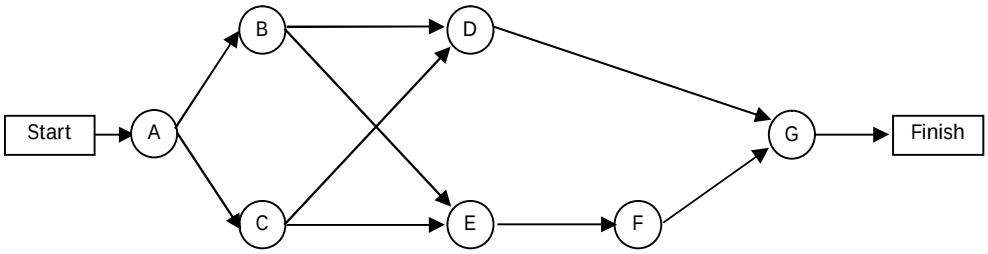


Figure 2.3. Example of Activity on Node network

Dummy activities (nodes) are only needed to satisfy the requirement that the network possesses only one initial and one terminal node.

The AoN has certain advantages over the AoA, since it represents activity interdependencies in a more natural way, it is easier to understand, even for inexperienced users, and easier to review when a change occurs in the network. A more thorough comparison of the two methods can be found in Kolisch and Padman (2001).

2.6 Objectives of Project Scheduling

Project scheduling objective functions can be *regular* or *non-regular*. A *regular performance measure* is a non-decreasing function of the activity completion times (in the case of a minimization problem), otherwise it is *non-regular*. Regular objective functions have received much more attention in the literature than non-regular ones, especially the makespan or project length.

Each of the objectives for deterministic project scheduling presented in the following sections can and has been examined for problems with a diversity of resource and activity characteristics.

2.6.1 Time-based objectives

Minimizing the project makespan is undoubtedly the most popular time-based objective function discussed in the project scheduling literature. Most often it is recognized as the most relevant objective in various review papers, Kolisch (1996b), Herroelen (2005), Hartmann et al. (2010). According to Kolisch (1996b):

1. The majority of income payments of projects (e.g. in the construction industry) occur at the end of a project or at the end of predefined project phases. Finishing the project early reduces the amount of tied-up capital.
2. The quality of forecasts tends to deteriorate with the distance into the future of the period for which they are made. Minimizing the

project duration reduces the planning horizon and, therefore, the uncertainty of data.

3. Finishing products as early as possible lowers the probability of time-overruns of the project.
4. By freeing resource capacity as early as possible the flexibility of the company can increase in order to better cope with changes of the economic environment.
5. Additionally, high resource utilization at the beginning of the planning horizon leads to a larger amount of free resources at the end of the planning horizon and, thus, raises the ability to accept and process new projects.

Other time-based objectives based on project *lateness*, *tardiness* and *earliness* exist. The *lateness* L_i of an activity i is the difference of its completion time and its due date. The lateness can be zero (if the task finishes on time), positive (if the task finishes later) or negative (if the task finishes earlier). The *tardiness* T_i is the same as lateness, but it cannot be negative ($T_i = \max(0, L_i)$). *Earliness* E_i is defined as $E_i = \max(0, -L_i)$.

In the literature, we encounter a number of objectives based on lateness, tardiness and earliness, such as *minimization of the weighted tardiness* (Kolisch, 2000), *minimization of the maximum lateness and of the weighted total tardiness* (Neumann, 2002), etc.

2.6.2 Maximizing the Net Present Value

The value of a certain amount of money is a function of the time of receipt or disbursement of the cash. Money received today is more valuable than money to be received in some future time period, because it can be invested to start earning interest immediately. The nature and timing of the cash flows in projects depend on the contract. The contractor would like to receive as

much as possible, as early as possible to initiate activities, while the client would like to delay payments for completion of parts of the project as long as possible, since progress payments represent expenses.

To cope with such problems we need to set financial objectives related to incoming and outgoing cash flows, including discount rates. Such objectives are referred to in the literature as *Maximizing the Net Present Value* (NPV) of the project and were introduced by Russell (1970).

In the *unconstrained* case, both the amount and timing of cash flows are known and we attempt to maximize their NPV. Cash flows can be associated with the completion of set of activities, occur at regular intervals (e.g. weekly) or be compounded to a single cash flow at the beginning/end of an activity.

When both the amount and timing of the cash flows must be determined, we have the *payment scheduling problem*. Finally, the *resource-constrained* case is more complex, since we need to additionally deal with limited resources.

2.6.3 Other objectives

A number of other objectives has been examined in the literature, such as *minimizing resource availability costs* (Demeulemeester 1995, Franck and Schwindt 1995, Kimms 1998, Möhring 1984, Zimmermann 1997), the *discrete time/resource trade-off problem* (Elmaghraby, 1977), *minimizing the sum of costs* (Möhring et al. 2003, Achuthan and Hardjawidjaja, 2001), etc.

Finally, certain problems deal with multi-objective scheduling that requires the optimisation of more than one objective (Nabrzyski and Węglarz 1994, Al-Fawzana and Haouari 2005, Bomsdorf and Derigs 2008).

2.7 A Classification Scheme

The increasing research interest in the area of Project Scheduling from both science and practice has led to an ever growing number of problem types. Various acronyms, such as RCPSP, MRCPSP, RCPSP-GPR have been extensively used to describe the problem class. However, these abbreviations offer an inadequate description of the problem characteristics and may often lead to misconceptions.

Herroelen et al. (1998) proposed a classification system compatible with what is generally accepted in machine scheduling (Graham et al., 1979) and resource-constrained machine scheduling (Blazewicz et al., 1983), because machine scheduling models are special cases of project scheduling models. The proposed scheme resembles machine scheduling problems in that it is also composed of three fields α / β / γ, but the composition of the fields and the precise meaning of the various parameters are mostly new and specific to the area of project scheduling. The meaning of each field is described below:

1. Field **α** – Represents resource characteristics and contains up to three elements,
2. Field **β** – Represents activity characteristics and contains up to nine fields and
3. Field **γ** – Contains one element and represents the performance measure.

Brucker et al. (1999) provided their own classification scheme, but Herroelen et al. (2000) revealed serious shortcomings and turned their own original scheme into a unified classification scheme for resource scheduling (Demeulemeester and Herroelen, 2002).

In the next sections, we cover the characteristic values of the three fields in the original classification scheme (Herroelen, 1998) that apply to deterministic project scheduling problems.

2.7.1 Field α – Resource Characteristics

Field α, is a set describing the resource characteristics and consists of at most three parameters α_1, α_2 and α_3. The symbol ◦ denotes an empty field and will be used when a field is omitted.

Parameter $\alpha_1 \in \{\circ, 1, m\}$ represents the number of resource types used in the problem:

$\alpha_1 = \circ$	no resource types are considered in the scheduling problem
$\alpha_1 = 1$	one resource type is considered
$\alpha_1 = m$	the number of resource types is equal to m

The second parameter, α_2, describes the resource types used. As mentioned in Section 2.4, in the project scheduling literature a common distinction is made between various types of resources:

$\alpha_2 = \circ$	absence of any resource type specification
$\alpha_2 = 1$	renewable resources, the availability of which is specified for the unit duration period
$\alpha_2 = T$	non-renewable resources, the availability of which is specified for the entire project horizon T
$\alpha_2 = 1T$	both renewable and nonrenewable resources (including also doubly constrained resources, the availability of which is specified on both a unit duration period and a total project horizon basis)
$\alpha_2 = v$	partially (non-)renewable resources the availability of which is renewed in specific time periods

Finally, parameter α_3 describes the resource availability characteristics of the problem.

$\alpha_3 = \circ$	(partially) renewable resources are available in constant amounts
$\alpha_3 = v\alpha$	(partially) renewable resources are available in variable amounts

2.7.2 Field β – Activity Characteristics

The second field describes the activity characteristics of the problem, using nine parameters. The first parameter β_1 indicates the possibility of activity preemption:

$\beta_1 = \circ$	no activity preemption is allowed
$\beta_1 = pmtn$	Activity preemption of type preempt-resume is allowed
$\beta_1 = pmtn - rep$	Activity preemption of type preempt-repeat is allowed

Parameter β_2 describes the activity precedence relations:

$\beta_2 = \circ$	no precedence constraints exist (activities can be executed at any order)
$\beta_2 = cpm$	only Finish-to-Start relationships with zero time lag are used, as in the PERT/CPM model
$\beta_2 = \min$	precedence diagramming relations with minimal time lags are used
$\beta_2 = gpr$	the activities are subject to generalized precedence relationships with minimal and maximal time lags

The third parameter β_3, denotes the ready times for activities:

$\beta_3 = \circ$	all ready times are zero
$\beta_3 = \rho_j$	ready times vary per activity

Parameter β_4 describes the duration of project activities:

$\beta_4 = \circ$	arbitrary integer durations
$\beta_4 = cont$	arbitrary continuous durations
$\beta_4 = (d_j = d)$	all activities have a duration equal to *d* units

The fifth parameter β_5 describes the project deadlines:

$\beta_5 = \circ$	no deadlines
$\beta_5 = d_j$	deadlines are imposed on individual project activities
$\beta_5 = \delta_n$	a deadline is imposed on the project

The next parameter β_6 expresses the nature of resource requirements for project activities:

$\beta_6 = \circ$	constant discrete resource requirements (e.g. a number of units for every time period of activity execution)
$\beta_6 = vr$	variable discrete resource requirements (e.g. a number of units which varies over the periods of activity execution)

If the activity durations have to be determined by the solution procedure on the basis of a resource requirement function, then the following settings are used:

$\beta_6 = disc$	The resource requirements are a discrete function of the activity duration
$\beta_6 = cont$	The resource requirements are a continuous function of the activity duration
$\beta_6 = \text{int}$	the activity resource requirements are expressed as an intensity or rate function

If needed, the user can be more specific on the type of resource requirement function (e.g. concave, convex, linear, e.t.c.)

The type and number of possible execution modes of project activities is described by parameter β_7.

$\beta_7 = °$	activities are performed in a single execution mode
$\beta_7 = mu$	activities have multiple preset execution modes
$\beta_7 = id$	Activities are subject to mode identity constraints

The next parameter β_8 is used to address financial issues of the project activities. Most models with cash flows, assume the cash flow amounts to be known. Other models assume that the cash flows are periodic in that they occur at regular time intervals or with a known frequency. Still other models assume that both the amount and the timing of the cash flows have to be determined.

$\beta_8 = °$	no cash flows are specified
$\beta_8 = c_j$	activities have associated cash flows

$\beta_8 = c_j^+$	activities have an associated positive cash flow
$\beta_8 = per$	periodic cash flows are specified
$\beta_8 = sched$	both the amount and timing of the cash flows have to be determined

Finally parameter β_9 denotes the changeover times. Changeover times are usually sequence-independent, so we include them in the activity durations. However, sometimes changeovers are sequence-dependent (e.g. equipping an excavator with different scoops and workers travelling between sites) and must be taken into account in project settings:

$\beta_9 = \circ$	no changeover times
$\beta_9 = s_{jk}$	Sequence-dependent changeover times

2.7.3 Field γ – Performance Measures

The last field is used to define the performance measures:

$\gamma = reg$	the performance measure is any regular measure
$\gamma = nonreg$	the performance measure is any nonregular measure

Obviously, the list of performance measures is practically endless. Some examples of such measures are:

$\gamma = C_{\max}$	minimize the project makespan
$\gamma = npv$	maximize the net present value of the project
$\gamma = \overline{F}$	minimize the average flow time over all subprojects or activities
$\gamma = curve$	determine the time vs cost trade-off curve

$\gamma = rac$	minimize the resource availability costs
$\gamma = L_{\max}$	minimize the project lateness
$\gamma = T_{\max}$	minimize the project tardiness
$\gamma = av$	minimize the resource allocations whilst meeting the project deadlines
$\gamma = multi$	multiple objectives

2.8 Test Instance Sets

When testing exact or heuristic methods for project scheduling, test instances are necessary for evaluation and comparison reasons. Although the first problem test sets were usually a collection of solved instances in the literature, several parameter-driven instance generators have been developed for the RCPSP and its extensions. A summary of online available test sets and their characteristics is provided in Table 2.1.

Table 2.1. Test set characteristics and available instances

Test Set	Problem Types	Number of Instances	Number of Activities	Number of Modes	Number of Resources
Patterson	RCPSP	110	7-50	1	1-3 Renewable
ProGen	RCPSP	2040	30,60,90,120	1	4 Renewable
	MRCPSP	11182	10,12,14,16,18, 20,30	1,2,3,4,5	1-3 Renewable 1-5 Non-Renewable
ProGen/max	RCPSP/max	2520	10,20,30,50,100, 200,500,1000	1	5 Renewable
	RIP/max	810	10,20,30	1	1,3,5 Renewable
	RCPSP/RLP	1890	10,15,20,30,50, 100,200,500,1000	1	1,3,5 Renewable
	MRCPSP/max	1620	10,20,30,50,100	2,3,4,5	3,5 Renewable 2,3 Non-Renewable
RanGen2	RCPSP	1800	30	1	4 Renewable

2.8.1 Patterson test set

Patterson (1984) was the first to assemble a set of 110 test problems (with 7 up to 50 activities and 1 to 3 renewable resource types) which over the years became a standard for validating optimal and suboptimal procedures for the RCPSP. Herroellen et al. (1998) pointed out that test sets should span the full range of complexity, from very easy to very hard problem instances, generated by using a controlled design of specified problem parameters. The generation of easy and hard problem instances, however, appears to be a very difficult task which heavily depends on the possibility to isolate the factors that precisely determine the computing effort required by the solution procedure used to solve a problem, and the calibration of the scale that characterises this effort. Such problem sets were generated by ProGen (Kolisch et al., 1995) and quickly replaced the Patterson test problem set.

2.8.2 ProGen and ProGen/max

ProGen was developed by Kolisch et al. (1995), as a network instance generator for the classical RCPSP as well as the multi-mode extension. A number of instances, systematically generated by ProGen, are available for researchers in PSPLIB (Kolisch and Sprecher, 1996), an online scheduling library (http://129.187.106.231/psplib/). PSPLIB test sets have been used as a benchmark in a large number of studies. According to the review papers by Herroelen (2005), Lancaster and Ozbayrak (2007) and Hartmann and Briskorn (2010), it is the most widely used test bed for the RCPSP and its variations.

The library contains data sets that can be used for the evaluation of solution procedures for single- and multi-mode resource-constrained project scheduling problems, as well as reported optimal and heuristic solutions. Researchers can download the benchmark sets to evaluate their algorithms, and send their results to be added to the library, or generate their own test-

data. The instances available in PSPLIB were generated by varying the *Network Complexity* (NC), *Resource Factor* (RF) and *Resource Strength* (RS) parameter values. NC is the average number of non-redundant arcs per activity (node), including the dummy source and sink activities. RF reflects the average quantity of resources required per activity, and is normalised in the interval [0,1]. A value of 1 suggests that all activities require all resources, while a value of 0, that no resource is required by any activity (which corresponds to the unconstrained case). The RS parameter measures the strength of the resource constraints. It is a scaling parameter expressing resource availability as a convex combination of a minimum and maximum level with values in the interval [0,1].

Schwindt (1996) developed a version called *ProGen/max* in order to include minimal and maximal time lags. This generator can also produce activities with multiple modes, as well as instances for the resource levelling and the resource investment problem.

2.8.3 RanGen and RanGen2

The generators presented in the previous sections do not generate strongly random networks because they do not allow selection from the full space of feasible networks. Hence, Demeulemeester et al. (2003) developed RanGen, which claims to generate strongly random networks that conform to desired values of complexity measures. *RanGen* produces single- and multi-mode project instances based on different control parameters than ProGen. Vanhoucke et al. (2008) enhanced RanGen to *RanGen2*, by incorporating further topological network measures.

RanGen and RanGen2 use different measures than ProGen. These six topological measures are displayed in Table 2.2. The first five indicators (I_1 to I_5) are based on the indicators proposed by Tavares et al. (1999). More

specifically, I_1, I_2 and I_4 are exact copies of the original ones, while I_3 and I_5 are improved versions. The last indicator I_6 is entirely new.

Table 2.2. Network structure topological indicators for RanGen2

Indicator	Description
I_1	Network size expressed as number of activities
I_2	closeness to a serial or parallel graph
I_3	distribution of activities over the progressive levels
I_4	presence of short arcs
I_5	presence of long arcs
I_6	topological float of activities

2.8.4 Other test instance generators

A test set generator for AOA project networks was proposed by Agrawal et al. (1996). In this generator, named *DAGEN*, the user can specify the level of a summary measure of network complexity. Browning and Yassine (2010) presented a generator for problems consisting of multiple projects with controlled resource distributions and amounts of resource contention. They also generated 12,320 test problems for a full-factorial experiment and used analysis of means to conclude that the generator produces “near-strongly random” problems.

2.9 Mathematical Programming

Mathematical Programming is the use of mathematical models, particularly optimisation models, to assist decision making. It is very different than and should not be confused with *computer programming*, even though solving most practical problems requires the use of computer calculation power.

The term programming is actually used in the sense of *planning/scheduling*. The goal of mathematical programming is to *optimise* (minimise or maximise) a quantity. This quantity is known as the *objective function*.

2.9.1 Mathematical Modelling

Many scientific applications utilise *models* as a structure to present features and characteristics of an "entity". Sometimes, models are *physical*, such as a model aircraft used in wind tunnels to test its aerodynamics. However, the models examined in operational research are *abstract*. These abstract models are usually *mathematical* and involve a set of mathematical relationships involving equations, inequalities and logical dependencies to describe real-world relationships and constraints.

Building a mathematical model gives us greater insight and understanding of the "entity" being modelled. A number of not so obvious aspects are revealed and further mathematical analysis can lead to different courses of action. Finally, it is easier to implement various "scenarios", however radical, and harmlessly observe their outcome.

It is important to understand that a model is actually defined by its relationships and constraints. These are, to a large extent, independent of the *data* in the model. This means that a model can be used in a variety of similar situations, which differ in the data involved (i.e., costs, resource availabilities, activity durations, etc).

According to Yang (2008), whatever the real-world problem is, it is usually possible to formulate the optimisation problem in a generic form. All optimization problems with explicit objectives can in general be expressed as nonlinearly constrained optimization problems in the following generic form:

$$\begin{aligned} &\text{minimise } / \underset{x\in\Re^n}{\text{ maximise }} f(x) \\ &\text{subject to} \\ &\quad \phi_m(x) = 0 \quad m = (1,...,M) \\ &\quad \psi_k(x) \le 0 \quad k = (1,...,K) \\ &\text{where } x = (x_1, x_2, ..., x_n)^T \in \Re^n \end{aligned} \qquad (1.1)$$

where $f(x)$, $\phi_m(x)$, and $\psi_k(x)$ are scalar functions of the real column vector x. The function $f(x)$ is called *objective function*, and is a quantitative measure of the performance of the system in question. The components x_i of vector x are called *decision variables*, or simply variables, and can be either continuous, discrete or a mix of these two. The variables are the unknowns, whose values are to be determined so that the objective function is optimised. Additionally, $\phi_m(x)$ are constraints in terms of M equalities, and $\psi_k(x)$ are constraints written as K inequalities. Therefore, there are M+K constraints in total. *Constraints* represent any restrictions that the decision variables must satisfy.

According to Nemhauser and Wolsey (1999), in mathematical programming (especially when integer decisions are involved) formulating a "good" model is of crucial importance to solving the problem. Indeed, the quality of a mathematical formulation strongly depends on the choice of decision variables and the constraints formulated. At this point it should be emphasized that it is instinctive to believe that the computational time increases and computational feasibility decreases as the number of constraints increases, however, trying to find a formulation with a small number of constraints is often not a good strategy (Nemhauser and Wolsey, 1999).

The procedure of identifying the decision variables, constraints and objective function is known as modelling. Depending on the properties of the functions f, ϕ, and ψ and the vector x, the mathematical program (1.1) is called:

- *Linear:* If x is continuous and the functions f, ϕ, and ψ are all linear.
- *Nonlinear:* If x is continuous and at least one of the functions f, ϕ, and ψ is nonlinear.
- *Mixed integer linear:* If x requires at least some of the variables x_i to take integer (or binary) values only; and the functions f, ϕ, and ψ are linear.
- *Mixed integer nonlinear:* If x requires at least some of the variables x_i to take integer (or binary) values only; and at least one of the functions f, ϕ, and ψ is nonlinear.

2.9.2 Types of optimal solutions

The *feasible region* contains the set of feasible solutions to the problem, $F = \left\{x \in \Re^n \mid \phi_m(x) = 0, \psi_k(x) \le 0\right\}$. A feasible solution x^* that optimises the objective function is called *optimal*, $x^* \in F : f(x^*) \le or \ge f(x), \forall x \in F$. The value of the objective function for the optimal solution $f(x^*)$ should be less or equal ($\le$) to every other value, with $x \in F$, when we have a minimisation problem and greater or equal ($\ge$) for a maximisation problem.

If, for a minimisation problem, the property $f(x^*) \le f(x)$ is satisfied for all $x \in F$, then x^* is a *global minimum*. If this condition is satisfied for all x in a neighbourhood of x^*, then it is a *local minimum*, as shown in Fig. 2.4. Finally, if the inequality holds strictly, x^* is called a *strong minimum*, whereas it is called a *weak minimum* otherwise.

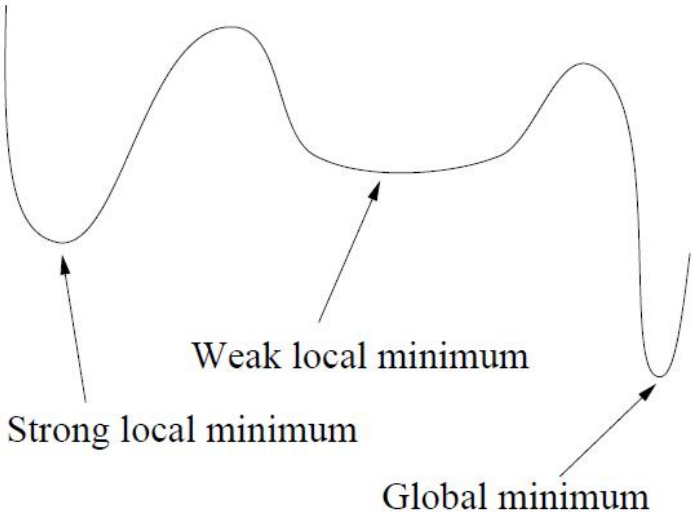


Figure 2.4. Types of minima

2.9.3 Linear Programming (LP)

Linear Programming (LP) is a mathematical method for determining a way to achieve the best outcome (such as maximum profit or lowest cost) in a given mathematical model for some list of requirements represented as linear relationships.

More formally, linear programming is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. Its feasible region is a convex polyhedron, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality. Its objective function is a real-valued affine function defined on this polyhedron. A linear programming algorithm finds a point in the polyhedron where this function has the smallest (or largest) value if such a point exists.

Linear programs are problems that can be expressed in canonical form:

maximise $c^T x$ (1.2)

subject to $Ax \leq b$

and $x \geq 0$

where x represents the vector of variables (to be determined), c and b are vectors of (known) coefficients, A is a (known) matrix of coefficients, and $(\cdot)^T$ is the matrix transpose. The expression to be maximised or minimised is called the objective function ($c^T x$ in this case). The inequalities $Ax \leq b$ are the constraints which specify a convex polyhedron over which the objective function is to be optimised. In this context, two vectors are comparable when they have the same dimensions. If every element of the first is less-than or equal-to the corresponding element of the second, then we say that the first vector is less-than or equal-to the second vector.

Two well-known solution procedures for LP problems are the Simplex algorithm and the interior point method.

Simplex Algorithm

The Simplex Algorithm, developed by George Dantzig in 1947, solves LP problems by starting with an initial *Basic Feasible Solution* - BFS (a basic solution that satisfies all the nonnegativity constraints) and testing its optimality. If the optimality condition is verified, then the algorithm terminates. Otherwise, the algorithm identifies an adjacent BFS, with a better objective value. The optimality of this new solution is tested again, and the entire scheme is repeated, until an optimal BFS is found. Since every time a new BFS is identified the objective value is improved (except from a certain pathological case that we shall see later), and the set of BFS's is finite, it follows that the algorithm will terminate in a finite number of steps (iterations).

It is also interesting to examine the geometrical interpretation of the behaviour of the Simplex algorithm. Given the above description of the algorithm and the correspondence of BFS's to extreme points, it follows that Simplex essentially starts from some initial extreme point, and follows a path along the edges of the feasible region towards an optimal extreme point, such that all the intermediate extreme points visited are improving (more accurately, not worsening) the objective function (see Fig. 2.5).

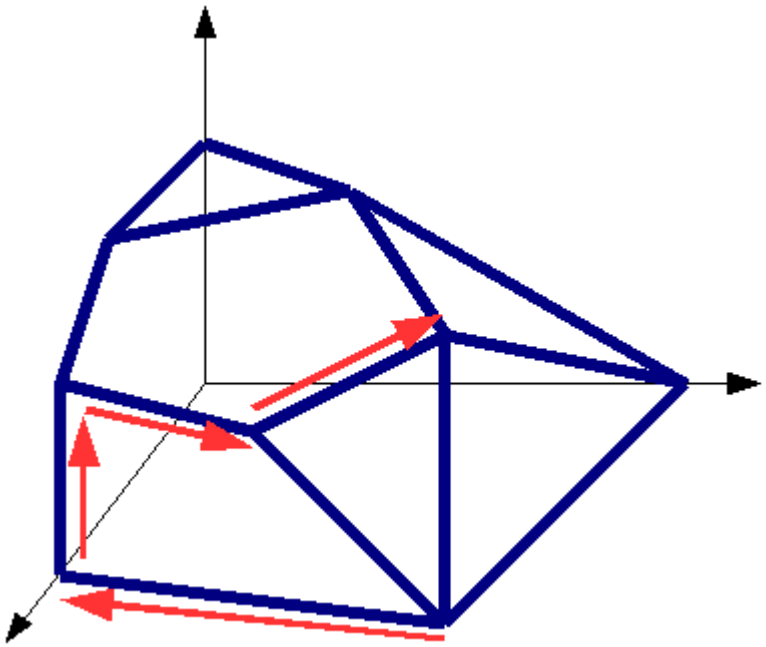


Figure 2.5. Graphical interpretation of the Simplex method

It is worth mentioning that in 1953 Dantzig and Orchard-Hays proposed the Revised Simplex method, which actually is not a different method but is a different (more efficient) way to carry out each computational step of the Simplex method.

Interior Point Method

An interior point method is one that improves a feasible interior point of the linear program by steps through the interior (see Fig. 2.6), rather than

around the boundary of the feasible region, as in the Simplex algorithm.

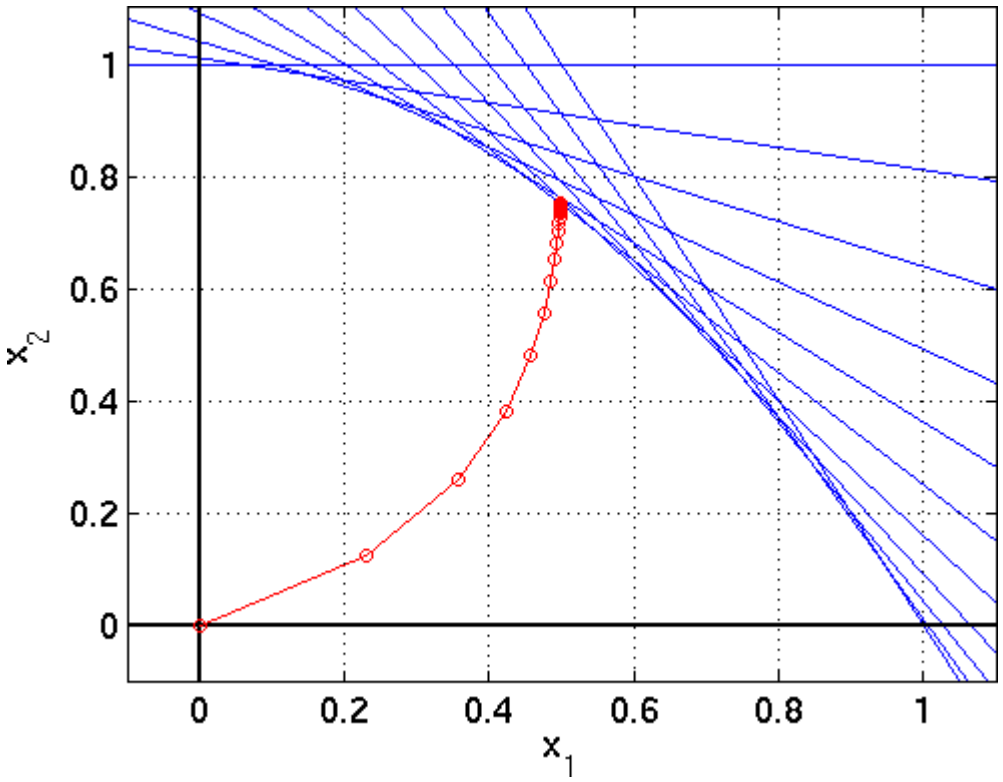


Figure 2.6. Graphical interpretation of the Interior-point method

A polynomial time linear programming algorithm using an interior point method was found by Karmarkar (1984). Arguably, interior point methods were known as early as the 1960s in the form of the barrier function methods, but the media hype accompanying Karmarkar's announcement led to these methods receiving a great deal of attention. Karmarkar's breakthrough revitalized the study of interior point methods and barrier problems, showing that it was possible to create an algorithm for linear programming which was characterised by polynomial complexity and competitive with the simplex method. Already Khachiyan's ellipsoid method (Khachiyan, 1979) was a polynomial time algorithm; however, in practice it was too slow to be of practical interest.

The key idea behind interior-point methods are as follows, assuming an initial feasible interior point is available and that all moves satisfy $Ax = b$:

1. Try to move through the interior in directions that show promise of moving quickly to the optimal solution.
2. Recognize that if we move in a direction that sets the new point too "close" to the boundary, this will be an obstacle that will impede our moving quickly to an optimal solution. One way around this is to transform the feasible region so that the current feasible interior point is at the center of the transformed feasible region. Once a movement has been made, the new interior point is transformed back to the original space, and the whole process is repeated with the new point as the center.
3. The simple stopping rule typically followed is to stop with an approximate optimal solution when the difference between iterations "deemed" sufficiently small in the original space.

2.9.4 Mixed Integer Programming (MIP)

A Mixed-Integer Programming (MIP) problem results when some of the variables in the model are real-valued and some of the variables are integer and/or binary. The model is therefore “mixed”. Integer variables are required when modelling indivisible entities, whereas a very common use of binary (0–1) variables is to represent binary choice. Consider an event that may or may not occur, and suppose that it is part of the problem to decide between these possibilities. In order to model such a choice, a binary variable, which typically equals 1 if the event occurs and zero otherwise, can be used. Depending on the specific problem, the event may represent yes/no decisions, logical conditions, fixed costs or piecewise linear functions.

Mixed Integer Programming (MIP) problems can be expressed in standard form, as follows:

$$\begin{aligned} &\text{maximise } c^T x + hy \\ &\text{subject to } Ax + Gy \le b \\ &x \ge 0 \text{ and} \\ &y \ge 0 \text{ is integer and/or binary} \end{aligned} \qquad (1.3)$$

where x represents the vector of non-negative variables, y represents the vector of integer and/or binary variables, c and b are vectors of coefficients, and A and G are matrices of coefficients. In this case, the objective function is $c^T x + hy$.

For further reading on MIP optimisation, we refer to the excellent books of Floudas (1995) and Nemhauser and Wolsey (1999). MIP problems are usually solved using branch-and-bound, cutting planes or branch-and-cut methods.

Branch-and-Bound Techniques

Branch and Bound (B&B) is by far the most widely used tool for solving large scale NP-hard combinatorial optimization problems (Floudas, 1995). The method was first proposed by Land and Doig (1960) for discrete programming. The Branch and Bound technique consists of a systematic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded *en masse*, by using upper and lower estimated bounds of the quantity being optimized.

The method is based on the observation that the enumeration of integer solutions has a tree structure. For example, consider the complete enumeration of a model having one general integer variable x_1, and two binary variables x_2 and x_3, whose ranges are $1 \le x_1 \le 3$, $0 \le x_2 \le 1$, and $0 \le x_3 \le 1$. Fig. 2.7 shows the complete enumeration of all of the solutions for these variables, even those which might be infeasible due to other constraints

on the model. The structure in Fig. 2.7 looks like a tree lying on its side with the *root* (or root node) on the left, labelled “all solutions”, and the *leaves* (or leaf nodes) on the right. The leaf nodes represent the actual enumerated solutions, so there are 12 of them: (3 possible values of x_1) × (2 possible values of x_2) × (2 possible values of x_3). For example, the node at the upper right represents the solution in which $x_1 = 1$, $x_2 = 0$, and $x_3 = 0$. The other nodes can be thought of as representing sets of possible solutions. For example, the root node represents all solutions that can be generated by growing the tree. Another intermediate node, e.g. the first node directly to the right of the root node, represents another subset of all of the possible solutions, in this case, all of the solutions in which $x_1 = 2$ and the other two variables can take on any of their possible values. For any two directly connected nodes in the tree, the *parent* node is the one closer to the root, and the *child* node is the one closer to the leaves.

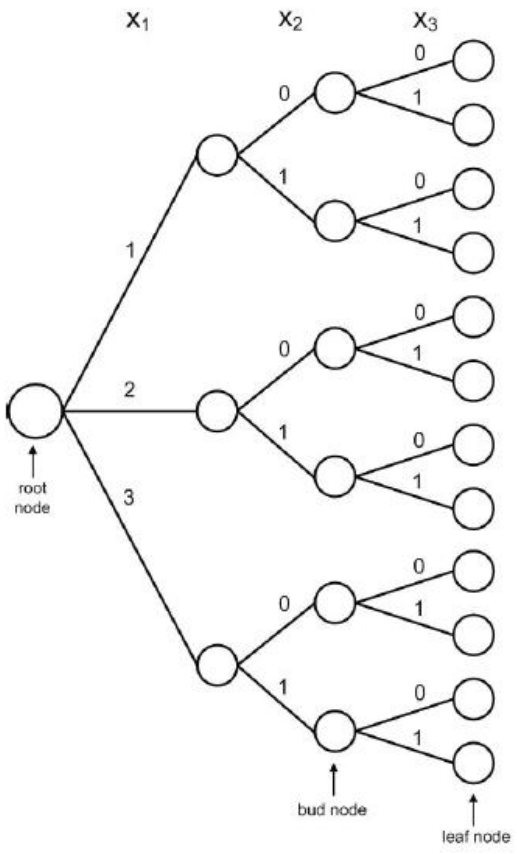


Figure 2.7. A full enumeration tree

The main idea in branch and bound is to avoid exploring the entire tree as much as possible, because it will just be too big in any real problem. Instead, branch and bound grows the tree in stages, exploring only the most promising nodes at any stage. It determines which node is the most promising by estimating a bound on the best value of the objective function that can be obtained by exploring that node in later stages. The name of the method originates from the *branching* that happens when a node is selected for further growth and the next generation of children of that node is created. The *bounding* originates in the bound on the best value attained that is estimated by growing a node. The method is applied in the hope that at the end only a very small fraction of the full enumeration tree will have grown.

Another important aspect of the method is *pruning*, in which nodes are cut off and permanently discarded when it can be shown that it, or any of its descendants, will never be either feasible or optimal. The name derives from gardening, in which pruning means to clip off branches on a tree, exactly what takes place in this case. Pruning is one of the most important aspects of branch and bound, since it is precisely what prevents the search tree from growing too much.

To describe branch and bound in detail, we first need to introduce some terminology:

- Node: any partial or complete solution. For example, a node that is two levels down in a 5-variable problem might represent the partial solution 3-17-?-?-?, in which the first variable has a value of 3 and the second variable has a value of 17. The values of the last three variables are not determined yet.
- Leaf (leaf node): a complete solution in which all of variable values are known.
- Bud (bud node): a partial solution, either feasible or infeasible. Think of it as a node that might yet grow further, just as on a real tree.

Branch and bound is a very general framework. To completely specify how the process is to proceed, we also need to define policies concerning selection of the next node, selection of the next variable, how to prune, and when to stop.

At any intermediate point in the algorithm, we have the current version of the branch and bound tree, which consists of bud nodes labelled with their bounding function values and other nodes labelled in various ways. The *node selection policy* governs how to choose the next bud node for expansion. There are three popular policies for node selection *best-first* or *global-best node selection*, *depth-first* and *breadth-first*.

Once a bud node has been chosen for expansion, the *variable selection policy* governs which variable to use in creating the child nodes of the bud node. There are few standard policies for variable selection. The variables are often selected just in their natural order, though a good variable selection policy can improve efficiency greatly.

We also need to establish policies and rules for pruning bud nodes. As mentioned above, there are two main reasons to prune a bud node: one can show that no descendent will be feasible, or that no descendent will be optimal.

Finally, we need a *terminating rule* to tell us when to stop expanding the branch and bound tree. To guarantee that we have reached optimality, we stop when the incumbent solution's objective function value is better than or equal to the bounding function value associated with all the bud nodes. This means that none of the bud nodes could possibly develop into a better solution than the complete feasible solution we already have in hand, so there is no point in expanding the tree any further. Of course, according to our policies for pruning, all bud nodes in this condition will already have been pruned, so this terminating rule amounts to saying that we stop when there

are no more bud nodes left to consider for further growth. This also proves that the incumbent solution is optimal.

Cutting Plane Methods

The term cutting-plane method is an umbrella term for optimization methods which iteratively refine a feasible set or objective function by means of linear inequalities, termed *cuts*. Such procedures are popularly used to find integer solutions to integer programming and MIP problems, as well as to solve general, not necessarily differentiable convex optimization problems. The use of cutting planes to solve MIP was introduced by Gomory (1958, 1960). However most experts, including Gomory himself, considered them to be impractical due to numerical instability, as well as ineffective because many rounds of cuts were needed to make progress towards the solution. Things turned around when in the mid-1990s Cornuejols and co-workers showed them to be very effective in combination with branch-and-cut and ways to overcome numerical instabilities. Nowadays, all commercial MILP solvers use Gomory cuts in one way or another. Gomory cuts, however, are very efficiently generated from a simplex tableau, whereas many other types of cuts are either expensive or even NP-hard to separate. Among other general cuts for MILP, most notably lift-and-project dominates Gomory cuts. Cutting-plane methods for general convex continuous optimization and variants are known under various names: Kelley's method, Kelley-Cheney-Goldstein method, and bundle methods.

Cutting plane methods for MIP work by solving a non-integer linear program, the linear relaxation of the given integer program. The theory of Linear Programming dictates that under mild assumptions (if the linear program has an optimal solution, and if the feasible region does not contain a line), one can always find an extreme point or a corner point that is optimal. The obtained optimum is tested for being an integer solution. If it is not, the existence of a linear inequality that *separates* the optimum from the convex

hull of the true feasible set, is guaranteed. Finding such an inequality is the *separation problem*, and such an inequality is a *cut*. A cut can be added to the relaxed linear program. Then, the current non-integer solution is no longer feasible to the relaxation. This process is repeated until an optimal integer solution is found.

Branch-and-Cut Techniques

Branch-and-cut (B&C) is a hybrid of branch and bound and cutting plane methods. It has proven to be effective in solving different combinatorial optimization problems, especially the traveling-salesman problem (TSP) and its variants (Bard et al. 2002). B&C methods have also been used to solve other combinatorial optimization problems. Problems addressed recently with cutting plane or branch-and-cut methods include the linear ordering problem, maximum cut problems, scheduling problems, network design problems, packing problems, the maximum satisfiability problem, biological and medical applications, and finding maximum planar subgraphs (Mitchell, 2001).

The linear program is solved without the integer constraint using the regular simplex algorithm. When an optimal solution is obtained, and this solution has a non-integer value for a variable that is supposed to be integer, a cutting plane algorithm is used to find further linear constraints which are satisfied by all feasible integer points but violated by the current fractional solution. If such an inequality is found, it is added to the linear program, such that resolving it will yield a different solution which is hopefully "less fractional". This process is repeated until either an integer solution is found (which is then known to be optimal) or until no more cutting planes are found.

In the latter case, the branch and bound part of the algorithm is started. The problem is split into two versions, one with the additional constraint that one fractional variable is greater than or equal to the next integer greater than its

current value, and one with the additional constraints that this variable is less than or equal to the next smaller integer. In this way new variables are introduced in the basis according to the number of basic variables that are non-integers in the intermediate solution but which are integers according to the original constraints. The new linear programs are then solved using the simplex method and the process is repeated until a solution satisfying all the integer constraints is found. During the branch and bound process, more cutting planes can be generated, which may be either *global cuts*, i.e., valid for all feasible integer solutions, or *local cuts*, meaning that they are satisfied by all solutions fulfilling the side constraints from the currently considered branch and bound subtree.

2.9.5 Preprocessing

Given a mathematical formulation, preprocessing refers to elementary operations that can be performed to enhance or simplify the formulation. The preprocessing phase often involves at least one of the following actions: (i) development of tightening constraints, (ii) addition of logical inequalities, (iii) fixing of variables, and (iv) removing redundant constraints. Actually, preprocessing can be considered as a phase between formulation and solution, and it can significantly improve the speed of a sophisticated mathematical framework that might, for instance, be unable to recognize the fact that some variables can be fixed and then be eliminated from the model (Nemhauser and Wolsey, 1999).

The most common preprocessing performed in mathematical models for the RCPSP is the definition of earliest and latest starting times (ES_i and LS_i, respectively) and earliest and latest finishing times (EF_i and LF_i, respectively) It should be noted that many mathematical formulations (i.e., time-indexed discrete-time models) for the RCPSP are highly sensitive to time horizon. The time-window preprocessing reduces: (i) the number of the time-indexed decision variables, and (ii) the total number of constraints needed.

2.10 Time representation

In all mathematical formulations, the representation of time is an important issue. Two main approaches are used: *discrete-time* and *continuous-time*, depending on whether the events of the schedule can only take place at some predefined time points, or can occur at any moment during the time horizon of interest, respectively.

Discrete-time models are based on: (i) dividing the scheduling horizon into a finite number of time intervals with predefined duration and, (ii) allowing the events such as the beginning or the end of tasks to take place only at the boundaries of these time periods (see Fig. 2.8a). Therefore, scheduling constraints have only to be monitored at specific and known time points, which reduces the problem complexity and makes the model structure simpler and easier to solve, particularly when resource limitations are taken into account. On the other hand, this type of problem simplification has two major disadvantages. First, the size of the mathematical model as well as its computational efficiency strongly depend on the number of time intervals postulated, which is defined as a function of the problem data and the desired accuracy of the solution. Note that a dense discretisation of the time horizon would increase the number of decision variables and lead to large combinatorial problems, which are hard to solve (or even intractable). Second, sub-optimal or even infeasible schedules may be generated because of the reduction of the domain of timing decisions. This time representation is an approximation of time, with the length of each interval usually set to be equal to the *greatest common factor* of activity processing times. Despite being a simplified version of the original scheduling problem, discrete formulations have proven to be very efficient, adaptable and convenient for a wide variety of industrial applications, especially in those cases where a reasonable number of intervals is sufficient to obtain the desired problem representation.

In order to overcome the previous limitations and generate data-independent models, a wide variety of optimisation approaches employ a *continuous-time* representation (see Fig. 2.8b). In these formulations, timing decisions are explicitly represented as a set of continuous variables defining the exact times at which the events take place. In the general case, a variable time handling allows obtaining a significant reduction in the number of variables of the model and at the same time, more flexible solutions in terms of time can be generated. Also, problems with non-integer activity durations could be modelled more accurately. This is of great importance for real-life problems, wherein processing times rarely have integer values. However, because of the modelling of variable processing times, resource limitations usually needs the definition of more complicated constraints involving many big-M terms, which tends to increase the model complexity and the integrality gap and may negatively impact on the capabilities of the method.

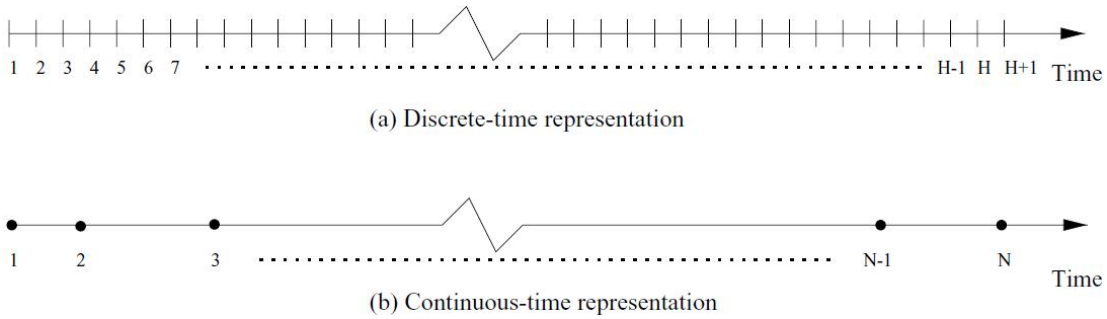


Figure 2.8. Time representations

2.11 Modelling and Solution Techniques

In this section we will review modelling and solution techniques for the Resource-Constrained Project Scheduling Problem and its variants. Blazewicz et al. (1983) have shown that solving the RCPSP is a strongly NP-hard problem and for some variants, such as the RCPSP/max, even the theoretically easier problem of checking its feasibility is NP-complete (Hartmann and Briskorn, 2010).

A number of solution methods, both *exact* and *approximate* have been proposed in the literature so far. *Exact* techniques found in the literature rely usually on mathematical programming formulations and specialised branch-and-bound algorithms. Due to the high degree of complexity of RCPSPs, a number of *approximate* methods such as heuristics and metaheuristics have also been proposed in the literature.

Roughly speaking, a *heuristic* is a technique designed to solve a problem, or find an approximate solution with few computational requirements, when classic methods fail to find the exact solution. By trading optimality, completeness, accuracy, and/or precision for speed, a heuristic can quickly produce a solution that is good enough for solving the problem at hand. The heuristic procedures broadly fall into two categories, namely constructive heuristics and improvement heuristics (Demeulemeester and Herroelen, 2002). *Constructive heuristics* start from an empty schedule and add activities one by one until a 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,* on the other hand, start from a feasible schedule that was obtained by some constructive heuristic. Operations are performed on a schedule which transform a solution into an improved one. These operations are repeated until a locally optimal solution is obtained. Steepest descent, fastest descent and iterated descent are various way of arriving in locally optimal solutions. Some improvement heuristics try to avoid getting stuck in a locally optimal solution by allowing an intermediate deterioration of the project makespan. In those cases, one has to avoid the phenomenon of cycling, i.e. repeatedly considering the same sequence of schedules. To that purpose, several metaheuristics like tabu search, simulated annealing and genetic algorithms have been proposed. A *metaheuristic* optimises a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. Metaheuristics make few or no assumptions about the problem being optimized and can search very large spaces of candidate

solutions. However, similar to heuristics, there is no guarantee on the quality of the solution obtained, and it is often impossible to tell how far the current solution is from optimality.

Finally, *hybrid* methods combine more than one solution technique, e.g. exact and heuristic, heuristic and metaheuristic, e.t.c.

We focus our review on solution procedures that aim at minimising the project makespan, which (as pointed out in section 2.6.1) is the most popular and important objective. For a thorough state-of-the-art, the interested reader is referred to the excellent works of Kolisch (1996a), Hartmann and Kolisch (2000), Demeulemeester and Herroelen (2002), Kolisch and Hartmann (2006), Hartmann and Briskorn (2010) and Węglarz et al. (2011).

2.11.1 RCPSP

The classical RCPSP belongs to the class of problems that are strongly NP-hard, as demonstrated by Blazewicz et al. (1983), and since the late 1950s most of the work in project scheduling has focused on developing solution techniques for it. Due to its generality, the classical RCPSP has attracted most of the project scheduling research effort. Our review will focus on the most important solution methods proposed in the literature.

Exact methods

Starting with *exact* methods for the RCPSP, a very early discrete-time mathematical model for the RCPSP was presented by Pritsker et al. (1969). This Binary Integer Programming (BIP) formulation was designed for multi-project scheduling, but can also be used for the single-project case. It is based on the definition of binary variables, which specify if an activity starts processing at a specific time point or not. Activities have a resource demand at the time point their processing starts, and no resource demand at the time

point of their completion. The MILP model accommodated a wide range of conditions and supported objectives for minimizing the makespan and minimizing the total lateness. No computational results are available for this early study.

Christofides et al. (1987) developed a formulation similar to Pritsker et al., which mainly differs in the formulation of the precedence constraints, with the new precedence constraints being disaggregated expressions of Pritsker's. They introduced CAT, a depth-first branch-and-bound procedure that generates a branch-and-bound tree, whose nodes correspond to semi-active feasible partial schedules. The procedure only branches to resolve a resource conflict. The reported computational results are on randomly generated problems involving up to 25 activities and 3 resources.

Kaplan (1988) developed a different time-indexed discrete-time BIP formulation for the preemptive version of the RCPSP, which can easily be adopted to the standard RCPSP. It is based on defining a single type of binary variables that specify if activities are in process in each time period, thus allowing a much simpler definition of the resource constraints. In this formulation, the dummy start and end activities are assigned durations equal to 1 which have to be considered when performing the forward and the backward pass, respectively. Moreover, transitive precedence relationships are introduced between activities.

The algorithm presented by Demeulemeester and Herroelen (1992, 1997a) is an extension of the depth-first branch and bound approach of Christofides et al. (1987). The new nodes in the enumeration tree are created by considering sets of activities which are delayed. Two dominance rules are used to prune the enumeration tree. The first one is a variation of the left-shift dominance rule. The second one makes use of a "cutset", i.e. unscheduled activities for which all predecessors belong to the partial schedule. Bounding is performed with the precedence-based and the critical-sequence-based lower bound used

by Stinson et al. (1978), and the weighted-node-packing-bound of Mingozzi et al. (1998).

Two formulations with an exponential number of variables are those of Alvarez-Valdés and Tamarit (1993) and Mingozzi et al. (1998), making them more useful when calculating lower bounds. The first one proposes a *continuous-time* formulation based on the definition of a set *IS* of all minimal *resource incompatible sets S*. A resource incompatible set is a set of activities between which no precedence relation exists, but which would violate the resource constraints, if performed in parallel. Their model is based on the definition of sequencing binary variables which define the sequencing of activities. The second one by Mingozzi et al. (1998), is a *discrete-time* BIP formulation based on *feasible subsets*, that is activities that can be simultaneously executed without violating resource or precedence constraints.

Klein (2000) proposed a discrete-time BIP model with precedence and resource constraints similar to those presented by Kaplan (1988). A single type of binary variables is used, specifying if an activity starts at the beginning of time point t or earlier.

Artigues et al. (2003) proposed a continuous MIP formulation based on sequencing and resource flow variables. Starting time continuous variables are defined for each activity. In addition, sequencing binary variables are introduced to define the sequence between any pair of activities. Finally, resource flow continuous variables are defined to denote the quantity of resource directly transferred from one activity (at its completion) to another (at the beginning of its processing). Notice that, in this mathematical formulation, the resource requirements of the dummy activities are set equal to the maximum available capacity, instead of zero. That way, the start dummy activity acts as a resource source while the end dummy activity plays the role of a resource sink.

Schmidt and Grossmann (1996) proposed a single mode, slot-based continuous-time formulation with no resource constraints for the optimal scheduling of testing tasks in the new product development process of an agricultural chemical or pharmaceutical company. In subsequent work, the focus changed to the development of realistic models that could be solved for large-scale problems. Jain and Grossmann (1999) extended Schmidt's work by including resource constraints, and Maravelias and Grossmann (2004) further extended that model by allowing the allocation of different levels of resources and capacity expansion. On the same problem of scheduling of clinical trials in the pharmaceutical research and development pipeline, Colvin and Maravelias (2008) developed a basic resource-constrained multi-stage stochastic programming formulation (MSSP) model which was extended in Colvin and Maravelias (2009) to account for outlicensing and resource planning including outsourcing. In their recent work, Colvin and Maravelias (2010) focused on the development of new results that lead to smaller MSSP mixed-integer programming (MIP) formulations and the development of a solution algorithm to address problems that cannot be generated using commercial tools. Papageorgiou, Rotstein, and Shah (2001) proposed a MILP model assuming that enough resources are always available. The formulation integrates the selection of both a product development and introduction strategy and a capacity planning and investment strategy. Levis and Papageorgiou (2004) proposed a mathematical model, which is an extension of the previous work, determining both the product portfolio and the multi-site capacity planning in the face of uncertain clinical trial outcomes while taking into account the trading structure of the company.

Schutt et al. (2009) presented a new exact approach solving the RCPSP by modelling cumulative constraints by decomposition and using lazy clause generation. Benchmarks from the PSPLib show the strong power of nogoods and Vsids style search to fathom a large part of the search space. The

approach competes with specialised RCPSP solution approaches and closed 63 open problems.

Recently, Koné et al. (2011) introduced two different event-based continuous-time MIP formulations, the *Start/End Event*-based (SEE) formulation, which is a variant of the event-based formulation proposed by Zapata et al. (2008), and the *On/Off Event*-based (OOE) formulation. The decision variables used, are indexed by event points instead of time points, which correspond to start or end times of activities. The variables in such formulations are fewer than in time-indexed ones, since they are not a function of the time horizon. The authors compared the event-based formulations with three other formulations issued from the literature, two of which (Pritsker et. al., 1969 and Christofides et al., 1987) use time-indexed variables, and a third formulation (Artigues et al., 2003) that uses sequential variables. The computational results proved that the formulation proposed by Christofides et al. (1987) yields better results for exact solving on traditional test instances. The event-based formulations (more particularly the OOE) and the one by Artigues et al. (2003), have the advantage of solving more easily the instances involving very large scheduling horizons. Finally, their OOE formulation consistently outperformed the SEE on all tested instance sets.

Approximate methods

A number of different *approximate* procedures have been developed for the RCPSP. *Priority-rule-based scheduling* is made up of two components: A *schedule generation scheme* (SGS) and a *priority rule*. Two different schemes for the generation of feasible schedules can be distinguished (see Kolisch, 1996a): the *serial* and the *parallel* method, respectively. Both generate a feasible schedule by extending a partial schedule in a stage-wise fashion. In each stage the generation scheme forms the set of all schedulable activities, called the decision set. A specific priority rule is then employed in order to choose one activity from the decision set which will be scheduled. While the

decision set of the serial method is made up of all currently unscheduled activities whose predecessors have already been scheduled, the parallel method defines the set by including all the precedence- and resource-feasible unscheduled activities which can be started at the schedule time. Depending on the number of passes performed, and hence the number of schedules generated, single- and multi-pass approaches can be distinguished. A number of such heuristics along with a computational study are reported by Kolisch (1996a).

Truncated branch-and-bound methods do not explore the entire enumeration tree, instead only a partial exploration is performed. An example of such a method is the work of Alvarez-Valdes and Tamarit (1989) that makes use of the enumeration tree as presented in Christofides et al. (1987), i.e., nodes consist of sets of activities which have to be delayed. Instead of enumerating all offspring nodes, the heuristic implicitly or explicitly chooses one node.

Kochetov and Stolyar (2003) devised an evolutionary algorithm which combines genetic algorithm, path relinking, and tabu search. Solutions are evolved and diversified in a genetic way. Evolution is done by choosing two solutions from the pool and constructing the path of solutions linking the selected solutions (path relinking). The best solution from the path is chosen and improved via tabu search. The latter employs a neighborhood where the activity list is divided in three parts. For the first and the last part the serial SGS is employed while for the mid part the parallel SGS is used. The best solution from the tabu search is added to the population and the worst solution is removed from the population.

Debels et al. (2006) proposed a new metaheuristic that combines elements from scatter search, a generic population-based evolutionary search method, and from a recently introduced heuristic method for the optimisation of unconstrained continuous functions based on an analogy with electromagnetism theory. The computational experiments show that the

procedure is capable of producing consistently good results for challenging instances of the resource-constrained project scheduling problem and that the algorithm outperforms previous state-of-the-art existing heuristics.

Debels and Vanhoucke (2008) presented a new genetic algorithm (GA) that is able to provide near-optimal heuristic solutions. This GA procedure is also extended by a so-called decomposition-based genetic algorithm (DBGA) that iteratively solves subparts of the project. The authors present computational experiments on two data sets. The first benchmark set is used to illustrate the performance of both the GA and the DBGA. The second set is used to compare the results with current state-of-the-art heuristics and to show that the procedure is capable of producing consistently good results for challenging problem instances. The GA outperforms all state-of-the-art heuristics and the DBGA further improves the performance of the GA.

Paraskevopoulos et al. (2012) proposed a new solution representation and an evolutionary algorithm for solving the RCPSP. The representation scheme is based on an ordered list of events, which are sets of activities that start (or finish) at the same time. The proposed solution methodology, namely SAILS, operates on the event list and relies on a scatter search framework. The latter incorporates an Adaptive Iterated Local Search (AILS) as an improvement method, and integrates an event-list based solution combination method. AILS utilizes new enriched neighborhoods, guides the search via a long term memory and applies an efficient perturbation strategy. Computational results on benchmark instances of the literature indicate that both AILS and SAILS produce consistently high quality solutions, while the best results are derived for most problem data sets.

2.11.2 Multi-mode Resource Constrained Project Scheduling Problem (MRCPSP)

When the solution has to additionally determine the assignment of modes (MRCPSP), further complexity is added since the solution search space is enlarged. The MRCPSP is NP-hard in the strong sense being a generalization of the RCPSP. Moreover, for more than one non-renewable resources, even the problem of finding a feasible solution is NP-complete (Kolisch, 1995).

Exact methods

Exact methods for the MRCPSP include the formulation of Talbot (1982), which considers the objective of minimizing the makespan under a given budget, and minimizing the total non-renewable resource consumption under the presence of a project due date.

Sprecher et al. (1997) extended the enumeration scheme of Demeulemeester and Herroelen (1992) for the single-mode to the multi-mode RCPSP. Hartmann and Drexl (1998) generalised the exact procedure of Stinson et al. (1978) to the multi-mode context.

Zhu et al. (2006) proposed a branch-and-cut method based on the Christofides et al. formulation (1987) for the MRCPSP. The primary contribution of their research aimed at the development of several techniques for accelerating the solution process, including variable reduction, cut generation, and bound tightening. In addition, a high-level search strategy referred to as local branching was applied to find good feasible solutions in the early stages of the computations. Their work gave very competitive results on benchmark problems and remains the best-performing exact method to date.

Sabzehparvar et al. (2008) presented a continuous-time formulation for the *MRCPSP with Generalized Precedence Relations* (MRCPSP-GPR) *with mode-dependent minimal or maximal time lags*. The proposed model has been inspired by the rectangle packing problems, but does not require a feasible solution to start. Their computational study consisted of a set of 60 test problems.

In a recent paper from the process systems industry, Zapata et al. (2008) developed 3 different MILP models to address large-scale *Multi-mode Resource Constrained Multi-Project Scheduling Problems* (multi-mode RCMPSP), using continuous divisible resources and continuous time. The "curse" of dimensionality of the problem (indexing of task execution modes, indexing of time periods, and the discrete character of the resources), limits the exact solution of the formulations to very small systems. Each of the 3 proposed models handles the time domain in a different way, but the computational results show that despite the theoretical advantages of the strategies used they are limited to problems in the same range of applicability of conventional multi-mode formulations.

Approximate methods

Starting with *approximate* solution procedures, Hartmann (2001) presented an efficient genetic algorithm, with the genetic encoding based on a precedence feasible list of activities and a mode assignment. After defining the related crossover, mutation, and selection operators, they used a local search extension to improve the schedules found by the basic genetic algorithm.

Alcaraz et al. (2003) proposed another genetic algorithm with slightly worse performance than that of Hartmann, but with a better fitness function.

Jarboui et al. (2008) propose a combinatorial Particle Swarm Optimisation (CPSO) algorithm that first generates an assignment of modes to activities which is called particle and then uses a local search to optimize the sequences when a new assignment is made.

Van Peteghem and Vanhoucke (2010) designed a genetic algorithm which uses two populations, one with left-justified schedules and one with right-justified schedules. They also introduced an extended serial schedule generation scheme, which improves the mode selection by choosing that feasible mode of a certain activity that minimises the finish time of that activity. According to their computational results, their algorithm outperforms all prior algorithms in literature, within reasonable computation times.

Coelho and Vanhoucke (2011) developed a new algorithm that splits the problem type into a mode assignment and a single mode project scheduling step. The mode assignment step is solved by a satisfiability (SAT) problem solver and returns a feasible mode selection to the project scheduling step. The project scheduling step is solved using an efficient meta-heuristic procedure from the literature for the RCPSP. Computational results show that the procedure can report similar or sometimes even better solutions than the algorithm of Van Peteghem and Vanhoucke (2010), although it often requires a higher CPU time. This makes the considered genetic algorithm approach the most powerful heuristic developed to date.

2.11.3 RCPSP/max

The addition of maximal time lags to the classical RCPSP significantly increases the complexity of the problem. Moreover, the generation of problem instances could be problematic because infeasible problems might be generated. The RCPSP/max is an NP-hard problem and even the theoretically easier problem of checking its feasibility is NP-complete (Hartmann and Briskorn, 2010).

The RCPSP/max has been mostly studied under the objective of minimising the project makespan. Non-regular objectives functions, such as resource levelling and net present value problems, are also considered in various works (Neumann and Zimmermann 1999, 2000 and Rieck et al., 2012). For an extensive review of project scheduling problems with time windows, the interested reader is referred to Neumann, Schwindt and Zimmermann (2002, 2003).

Exact methods

Most exact solution procedures for the RCPSP/max in the literature are usually branch-and-bound algorithms.

The early work of Bartusch et al. (1988), mainly aimed at the mathematical properties of the problem, but also proposed a B&B procedure that extends the set of precedence relations to eliminate all reduced forbidden sets in an initial time-feasible solution. The computational results are limited to a single bridge construction project.

Demeulemeester and Herroelen (1997b) extend their DH-procedure to the Generalized Resource-Constrained Project Scheduling Problem (GRCPSP), which only deals with minimal time lags.

De Reyck and Herroelen (1998) proposed a hybrid depth-first/laser beam search B&B algorithm. Schwindt (1998a) delayed activities by adding special precedence constraints (i.e., disjunctive precedence constraints) between sets of activities, unlike the previous two methods which used activity pairs.

Fest et al. (1998) proposed a similar approach, but instead of introducing precedence constraints, the resource conflicts were resolved only locally by a dynamic update of job release dates.

Dorndorf et al. (2000) presented a time-oriented B&B algorithm that uses constraint-propagation techniques which actively exploit the temporal and resource constraints of the problem in order to reduce the search space.

Recently, Bianco and Caramia (2012b) proposed a B&B algorithm that utilises a new mathematical formulation, which uses a discrete-time approach with one continuous and two binary variables. Continuous variable x_{it} represents the percentage of activity i executed till the end of time period t. Binary variables s_{it} / f_{it} assume value *1* for every $t \geq \tau$, where τ is the time activity i started/finished, and value *0* otherwise. Lower bounds are calculated through a Langrangian relaxation of the former model. The extensive computational comparison with known lower bounds and exact methods, displays the efficiency of their algorithm in the calculation of bounds and solutions.

Schutt et al. (2012) use lazy clause generation, i.e., a hybrid of finite domain and Boolean satisfiability solving, in order to apply no-good learning and conflict-driven search to the solution generation. They close 573 open problem instances and generate better solutions in most of the remaining instances. Surprisingly this exact method outperforms the published non-exact methods on benchmarks, in terms of the quality of solutions.

Approximate methods

Some representative *approximate* approaches are mentioned below. Franck and Neumann (1998) propose a two-step method: (a) a sophisticated decomposition analysis is performed to identify critical sub-components which can be scheduled independently, and (b) the scheduled sub-components (partial schedules) are integrated into one using a set of priority rules.

Cesta et al. (2002) presented an algorithm based on a constraint satisfaction problem solving search procedure. Cicirello and Smith (2004) described a

framework for integrating multiple heuristics within a stochastic sampling search algorithm, and validated it on the RCPSP/max. Smith (2004) developed a non-systematic hybrid that combines squeaky wheel optimisation with an effective window-based conflict resolution mechanism.

Finally, Ballestin et al. (2011) proposed an evolutionary algorithm that utilises the double justification technique of Valls et al. (2005) to improve solutions generated in the evolutionary process.

2.12 Modelling and Optimisation Software

Solving mathematical programming problems without the use of a computer is extremely hard even for small cases. Available commercial software and hardware have evolved greatly the last few years, reducing running times, computational costs and solving larger problems. Examples of commercial modelling and optimisation software with common features are GAMS, AIMMS, AMPL, Gurobi and ILOG. GAMS is the most commonly used tool in the Process Systems Engineering community, with which we attempt to exchange methods in this thesis. The rest of this section presents the GAMS system and CPLEX solver that were selected to solve the optimisation problems developed.

2.12.1 General Algebraic Modelling System (GAMS)

The General Algebraic Modelling System (GAMS) is a high-level modelling system for mathematical optimisation. It is designed for modelling and solving linear, nonlinear, and mixed-integer optimisation problems. GAMS was the first algebraic modelling language (AML) and is formally similar to commonly used fourth-generation programming languages. It is available for use on various computer platforms and the models are portable from one platform to another.

GAMS allows the users to implement a sort of hybrid algorithms combining different solvers in a seamless way. Models are described in concise algebraic statements which are easy to read, both for humans and machines. Although initially designed for applications related to economics and management science, it has a large community of users from various backgrounds of engineering and science.

GAMS contains an integrated development environment (IDE) which is connected to a group of third-party optimisation solvers, such as BARON, COIN solvers, CONOPT, CPLEX, DICOPT, GUROBI, MOSEK, SNOPT, and XPRESS. The GAMS system is tailored for complex, large-scale modelling applications and allows the user to build large maintainable models that can be adapted to new situations.

According to Rosenthal (2012) and Castillo, Conejo, Pedregal, García, and Alguaci (2001), some of the more remarkable features of GAMS algebraic modelling language are:

- The model representation is analogous to the mathematical description of the problem. Therefore, learning GAMS programming language is almost natural for those working in the optimisation field. Additionally, GAMS is formally similar to commonly used programming languages.
- Models are described in compact and concise algebraic statements which are easy for both humans and machines to read.
- The modelling task is completely apart from the solving procedure. Once the model of the system in question has been built, one can choose among the diverse solvers available to optimise the problem.
- Allows changes to be made in model specifications simply and safely.
- Allows unambiguous statements of algebraic relationships.
- Permits model descriptions that are independent of solution algorithms.
- All data transformations are specified concisely and algebraically. This means that all data can be entered in their most elemental form and

that all transformations made in constructing the model and in reporting are available for inspection.

- The ability to model small size problems and afterwards transform them into large-scale problems without significantly varying the code.
- Decomposition algorithms can be programmed in GAMS by using specific commands, thus not requiring additional software.
- GAMS imports/exports data from/to Microsoft EXCEL. Additionally, GAMS can be easily linked with MATLAB (The Mathworks, 1998) using the matgams library (Ferris, 1999) if some special data manipulation is needed.

2.12.2 CPLEX Solver

IBM ILOG CPLEX, which is often informally referred to simply as CPLEX, is an optimisation solver package. The CPLEX Optimiser was named after the C programming language in which it was implemented and the Simplex method. However, today it provides additional methods for mathematical programming and offers interfaces for other languages such as C++, C#, and Java, Python (through the C interface). Additional connectors to Microsoft Excel and MATLAB are also provided. The CPLEX Optimiser is accessible through independent modelling systems such as AIMMS, AMPL, GAMS, MPL, OpenOpt, OptimJ and TOMLAB. Specifically for GAMS, GAMS/CPLEX is a solver that allows users to combine the high level modelling capabilities of GAMS with the power of CPLEX optimisers.

The CPLEX Optimiser is designed to solve large, difficult problems quickly and with minimal user intervention. Access is provided (subject to proper licensing) to solution algorithms for linear, integer, quadratically constrained and mixed integer programming problems.

While numerous solving options are available, GAMS/CPLEX automatically calculates and sets most options at the best values for specific problems. It is

worth mentioning that for problems with integer variables, CPLEX uses a branch-and-cut algorithm which solves a series of LP subproblems. Because a single mixed integer problem generates many subproblems, even small MIP problems can be very computationally intensive and require significant amounts of physical memory.

2.13 Concluding Remarks

In this chapter, various aspects of the RCPSP and its variants have been reviewed, along with test instances commonly used by researchers for benchmarking. Also, major optimisation techniques and tools used throughout this thesis have been presented. The main concepts beneath each method have been briefly described in order to provide the reader with a general understanding of the theory involved into the solution approaches. Finally, exact and approximate solution procedures for the RCPSP from the literature have been thoroughly reviewed.

At this point, it is worth noticing that the process of building a mathematical model is often considered to be as important as solving it because this process provides insight about how the system works and helps organise essential information about it. Models of the real world are not always easy to formulate because of the richness, variety, and ambiguity that exists in the real world or because of our ambiguous understanding of it. As a result, building up concise, useful and efficient mathematical models/approaches is a very difficult and challenging task and this thesis has placed particular attention towards this direction.

Chapter 3

RTN-based MILP Formulations for Single- and Multi-Mode Resource-Constrained Project Scheduling Problems

This chapter presents new mixed-integer linear programming models for the deterministic single- and multi-mode resource-constrained project scheduling problem with renewable and non-renewable resources. The modelling approach relies on the Resource-Task Network (RTN) representation, a network representation technique used in process scheduling problems, based on continuous time models. First, we propose new RTN-based network representation methods, and then we efficiently transform them into mathematical formulations including a set of constraints describing precedence relations, different types of resources and multiple objectives. Finally, the applicability of the proposed formulations is illustrated using several example problems under the most commonly addressed objective, the makespan minimization.

3.1 Introduction

The main aim of this chapter is to extend and therefore apply modelling and network representation techniques used in the process industry to RCPSP and MRCPSP problems.

A general project consists of a set of interconnected activities and resources, logically linked. These activities usually have to be performed for a successful project completion. However, there may exist alternatives for some activities, which vary in aspects such as duration, cost and required sets of resources. Since resources impose restrictions on project scheduling, they must be

included in the general project description. Logical links and precedence constraints for activities must also be incorporated.

Traditionally networks included only two different types of nodes for the description of the logic links and dependencies: *activities* and *decision boxes* (Eisner 1962, Minieka 1978). For generalised projects, *resources* should be included as an additional element. The overall structural components of a general project that must be described before a feasible schedule is computed are:

1. Activity: A project comprises of several activities. Each activity is typically described by its duration and a set of resources. These resources can be used throughout the entire activity, consumed at the beginning of the activity or produced at its end. Each activity is considered non-preemptive (once started, it must be performed to completion).
2. Resource: Each resource is defined by its initial amount and its maximum availability throughout the entire project.
3. Decision box: A decision box is characterised by the conditions placed on the activities entering it and by the conditions on the activities emanating from it. A decision box provides logical links between project activities. Assuming I_{in} to be the number of *input activities* of such a decision box, the following different cases exist:
   - all activities (I_{in}) entering the decision box must be performed,
   - exactly x activities entering the decision box must be performed, with $1 \leq x < I_{in}$,
   - at least x activities entering the decision box must be performed, with $1 \leq x < I_{in}$.

   The first is a special case of the second, since exactly *all input activities must be performed* is the same with *exactly* I_{in} *activities*

must be performed. This enables us to group the *all* input condition with the *exactly* x, by allowing x to take the value I_{in}.

The input condition must be satisfied for some of the activities emanating from the decision box to be performed. A decision box may have two different *output logics*:

- at least one activity emanating from the decision box can be performed,
- exactly one activity emanating from the decision box can be performed.

4. Project end: The simplest way of describing the project end is as a special decision box with one or more activities entering it.

The chapter is organised as follows. In section 3.2, a new network representation of project scheduling problems based on the RTN concept is presented. In sections 3.3 and 3.4 we propose new MILP formulations for the RCPSP and MRCPSP, including precedence relations, different types of resources, time, and other constraints. Afterwards, the applicability of the proposed formulations on several project scheduling problems is demonstrated in section 3.5. Finally, concluding remarks are drawn in section 3.6.

3.2 A new network representation for the RCPSP

The RTN process representation although simple, can describe a very wide variety of process scheduling problems (Pantelides, 1994). Indeed, it has been extensively used in the process scheduling literature (e.g. Castro et al. 2001, 2004). Its bipartite directed graph for general processes consists of resources, represented as circles and tasks/activities, represented as rectangles. A task/activity consumes and/or produces a set of resources that can be anything from raw materials, intermediate and final products to manpower and processing equipment. A new network representation method

is proposed, based on the RTN, that can express more complex activity precedence relations, than the AoN and AoA representations.

3.2.1 Conversion of General Projects to RTN form

Creating a mathematical formulation based on the RTN, requires the definition of a framework that converts the general project characteristics aforementioned to their equivalent RTN components. Some conventions are easier than others; for example, *Activities correspond to Tasks* and *Project Resources to RTN Resources*. On the other hand, *Decision Boxes* are more complex and they will be modelled as Resources, with specific values on their following parameters:

1. the activity consumption and production coefficients (μ_{ri} and $\overline{\mu}_{ri}$) and
2. the minimum and maximum excess levels ($R_r^{\min}$ and $R_r^{\max}$)

Special consideration will be given to Project End translation. The Project End is treated as a special Decision Box, which in turn is converted to an RTN Resource.

Precedence Constraints

Projects consist of coordinated and controlled activities with start and finish times. To achieve the required execution sequence, precedence constraints on each activity are imposed. These constraints will be modelled indirectly in the proposed representation, by adding a new type of resource, called *Logical* and adjusting the quantities consumed and produced by each activity.

So, besides the physical resource requirements, each activity requires one unit of the logical resource assigned to it. This unit is produced by the activity's immediate predecessor(s) in equal quantities and is addressed by properly adjusting the production coefficients ($\overline{\mu}_{ri}$) for the preceding activities

and the consumption coefficient (μ_{ri}) for the succeeding activity, as shown in Figure 3.1.

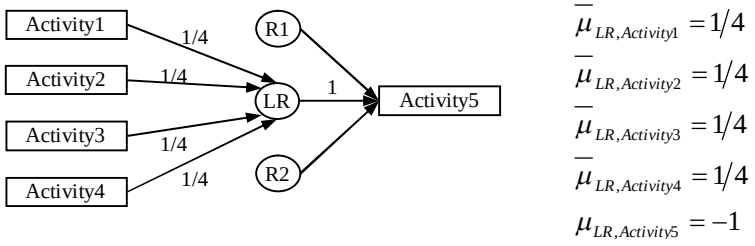


Figure 3.1. Example of the use of Logical Resources

General Resource Management

Resource limitations can extend the project execution time, as they often disallow parallel execution for activities that require the same resource(s). Different activities may require the same resource, so we must define a minimum and maximum resource quantity available throughout the duration of the project, as well as the initial amount.

Let's consider the example in Figure 3.2, in which the number of tasks that can be executed simultaneously depends on the number of available Operators. If only one operator is available, then:

$$R_{OP}^{initial} = 1$$

$$R_{OP}^{\min} = 0$$

$$R_{OP}^{\max} = 1$$

and only one task can be active at any given time. On the other hand, if two operators are present, Cleaning and Settling can be performed simultaneously, making the Blender available for use sooner.

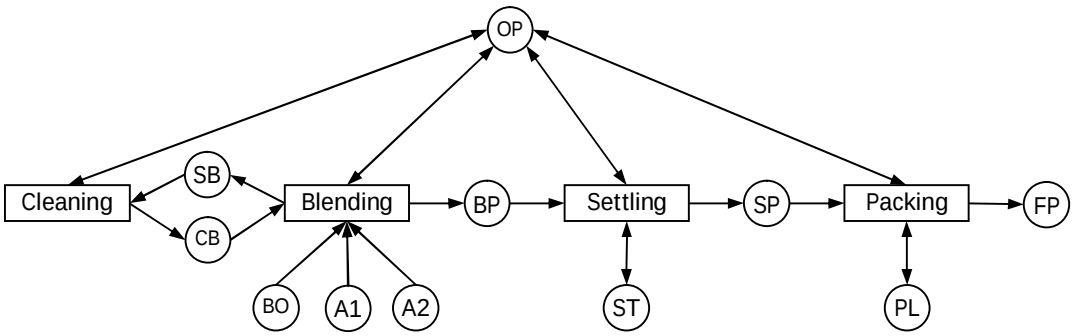


Figure 3.2. Graphic Representation of a Resource Task Network

Converting Decision Boxes to Resources

A decision box is used to represent complex interactions among activities. For example, activities having more than one immediate predecessors or alternate activities, that must be modelled properly to create a feasible schedule. The RTN formulation equips tasks and resources to model processes. Project activities and resources are easily represented as tasks and resources, respectively. Decision boxes though, are more complex, as they are logical components of a project.

Decision boxes will be represented as a special type of resources, called *logical resources*, with specific modifications to some of their coefficients.
Before moving on, we must define a number of sets:

I_r the set of all activities interacting with logical resource r

I_r^{in} the set of input activities of logical resource r

I_r^{out} the set of output activities of logical resource r

with I_r^{in} and I_r^{out} being disjoint sets, as an activity cannot be both input and output to a decision box:

$$I_r = I_r^{in} \cup I_r^{out} \quad \text{and} \quad I_r^{in} \cap I_r^{out} = \varnothing$$

Let IN_r and OUT_r be the number of input and output activities, respectively, to logical resource *r*. The minimum excess resource $R_r^{\min}$ for logical resources is set to zero $R_r^{\min} = 0$, as they are initially not available in any amount and we can assume that they may or may not be produced throughout the duration of the project.

On the other hand, the maximum excess resource $R_r^{\max}$ and the activity consumption and production coefficients (μ_{ri} and $\overline{\mu}_{ri}$) values, depend on the type of decision box they represent.
Starting from the input conditions, we distinguish two cases:

- *Exactly* x inputs.

  We set $R_r^{\max} = 1$ and $\overline{\mu}_{ri} = 1/x, \forall i \in I_r^{in}$. The first constraint, limits the excess resource of *r* to 1.

  The second one, defines that each input activity ($i \in I_r^{in}$) produces exactly $1/x$ quantity of logical resource *r*, so that when all x activities are performed, one unit of *r* will be produced.
- *At least* x *inputs.*

  We set $R_r^{\max} = IN_r / x$ and $\overline{\mu}_{ri} = 1/x, \forall i \in I_r^{in}$, allowing any number of activities to start, but at least x of them must be completed, before one unit of logical resource *r* is produced.

Consider the example in Figure 3.3, where a decision box/logical resource has five activities entering, and exactly 3 of them must be performed. The values of the coefficients for resource *r* would be $R_r^{\max} = 1$ and $\overline{\mu}_{ri} = 1/3, \forall i \in I_r^{in}$. So, when three activities have completed, the amount of resource *r*, produced would be $1/3 + 1/3 + 1/3 = 1$. And due to the restriction $R_r^{\max} = 1$, no other activity would be allowed to execute.

Similarly, if the condition was at least 3, then $R_r^{\max} = 5/3$ and $\overline{\mu}_{ri} = 1/3, \forall i \in I_r^{in}$, then to produce one unit of resource r, at least three activities would have to complete. The maximum quantity $R_r^{\max} = 5/3$, would be achieved, if all activities were executed.

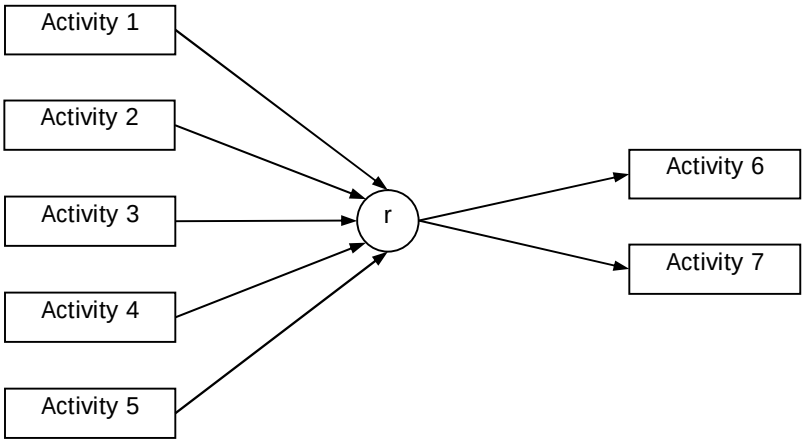


Figure 3.3. Production of Logical Resource

As far as the output conditions are concerned, if we allow only one output activity of I_r^{out} to start after reaching the necessary amount $R_r^{\max}$ (case exactly 1), we specify the consumption value of the output activities to $\mu_{ri} = -1$. For the previous example in Figure 3.3, the highest possible value for the excess resource r is $5/3$. We have two possible output activities, 6 and 7, but only one of them is allowed to start. The start limit is 1, but as soon as 6 or 7 start, it is reduced by 1 and no second start is possible.

For a possible start of several output tasks, we have to define one logical resource for each output task, as in Figure 3.4. The production of each of these resources is similar to what was described previously.

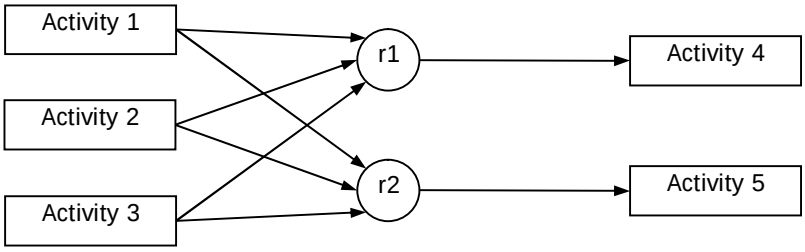


Figure 3.4. Production of Logical Resources for several output activities

After the input conditions are satisfied, resources *r1* and *r2* are available in their necessary amounts and the output tasks can be performed.

The possible combinations for input and output conditions are:

- Exactly x – exactly one
- At least x – exactly one
- Exactly one – at least one
- Exactly x – at least one
- At least x – at least one

These possible combinations can be described by the two networks of Figures 3.5 and 3.6.

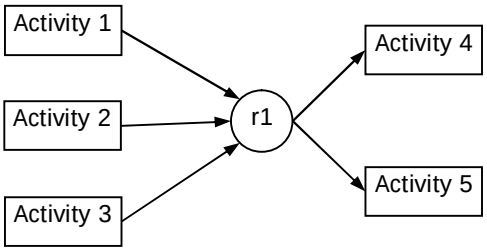


Figure 3.5. Conditions Exactly x/At least x - Exactly one and Exactly one - At least one

A decision box with *output* condition *Exactly 1*, is represented using one only one new logical resource. This case corresponds to Figure 3.5. A decision box with *input* condition *Exactly 1*, can also be represented as depicted in Figure 3.5, but we have to distinguish among different output conditions. If the

output condition is also *Exactly 1*, we set the consumption value to $\mu_{ri} = -1$, to allow only one output activity to start. If the output condition is *At least 1*, we set the consumption value to $\mu_{ri} = -1/OUT_r$ so that every output activity receives part of the available amount 1.

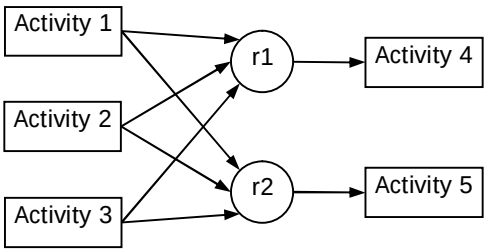


Figure 3.6. Conditions Exactly x/At least x - At least one

For the other cases we define OUT_r logical resources, as in Figure 3.6. Each input activity contributes equally to the production of every logical resource. Since the output condition is at least 1, we have to allow the start of every output activity after the satisfaction of the input condition. To model this, each output activity has its own logical input resource, which it can consume entirely. So, we set the consumption value of all output activities to $\mu_{ri} = -1$.

Another case is to allow some activities to start immediately after their predecessors without any delay. This is possible only between tasks whose interaction is known in advance (case A*ll*E*xactly 1*). It can be achieved by not allowing the excess resource of r to reach values greater or equal to 1 $\left(R_r^{\max} = 1 - \left(1/IN_r\right)\right)$. This forces the consumption of the logical resource to take place at the same time as the production of the last amount $1/IN_r$.

A summary of the parameter values for the various conditions of decision boxes is given in Table 3.1.

Table 3.1 Parameter values for decision boxes

Input Condition	Output Condition	$\bar{\mu}_{ri} \forall i \in I_r^{in}$	$\mu_{ri} \forall i \in I_r^{out}$	$R_r^{\max}$	Representation
Exactly x, $1 \le x \le IN_r$	Exactly 1	$1/x$	-1	1	Figure 3.5
At least x, $1 \le x \le IN_r$	Exactly 1	$1/x$	-1	IN_r/x	Figure 3.5
Exactly 1	At least 1	1	$-1/OUT_r$	1	Figure 3.5
All	1 Immediate	$1/IN_r$	-1	$1-(1/IN_r)$	Figure 3.5
Exactly x, $1 < x \le IN_r$	At least 1	$1/x$	-1	1	Figure 3.6
At least x, $1 \le x \le IN_r$	At least 1	$1/x$	-1	IN_r/x	Figure 3.6

There are two special subcases that require additional mechanisms to enforce the required logical dependencies. The first such case occurs when *Exactly x* input activities must be executed. The value $R_r^{\max} = 1$ is used to ensure that no more than x input activities are performed. And it does so until an output activity consumes a certain quantity of r and the excess resource becomes less than 1. This will allow another input activity to execute, violating the *Exactly x*, condition.

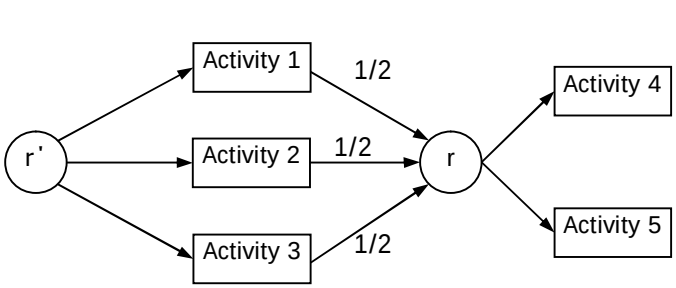
To resolve this problem, a new resource r' is introduced as input to each input activity $i \in I_r^{in}$ with

$$\mu_{r'i} = -1$$

Hence, in order for one of the input tasks to execute, a unit amount of resource r' is required. If the available quantity of this resource is limited to x, no more than x activities are allowed to start. This can be done by setting:

$$R_{r'}^{initial} = x$$

An example of this case for 3 activities entering a decision box with an *Exactly 2 – Exactly 1* condition is depicted in Figure 3.7.



Case

Exactly 2 – Exactly 1

$x = 2,$

$IN_r = 3,$

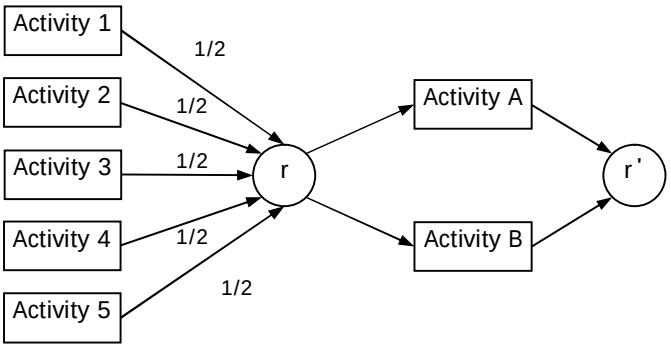
$R_{r'}^{initial} = 2,$

$\mu_{r',Activity1} = \mu_{r',Activity2} =$

$\mu_{r',Activity3} = -1$

Figure 3.7. Special case 1 example Exactly x with $x < IN_r$

The second special condition occurs for decision boxes with input logic *At least x* and output logic *Exactly one*. An example for this case is given in Figure 3.8, with logic At least 2 – Exactly 1.



Case

At least 2 – Exactly 1

$x = 2,$

$IN_r = 5,$

$R_{r',Final}^{\max} = 1,$

$\overline{\mu}_{r',ActivityA} = \overline{\mu}_{r',ActivityB} = 1$

Figure 3.8. Special case 2 example At least x-Exactly 1 with $IN_r > 2x$

The problem arises when $IN_r > 2x$. If more than $2x$ input activities do take place, then they will produce 2 or more units of excess resource. This, in turn, will allow more than one output activity to be executed, which is contrary to the intention of the decision box. This problem can be easily overcome by making each output activity $i \in I_r^{out}$ produce a unit amount of a new resource r' (i.e. $\overline{\mu}_{r'i} = 1, \forall i \in I_r^{out}$) and setting:

$$R_{r',Final}^{\max} = 1$$

Since the new resource r' is not consumed by any activity in the project RTN, this immediately implies that no more than one activity $i \in I_r^{out}$ may be executed.

3.2.2 Project End Formulation

Projects are completed successfully, after all required activities have been performed and all constraints satisfied. A project may end in several ways:

- One activity must be performed.
- More than one activities must be performed.
- Alternative final activities exist.
- More complex logical conditions.

The project end is modelled as a decision box/logical resource with no output activities. This resource can only be produced and not consumed. Completion of the project, calls for the production of this logical resource in the required quantities.

To make sure that the project end resource is produced, lower and upper bounds are imposed on its initial, overall and final excess quantities $R_r^{initial}$, $R_r^{\min}$, $R_r^{\max}$, $R_{r,Final}^{\min}$ and $R_{r,Final}^{\max}$. For intermediate resources, the overall and final values are identical. Figure 3.9 depicts an example of the simple case where the project end requires the execution of a single final activity.

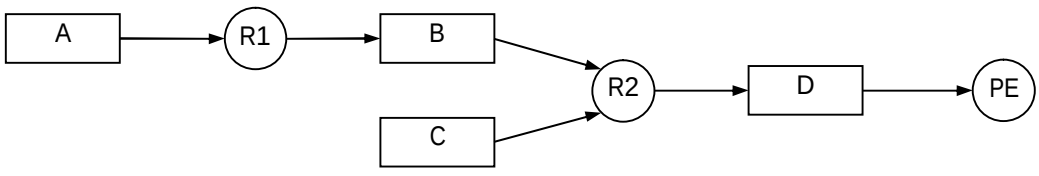


Figure 3.9. An example of an RTN representation of a project

The successful completion of the project in the example requires the execution of activity D. It can be modelled by setting a lower bound of 1 on the final excess quantity of PE. The complete set of data values is:

$$R_{PE}^{initial} = 0$$

$$R_{PE}^{\min} = 0, R_{PE}^{\max} = 1$$

$$R_{PE,Final}^{\min} = R_{PE,Final}^{\max} = 1$$

$$\overline{\mu}_{PE,D} = 1$$

Project End with more than one final activity

Usually, projects require the completion of more than one activity. In such cases, we consider the *Project End* (PE) resource to be a decision box with input logic *All*. Suppose the set of activities to be performed is *F* and it contains *NF* activities.

Each activity contributes an amount of $1/NF$ of PE:

$$\overline{\mu}_{PE,i} = 1/NF, \forall i \in F$$

By demanding the minimum and maximum final excess resources of PE to be 1, all input activities are enforced to be executed. The rest of data values are the same as the case with one activity:

$$R_{PE}^{initial} = 0$$

$$R_{PE}^{\min} = 0$$

$$R_{PE}^{\max} = 1$$

$$R_{PE,Final}^{\min} = 1$$

$$R_{PE,Final}^{\max} = 1$$

$$\overline{\mu}_{PE,i} = 1/NF, \forall i \in F$$

An example of modelling the project end in a project with 4 final activities is illustrated in Fig. 3.10.

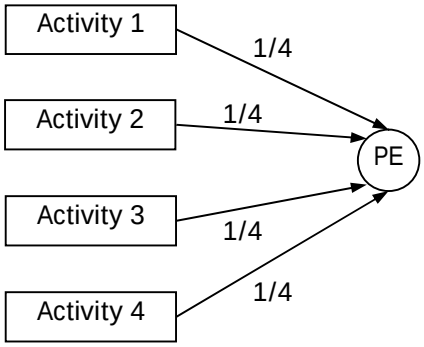
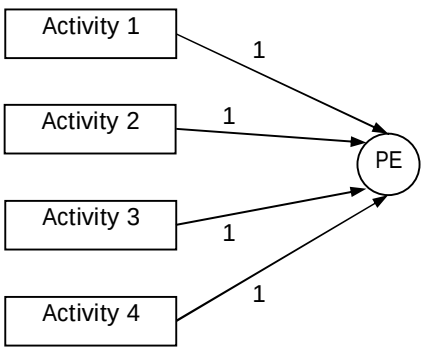


Figure 3.10. Project End with 4 activities

Project End with alternative final activities

Having examined the case of multiple required final activities, we move on to another one. It is possible to have alternatives among final activities, meaning that project completion can be achieved by performing only one of them. This is equivalent to the input condition *Exactly 1* in a decision box.

Therefore, a logical final resource PE, with no output conditions is introduced. Each alternate input activity i is set to produce $\overline{\mu}_{PE,i} = 1$ quantity of PE, and the maximum final excess resource is set to 1, $R^{\max}_{PE,Final} = 1$. This allows only one activity to produce this resource, but by itself is not enough. To ensure that Exactly 1 of PE, and no less, is produced, we also require that $R^{\min}_{PE,Final} = 1$. For an illustrative example see Figure 3.11.



Case

4 alternate activities

$R^{\max}_{PE,Final} = 1,$

$R^{\min}_{PE,Final} = 1$

$\overline{\mu}_{PE,Activity1} = \overline{\mu}_{PE,Activity2} =$

$\overline{\mu}_{PE,Activity3} = \overline{\mu}_{PE,Activity4} = 1$

Figure 3.11. Example of Project End with 4 alternate activities

Project End with general conditions

To provide a complete formulation of projects using the RTN, more complex conditions that identify project completion must be considered. These could involve a combination of disjunctions (logic exclusive OR - XOR) and/or conjunctions (logic AND).

We can initially try to represent such formulations as a logical tree with circles corresponding to operators and rectangles to activities. The root node, which is also modelled as a circle, is the condition for project completion. An example of such a tree is shown in Figure 3.12.

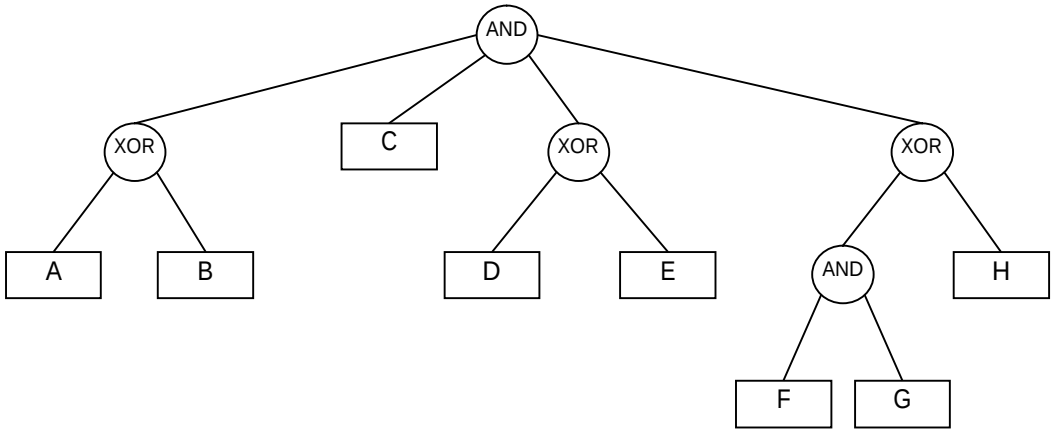


Figure 3.12. Example of logical tree

The example corresponds to the condition, that the project will be completed when all of the following are satisfied:

- complete either A and B, and
- complete C, and
- complete either D or E, and
- complete either F and G, or H

We can write this succinctly as:

(A XOR B) AND C AND (D XOR E) AND ((F AND G) XOR H)

It is relatively easy to transform the logical tree into an RTN. Conjunctions of activities can be translated by creating a new output logical resource for each

activity with an *Exactly 1 – Exactly 1* logic. These resources are required by a conjunction activity in equal unit amounts, to produce the final logical resource. The translation process for the F AND G conjunction of Figure 3.12 is depicted in Figure 3.13.

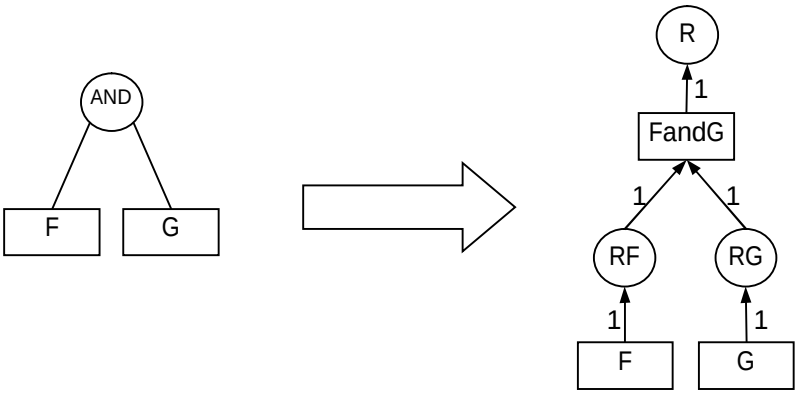


Figure 3.13. Transformation of conjunctions to RTN

Let's examine the complete data set per activity, for the previous example:

- For activity F:

$$\overline{\mu}_{RF,F} = 1$$

$$R_{RF}^{initial} = 0$$

$$R_{RF}^{\min} = 0$$

$$R_{RF}^{\max} = 1$$

- For activity G:

$$\overline{\mu}_{RG,G} = 1$$

$$R_{RG}^{initial} = 0$$

$$R_{RG}^{\min} = 0$$

$$R_{RG}^{\max} = 1$$

- For activity FandG:

$$\mu_{RF,FandG} = -1$$

$$\mu_{RG,FandG} = -1$$

$$\overline{\mu}_{R,FandG} = 1$$

$$R_{R}^{initial} = 0$$

$$R_{R,Final}^{\min} = 1$$

$$R_{R,Final}^{\max} = 1$$

Disjunctions are much easier to represent, as they are, essentially, a decision box with an *Exactly 1* input logic. Figure 3.14, illustrates an example of such a disjunction of two activities.

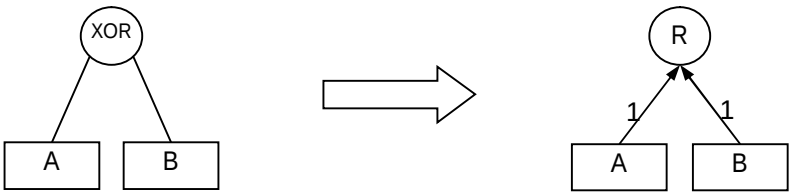


Figure 3.14. Transformation of disjunctions to RTN

The data set for this transformation is:

$$\overline{\mu}_{A,R} = 1$$

$$\overline{\mu}_{B,R} = 1$$

$$R_R^{initial} = 0$$

$$R_R^{\min} = 0$$

$$R_R^{\max} = 1$$

$$R_{R,Final}^{\min} = 1$$

$$R_{R,Final}^{\max} = 1$$

Using these transformations we can convert the logical tree of Figure 3.12, to its equivalent RTN of Figure 3.15, with all arrows corresponding to unit consumption/production of resource.

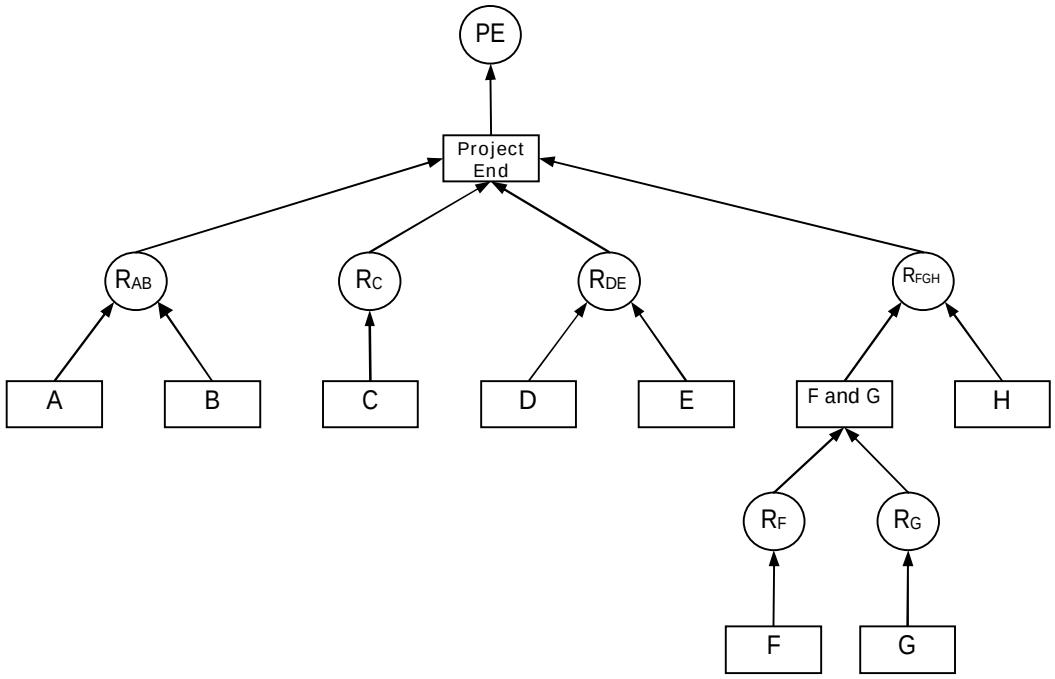


Figure 3.15. Equivalent RTN for logical tree in Figure 3.12

3.3 MILP Formulation for the Single-Mode RCPSP

The Single-mode RCPSP consists of scheduling the project activities under specific precedence and resource constraints, while minimizing the project makespan. The mathematical model, proposed in this section, is based on the time-slot synchronised formulation introduced by Schilling and Pantelides (1996), where the variable time horizon H, is divided into T slots with variable time duration. Although, the RTN representation is simple and elegant for process scheduling, it can become even more simplified when employed for project scheduling.

It should be emphasised, that the underlying formulation of Schilling and Pantelidis (1996) is not the best performing RTN-based formulation in the literature, but it has been merely chosen to illustrate the applicability and potential of the proposed RCPSPs representation. More efficient formulations include the improved continuous-time RTN formulation of Castro et al. (2001) that relaxes the non-linear timing constraint producing an easier to solve,

pure MILP problem and also the work of Castro et al. (2004), where a new set of timing constraints further reduces the computational cost. The main objective of this section is to establish a new modelling framework for RCPSPs utilising techniques from the process scheduling area. The representation of RCPSPs could provide the basis for translations into more efficient RTN formulations than the original work of Schilling and Pantelidis (1996).

The complete list of the notation used throughout this chapter is given in Section 3.7. We assume that no more than one instance of an activity can be executed at any time point. This assumption converts the original RTN variable N_{it}, representing the number of activity instances starting at time t, from integer to binary, since the only possible values are 0 and 1, depending on whether none or one instance of the activity is being executed.

Additionally, resource consumption and production does not depend on the size of the activity, so the corresponding size-dependent coefficients' values are 0, $v_{ri} = \bar{v}_{ri} = 0$.

Activities can be executed only once over the duration of the project. If an activity is to be executed again, we represent it as a new one. This is because it is bound to have different constraints, or else it would be included in the first one, requiring double input and producing double output resources.

3.3.1 Constraints

Schilling and Pantelides (1996) distinguish four types of constraints: timing, slot, excess resource balances and excess resource capacity constraints. We will adapt these constraints to project scheduling problems.

Two binary variables y_{it} and $\bar{y}_{it}$ are defined to express starting time and spanning over consecutive time slots. The first set of binary variables y_{it},

replace the RTN variables N_{it} and take a value of 1 if activity i starts at t or 0 otherwise. The second new variable $\bar{y}_{it}$, takes a value of 1 if activity i is active over both slots t and t-1 or 0 otherwise. The model can be simplified by setting $\bar{y}_{i,1} = 0$, since no task can be active over $t = 0$, as it is out of the scheduling horizon. Using the $\bar{y}_{it}$ variable (activity spanning over two adjacent slots), instead of $y_{itt'}$ (activity spanning over multiple slots) used in the original RTN, results in a simpler mathematical model with far fewer binary variables than the original one.

Timing Constraints

The total duration of all time slots τ_t must be equal to the time horizon H:

$$\sum_{t=1}^{T} \tau_t = H \tag{1}$$

A project activity i may extend over one or more consecutive time slots and the sum of the durations of these slots must be equal to the duration θ_i of the activity. Using the new variables y_{it} and $\bar{y}_{it}$, instead of N_{it} and $y_{itt'}$ the timing constraints of the general RTN formulation (Schilling and Pantelides, 1996) are transformed to:

$$\sum_{t=1}^{T} \left(y_{it} + \bar{y}_{it} \right) \tau_t = \theta_i \sum_{t=1}^{T} y_{it}, \forall i \tag{2}$$

We note that constraint (2) involves nonlinearities since both the binary variables y_{it} and $\bar{y}_{it}$ and the slot durations τ_t are decision variables. These nonlinearities can be removed using standard techniques (Glover, 1975). Considering that at most one instance of task i can be active at any time, we define the new variables $\tau lin_{it} \equiv \left(y_{it} + \bar{y}_{it} \right) \tau_t$. This definition can be effected using the following linear constraints:

$$\tau^{\min}\left(y_{it}+\bar{y}_{it}\right)\le \tau lin_{it} \le \min\left[\tau^{\max},\theta_i\right]\left(y_{it}+\bar{y}_{it}\right),\forall i,t \tag{3}$$

$$\tau_t-\tau^{\max}\left(1-y_{it}-\bar{y}_{it}\right)\le \tau lin_{it} \le \tau_t,\forall i,t \tag{4}$$

where $\tau^{\min}$ and $\tau^{\max}$ are the minimum and maximum slot durations, respectively. We can set the maximum slot duration to be equal to the value of the greatest activity duration $\theta_i^{\max}$ and the minimum duration equal to 1. After applying this linearization technique, constraint (2) becomes linear:

$$\sum_{t=1}^{T}\tau lin_{it}=\theta_i\sum_{t=1}^{T}y_{it},\forall i \tag{5}$$

Slot Constraints

Variable $\bar{y}_{i,t+1}$ can take a value of 1 only if activity i started at an earlier time slot ($<t+1$) and is still active over $t+1$. For an activity to start at an earlier time, one of the variables y_{it} or $\bar{y}_{it}$ must take a value of 1. To ensure proper activity execution over multiple time slots, we now need both new variables y_{it} and $\bar{y}_{it}$, instead of $y_{itt'}$ and this is expressed as:

$$\bar{y}_{it}+y_{it}\ge\bar{y}_{i,t+1},\forall i,t\in[1,T-1] \tag{6}$$

If activity i has not been active over slot t, the value of all three variables must be 0. The inequality is necessary for the case that activity i is actually completed at the end of slot t. The possible combinations for the values of the previous variables are shown on Table 3.2. Notice that combinations not allowed by constraints (6) are not displayed.

Table 3.2. Combination of values for $\overline{y}_{i,t-1} + y_{i,t-1} - \overline{y}_{it}$

$\overline{y}_{i,t-1}$	$y_{i,t-1}$	$\overline{y}_{it}$	$\overline{y}_{i,t-1} + y_{i,t-1} - \overline{y}_{it}$	Production
0	0	0	0	No amount of *r* is produced, since activity *i* is not executed
0	1	0	1	An amount of *r* is produced, since activity *i* started and completed execution over *t-1*
0	1	1	0	No amount of *r* is produced, since activity *i* started at *t-1*, but is still active.
1	0	0	1	An amount of *r* is produced, since activity *i* finished at *t-1*
1	0	1	0	No amount of *r* is produced, since activity *i* is active from at least *t-2* and is still active

Another important constraint expresses that an activity can be executed at most once over the time horizon H:

$$\sum_{t=1}^{T} y_{it} \leq 1, \forall i \tag{7}$$

Excess Resource Balances

The balance for every resource r at slot boundary t considers all starting and ending tasks I_r interacting with the resource r. It adds the changes at time point t to the excess amount $R_{r,t-1}$ over the previous slot $t-1$ in order to obtain the amount of excess resource over the new slot t:

$$R_{r,t} = R_{r,t-1} + \sum_{i \in I_r} \left[\mu_{ri} y_{it} + \overline{\mu}_{ri} \left(\overline{y}_{i,t-1} + y_{i,t-1} - \overline{y}_{it} \right) \right], \forall r, t \in [1, T+1] \tag{8}$$

The first term $\mu_{ri} y_{it}$ in the summation represents the amount of resource r consumed by starting activities, while the second one corresponds to the amount produced. For $t = 0$, $R_{r,0}$ is the quantity of resource r initially available $R_r^{initial}$. To deal with renewable resources, we set the production coefficient of all activities requiring it equal to the consumption coefficient. Thus we express that the amount of renewable resources consumed (used) by an activity, is released upon its completion.

Excess Resource Capacity Constraints

Since the availability of resources is limited, we have to introduce an upper and lower bound on their capacity. During the entire time horizon H, the actual excess amount of any resource r has to lie between these boundaries:

$$R_r^{\min} \le R_{rt} \le R_r^{\max}, \forall r, t \tag{9}$$

Additionally, we introduce similar boundaries for the excess amount of each resource r at the end of the project:

$$R_{r,Final}^{\min} \le R_{r,T+1} \le R_{r,Final}^{\max}, \forall r \tag{10}$$

Objective Function

The objective function for the proposed formulation aims at *minimizing the project duration* and is expressed as:

$$Minimize \quad H$$

3.3.2 Improvement to the formulation

Despite its simplicity and compactness, the proposed mathematical model can be improved to achieve better computational performance. Considering the various aspects of project scheduling, we can set tighter bounds on variables and reduce the size of the sets that participate in the constraints.

Slot bounds for activities

The initial model, assumes that all activities can be executed over any time slot. Given the existence of precedence constraints, this is not a realistic approach. Consider the example displayed in Figure 3.16. Activity C is

preceded by A and cannot be executed before A finishes. This sets a lower bound of 2 on C's starting time slot.

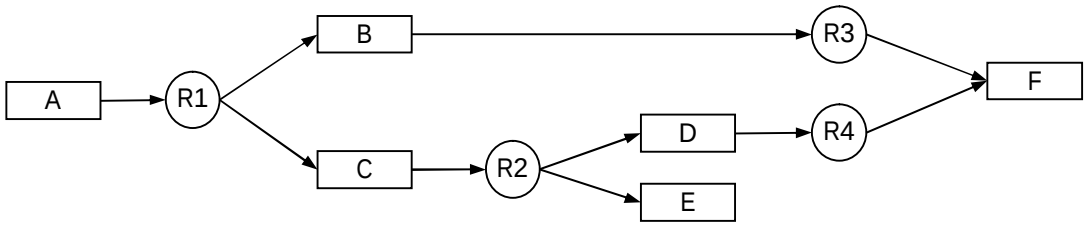


Figure 3.16. Slot boundaries example

We can set proper lower and upper bounds on the starting time slots of the activities which allow successful project completion and at the same time improve the computational performance of our model. For an activity i, these bounds can be calculated using the number of maximum preceding PA_i and succeeding SA_i decision boxes:

$$tl_i = 1 + PA_i, \forall i$$

$$tu_i = T - SA_i, \forall i$$

We always consider the worst case for these bounds, to avoid eliminating a feasible solution. This means that in our example $tl_F = 4$ and not 3, since the worst case requires that activities A, C and D are performed and not A, B (activity E can be executed in parallel with D, so it does not count).

Let assume that the number of time slots, for the example in Figure 3.16 is *T=5*. Table 3.3 contains the appropriate lower and upper bounds for this case.

This improvement reduces the number of y_{it}, $\overline{y}_{it}$ and τ_{it} variables, as well as the related constraints. Without slot bounds one would have to define a number of y_{it} and τ_{it} variables equal to $Number\ of\ Activities\ x\ T$. This number

of variables (*6x5 = 30* for this example) could be reduced to 14, as given in Table 3.3, which is 46.66% of the original figure. Similarly for the binary variable $\bar{y}_{it}$, the initial number of variables is equal to *Number of Activities x (T-1) = 6x4 =24*, as it is defined per pair of time slots (t and $t-1$) for each activity. Therefore, the total number of $\bar{y}_{it}$ variables after applying the improvement is $\sum_i (tu_i - tl_i) = 8$.

Table 3.3. Improved time slot bounds

Activity i	A	B	C	D	E	F
PA_i	0	1	1	2	2	3
SA_i	3	1	2	1	0	0
tl_i	1	2	2	3	3	4
tu_i	2	4	3	4	5	5
$tu_i - tl_i + 1$	2	3	2	2	3	2
$\sum_i (tu_i - tl_i + 1)$	14					

In the case of alternative preceding activities or complex project completion conditions, this procedure becomes more complicated.

Mandatory Activities

Projects contain various activities that may or may not be performed. For mandatory activities I_{sure}, such as end-activities, we can further tighten the formulation, by replacing constraint (7) by:

$$\sum_{t=1}^{T} y_{it} = 1, \forall i \in I_{sure} \tag{11}$$

Exploiting, constraint (11), we can also transform (5) to:

$$\sum_{t=1}^{T} \tau lin_{it} = \theta_i, \forall i \in I_{sure} \tag{12}$$

Alternative Activities

Further improvements can be achieved in case alternative activities exist. Assume sets I_{ALT_X} with only one of their alternatives being performed through the entire project. For these activities, the slot constraint (7) can be transformed to:

$$\sum_{I_{ALT_x}} \sum_{t} y_{it} \leq 1, \forall x \tag{13}$$

This results in a reduction of $\left(N_{ALT_x} - 1\right) \times T$ for the constraint, with N_{ALT_x} denoting the number of activities in set ALT_x.

The proposed model for RCPSPs, which is named SMRTN, consists of constraints (1), (3) – (13) and the objective function represents the total project duration.

3.3.3 Using the RCPSP formulation in MRCPSPs

The formulation for RCPSPs (SMRTN model) introduced in this section can also be used for MRCPSPs by disaggregating the multi-mode activities into different activities. The activity modes are modelled using decision boxes with multiple inputs, exactly one of which is to be executed. An example of how activities with multiple modes are modelled is shown in Figure 3.17.

Activity A can be performed in 3 modes which we represent as different activities called A1, A2 and A3. Aside from physical resources, the activity requires 1 unit of logical resource LRA ($\mu_{LRA,A1} = \mu_{LRA,A2} = \mu_{LRA,A3} = -1$). To ensure that only 1 mode is executed, the initial quantity of LRA is limited to 1 ($R_{LRA}^{initial} = 1$).

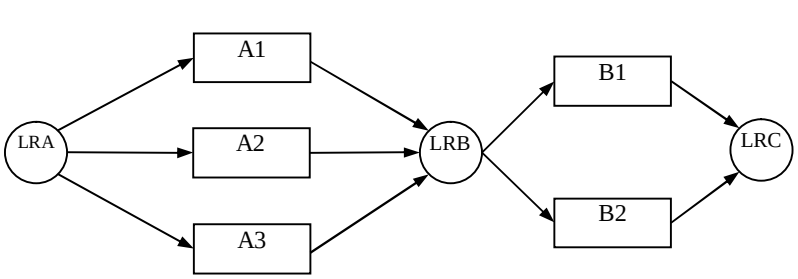


Figure 3.17. Modelling activities with multiple modes

Upon completion, activity A produces 1 unit of LRB ($\bar{\mu}_{LRB,A1} = \bar{\mu}_{LRB,A2} = \bar{\mu}_{LRB,A3} = 1$). Similarly, by properly adjusting the consumption coefficients of activity B, to $\mu_{LRB,B1} = \mu_{LRB,B2} = -1$ we allow only one of its modes to perform.

This extension for MRCPSPs, which is named MMRTN1, consists of constraints (1), (3) – (13) and the objective function represents the total project duration.

3.4 MILP Formulation for the MRCPSP

The standard Multi-Mode RCPSP requires sequencing the project activities, so that the precedence constraints are met, determining the execution mode for each activity, meeting the resource constraints and minimizing the project duration.

In this section we propose a MILP formulation for the MRCPSP. The formulation is an extension of the RCPSP introduced in the previous section and is also based on the RTN.

3.4.1 Constraints

The formulation consists of five types of constraints, Timing, Slot, Excess Resource Balances, Excess Resource Capacity and Task Operation

constraints. As in the RCPSP formulation, we can simplify the model, by setting $\bar{y}_{i,1} = 0$, since no task can be active over $t = 0$.

Timing Constraints

The total duration of all time slots must be equal to the time horizon:

$$\sum_{t=1}^{T} \tau_t = H \tag{14}$$

A project activity i may extend over one or more consecutive time slots and the sum of the durations of these slots must be equal to the duration θ_{mi} of activity i in mode m:

$$\sum_{t=1}^{T} \left(y_{it} + \bar{y}_{it} \right) \tau_t = \sum_{m} \theta_{mi} z_{mi}, \forall i \tag{15}$$

where z_{mi} is a binary variable that expresses whether alternate mode m of activity i is chosen (1) or not (0).

The linearization of the left of constraint (15) can be performed as described in Section 3.3.1 We define new variables $\tau lin_{it} \equiv \left(y_{it} + \bar{y}_{it} \right) \tau_t$, through the linear constraints:

$$\tau^{\min} \left(y_{it} + \bar{y}_{it} \right) \leq \tau lin_{it} \leq \min \left[\tau^{\max}, \max(\theta_{mi}) \right] \left(y_{it} + \bar{y}_{it} \right), \forall i, t \tag{16}$$

$$\tau_t - \tau^{\max} \left(1 - y_{it} - \bar{y}_{it} \right) \leq \tau lin_{it} \leq \tau_t, \forall i, t \tag{17}$$

where τ^{min} and τ^{max} are the minimum and maximum slot durations, respectively. We can set the maximum slot duration to be equal to the value of the greatest activity duration $\theta_{mi}^{\max}$. After applying this linearization technique, constraint (15) becomes linear:

$$\sum_{t=1}^{T} \tau lin_{it} = \sum_{m} \theta_{mi} z_{mi}, \forall i \tag{18}$$

Slot Constraints

As in the RCPSP formulation, variable $\overline{y}_{i,t+1}$ can take a value of 1 only if activity i started at an earlier time slot ($< t+1$) and is still active over $t+1$. For an activity to start at an earlier time one of the variables y_{it} or $\overline{y}_{it}$ must take a value of 1. These constraints can be expressed by the following inequality:

$$\overline{y}_{it} + y_{it} \geq \overline{y}_{i,t+1}, \forall i, t \in [1, T-1] \tag{19}$$

If activity i has not been active over slot t, the value of all three variables must be 0. The inequality is necessary for the case that activity i is actually completed at the end of slot t.

An activity can be executed at most once over the time horizon H:

$$\sum_{t=1}^{T} y_{it} \leq 1, \forall i \tag{20}$$

and if so, only one activity mode is performed:

$$\sum_{m} z_{mi} = \sum_{t=1}^{T} y_{it}, \forall i \tag{21}$$

Excess Resource Balances

The balance of a resource r at slot boundary t is given by:

$$R_{r,t} = R_{r,t-1} + \sum_{i \in I_r} \left[-\left(S_{ri}^{L} + S_{ri}\right) y_{it} + \left(\overline{S}_{ri}^{L} + \overline{S}_{ri}\right)\left(\overline{y}_{i,t-1} + y_{i,t-1} - \overline{y}_{it}\right) \right], \forall r, t \tag{22}$$

The μ_{ri} and $\overline{\mu}_{ri}$ coefficients of the RCPSP formulation, are now replaced by the minimum resource consumption and production values S_{ri}^{L} and $\overline{S}_{ri}^{L}$ plus their surplus variables S_{ri} and $\overline{S}_{ri}$ respectively.

The values of S_{ri}^{L}/S_{ri}^{U} coefficients representing the minimum/maximum quantities of resource *r* required by the various modes *m* at the beginning of activity *i*, are given by:

$$S_{ri}^{L} = \min\left(a_{imr}\right) \text{ and } S_{ri}^{U} = \max\left(a_{imr}\right)$$

Similarly, the values of the minimum/maximum quantities of resource *r* produced by the various modes *m* at the end of activity *i*, $\overline{S}_{ri}^{L}/\overline{S}_{ri}^{U}$ are given by:

$$\overline{S}_{ri}^{L} = \min\left(\overline{a}_{imr}\right) \text{ and } \overline{S}_{ri}^{U} = \max\left(\overline{a}_{imr}\right)$$

Surplus variables S_{ri} and $\overline{S}_{ri}$ depend on the selected mode *m*, are bounded by constraints (30) and (31) and calculated through (32) and (33), as presented in the Task Operation Constraints section below.

The excess resource balance contains a type of nonlinearities similar to constraint (15), which can be removed by introducing two new variables:

$$S_{rit} \equiv y_{it} S_{ri} \text{ and } \overline{S}_{rit} \equiv \left(\overline{y}_{i,t-1} + y_{i,t-1} - \overline{y}_{it}\right)\overline{S}_{ri}$$

The first variable S_{rit} is defined:

$$0 \le S_{rit} \le \left(S_{ri}^{U} - S_{ri}^{L}\right) y_{it}, \forall t, (ri) \tag{23}$$

$$\sum_{t} S_{rit} = S_{ri}, \forall (ri) \tag{24}$$

Similarly for the second variable $\overline{S}_{rit}$:

$$0 \le \overline{S}_{rit} \le \left(\overline{S}^{U}_{ri} - \overline{S}^{L}_{ri}\right)\left(\overline{y}_{i,t-1} + y_{i,t-1} - \overline{y}_{it}\right), \forall t, (ri) \tag{25}$$

$$\sum_{t} \overline{S}_{rit} = \overline{S}_{ri}, \forall (ri) \tag{26}$$

Now we can replace (22) with:

$$R_{r,t} = R_{r,t-1} + \sum_{i \in I_r}\left[-S^{L}_{ri} y_{it} - S_{rit} + \overline{S}^{L}_{ri}\left(\overline{y}_{i,t-1} + y_{i,t-1} - \overline{y}_{it}\right) + \overline{S}_{rit}\right], \forall r, t \tag{27}$$

Excess Resource Capacity Constraints

Since resources are limited in their availability, we have to introduce an upper and lower bound on their capacity. During the whole time horizon H, the actual excess amount of any resource r has to lie between these boundaries:

$$R^{\min}_{r} \le R_{rt} \le R^{\max}_{r}, \forall r, t \tag{28}$$

Additionally, we introduce similar boundaries for the excess amount of each resource r at the end of the project:

$$R^{\min}_{r,Final} \le R_{r,T+1} \le R^{\max}_{r,Final}, \forall r \tag{29}$$

Task Operation Constraints

The surpluses of resource r consumed and produced by activity i are bound to:

$$0 \le S_{ri} \le S^{U}_{ri} - S^{L}_{ri}, \forall (r, i) \tag{30}$$

$$0 \le \overline{S}_{ri} \le \overline{S}^{U}_{ri} - \overline{S}^{L}_{ri}, \forall (r, i) \tag{31}$$

For logical resources S^{L}_{ri} and $\overline{S}^{L}_{ri}$ are equal to 0.

The surpluses S_{ri} and $\overline{S}_{ri}$ consumed and produced, depending on the selected mode, are calculated by:

$$\sum_m \left(a_{imr} - S_{ri}^L\right) z_{mi} = S_{ri}, \forall r, i \tag{32}$$

$$\sum_m \left(\overline{a}_{imr} - \overline{S}_{ri}^L\right) z_{mi} = \overline{S}_{ri}, \forall r, i \tag{33}$$

Due to constraints (32) and (33), variables z_{mi} are not required in the Excess Resource Balances constraint (22), thus reducing the number of binary variables involved.

Objective Function

The minimization of the project duration is the optimisation goal, similar to the previous model.

3.4.2 Improvement to the formulation

The extended formulation proposed in this section can achieve better computational performance using the improvement techniques of Section 3.3.2. To account for the new variables introduced, we need to clarify a few points regarding each constraint.

Slot bounds for activities

Multi-mode problems include activities that can be executed in various modes, possibly with different durations. The slot bounds improvement is used to define lower and upper slot bounds on activity starting and finishing time slots, respectively. These bounds are calculated using precedence relations between activities and not their durations. Therefore, the improvement can also be applied to the multi-mode case, as it is not affected

by the varying activity duration caused by the multiple modes. As a result, we achieve a reduction in the number of y_{it}, $\overline{y}_{it}$, τ_{it}, S_{rit}, $\overline{S}_{rit}$ variables and related constraints.

Mandatory Activities

This improvement is used for project activities that must be performed. For such activities, in the multi-mode case, we can replace constraint (20) with constraint (11), as in the single-mode case:

$$\sum_{t=1}^{T} y_{it} = 1, \forall i \in I_{sure} \tag{11}$$

However, in contrast to the single-mode case, we cannot exploit constraint (11) to transform timing constraint (19), due to the introduction of variables z_{mi}. Instead, we can use it to replace constraint (21) for mandatory activities, with:

$$\sum_{m} z_{mi} = 1, \forall i \in I_{sure} \tag{34}$$

Alternative Activities

An MRCPSP can contain sets I_{ALTx} of activities with only one of their alternatives being executed. For these activities, we can transform slot constraint (20) to constraint (13), similarly to the single-mode case:

$$\sum_{\mathrm{I}_{ALT_X}} \sum_{t} y_{it} \leq 1, \forall i \tag{13}$$

This model for MRCPSPs, which is named MMRTN2, consists of constraints (11), (13), (14), (16) – (21), (23) – (34), and the objective function represents the total project duration.

3.5 Computational Results

In this section, we consider a typical MRCPSP with all typical modelling aspects and then solve a number of RCPSP and MRCPSP problems using the proposed formulations to illustrate their applicability. The following notation is used for a consistent reference to the proposed formulations:

- SMRTN – the formulation used for RCPSPs, presented in Section 3.3
- MMRTN1 – the RCPSP formulation with the modifications in Subsection 3.3.3 used for MRCPSPs, and
- MMRTN2 – the formulation used for MRCPSPs, presented in Section 3.4.

All formulations were solved on an Intel Core 2 Quad Q8300 @ 2.5GHz and 4GB RAM using CPLEX 11.1.1 (GAMS Development Corporation, 2007) via a GAMS 22.8.1 (Rosenthal, 2012) WIN 6007.6015 VIS interface.

3.5.1 Example problem

In this section, we consider a typical MRCPSP from PSPLIB (Kolisch et al., 1996). The test instance used is j10 2_2 with 10 activities, each having 3 possible execution modes. Table 3.4 displays the number of modes, the number of successors and precedence relations between activities. Activities 1 and 12 are dummy activities representing the start and end of the project.

Table 3.4. Precedence Relations for test instance j10 2_2

Activity Number	Nr. of modes	Nr. of successors	Successors
1	1	3	2 3 4
2	3	2	5 6
3	3	2	10 11
4	3	1	9
5	3	2	7 8
6	3	2	10 11
7	3	2	9 10
8	3	1	9
9	3	1	12
10	3	1	12
11	3	1	12
12	1	0	-

Activity durations and resource requests for each mode are provided in Table 3.5. The maximum project duration is 86, and we calculate it by summing the maximum mode duration per activity.

Table 3.5. Project Mode Requests/Durations

Activity	Mode	Duration	Resource			
			R1	R2	N1	N2
1	1	0	0	0	0	0
2	1	3	6	0	9	0
	2	9	5	0	0	8
	3	10	0	6	0	6
3	1	1	0	4	0	8
	2	1	7	0	0	8
	3	5	0	4	0	5
4	1	3	10	0	0	7
	2	5	7	0	2	0
	3	8	6	0	0	7
5	1	4	0	9	8	0
	2	6	2	0	0	7
	3	10	0	5	0	5
6	1	2	2	0	8	0
	2	4	0	8	5	0
	3	6	2	0	0	1
7	1	3	5	0	10	0
	2	6	0	7	10	0
	3	8	5	0	0	10
8	1	4	6	0	0	1
	2	10	3	0	10	0
	3	10	4	0	0	1
9	1	2	2	0	6	0
	2	7	1	0	0	8
	3	10	1	0	0	7
10	1	1	4	0	4	0
	2	1	0	2	0	8
	3	9	4	0	0	5
11	1	6	0	2	0	10
	2	9	0	1	0	9
	3	10	0	1	0	7
12	1	0	0	0	0	0

This project uses 2 renewable (R1, R2) and 2 non-renewable resources (N1, N2) and their availabilities are displayed on Table 3.6. For the renewable resources, the availabilities are per period.

Table 3.6. Resource Availabilities

R1	R2	N1	N2
9	4	29	40

The activities network and coefficients using our representation is shown in Figure 3.18. The resources L2-L12 are logical resources.

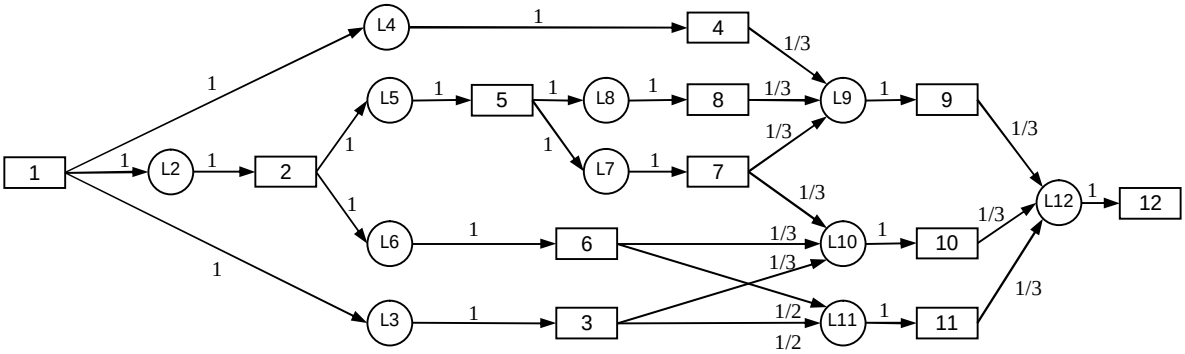
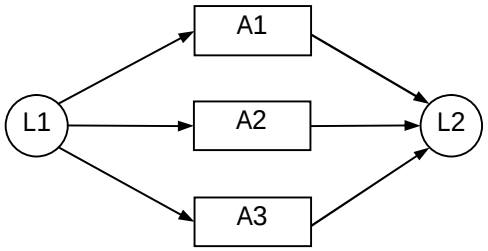


Figure 3.18. Activity network for example problem

Each activity is performed in one out of three modes. When using the MMRTN1 formulation for the MRCPSP case, we use activity alternatives to represent modes. The activity modes are modelled using decision boxes with multiple inputs, exactly one of which is to be executed, as in Figure 3.19:



$$R_{L1}^{initial} = 1,$$
$$\mu_{L1,A1} = \mu_{L1,A2} = \mu_{L1,A3} = -1$$
$$\overline{\mu}_{L2,A1} = \overline{\mu}_{L2,A2} = \overline{\mu}_{L2,A3} = 1$$

Figure 3.19. Modelling activities with multiple modes

Activity A can be performed in 3 modes which are represented as different activities called A1, A2 and A3. Aside from physical resources, the activity requires 1 unit of logical resource L1 ($\mu_{L1,A1} = \mu_{L1,A2} = \mu_{L1,A3} = -1$). To ensure

that only 1 mode is executed, the initial quantity of L1 is limited to 1 ($R_{L1}^{initial}=1$). Upon completion, activity A produces 1 unit of L2 ($\overline{\mu}_{L2,A1}=\overline{\mu}_{L2,A2}=\overline{\mu}_{L2,A3}=1$).

The S_{ri}^{L}, $\overline{S}_{ri}^{L}$, S_{ri}^{U} and $\overline{S}_{ri}^{U}$ parameter values when using the MMRTN2 formulation, are shown on Table 3.7.

Table 3.7. S_{ri}^{L}, $\overline{S}_{ri}^{L}$, S_{ri}^{U} and $\overline{S}_{ri}^{U}$ parameter values

S_{ri}^{L}					S_{ri}^{U}				
Resource / Activity	R1	R2	N1	N2	Resource / Activity	R1	R2	N1	N2
1	-	-	-	-	1	-	-	-	-
2	-	-	-	-	2	6	6	9	8
3	-	-	-	5	3	7	4	-	8
4	6	-	-	-	4	10	-	2	7
5	-	-	-	-	5	2	9	8	7
6	-	-	-	-	6	2	8	8	1
7	-	-	-	-	7	5	7	10	10
8	3	-	-	-	8	6	-	10	1
9	1	-	-	-	9	2	-	6	8
10	-	-	-	-	10	4	2	4	8
11	-	1	-	7	11	-	2	-	10
12	-	-	-	-	12	-	-	-	-

$\overline{S}_{ri}^{L}$			$\overline{S}_{ri}^{U}$		
Resource / Activity	R1	R2	Resource / Activity	R1	R2
1	-	-		-	-
2	-	-		6	6
3	-	-		7	4
4	6	-		10	-
5	-	-		2	9
6	-	-		2	8
7	-	-		5	7
8	3	-		6	-
9	1	-		2	-
10	-	-		4	2
11	-	1		-	2
12	-	-		-	-

The MMRTN1 formulation obtained the optimal makespan (20 time units) in 239.12 CPU s (see Table 3.9) and the optimal project schedule is shown in Figure 3.20. Activities 1 and 12 are not included in the GANTT chart since they are dummy and have zero duration.

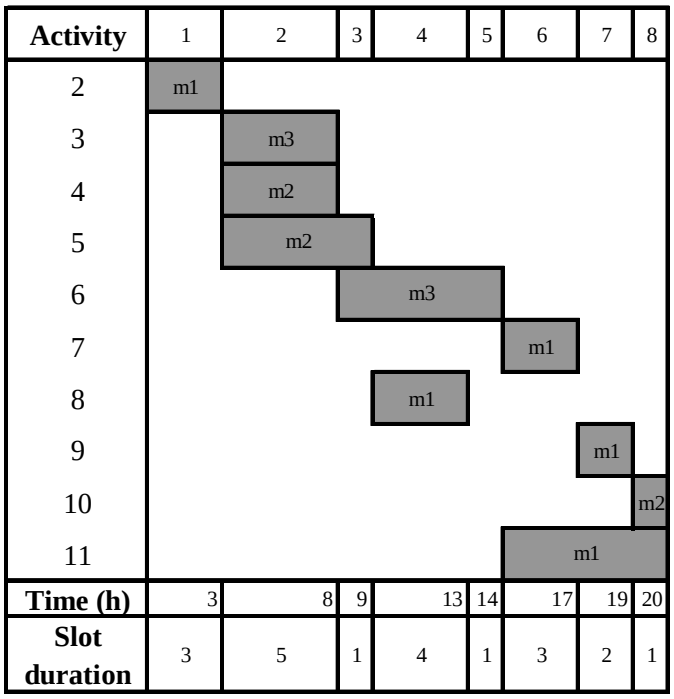


Figure 3.20. GANTT Chart of optimal solution for the example problem

3.5.2 Results for various problem instances

The SMRTN formulation was used to solve a number of single-mode RCPSPs, with 12, 16 and 20 activities, utilizing up to 4 renewable resources. The computational results are presented in Table 3.8.

The formulation managed to find the optimal solution for each instance, and as expected the CPU time required increases with the number of activities.

Table 3.8. Computational Results for various Single-mode RCPSP test instances

Activities	Renewable Resources	Resource Complexity	CPU Time	Number of Equations	Binary Variables	Cont. Variables	Nr. of Nodes	Time Horizon
12	0	-	0.36	761	287	350	257	44
	1	Required by all activities	0.50	774	288	363	157	47
	2	Required by all activities	0.50	787	287	376	141	54
	4	1 Resource by each activity	0.57	813	287	402	183	45
	4	Required by all activities	0.36	813	287	402	162	51
16	0	-	6.60	1286	511	577	2816	73
	3	1 Resource by each activity	9.57	1337	512	628	2509	73
	4	1 Resource by each activity	8.10	1354	512	645	1949	73
	4	Required by all activities	2.71	1354	512	645	852	97
20	0	-	9.12	1926	799	881	2840	92
	1	Required by all activities	17.40	1947	800	902	6142	93
	4	1 Resource by each activity	11.07	2010	800	965	2488	92
	4	Required by all activities	4.46	2010	800	965	1362	116

For the MRCPSP case a set of multi-mode test instances from PSPLIB was used. The multi-mode problem sets were selected from instances j10, j12, j14, c15, c21. For problem sets *j* and *c* each activity may be performed in 1 out of 3 modes and requires 2 renewable and 2 non-renewable resources. The duration of a mode varies between 1 and 10 periods. The problems from sets j10, j12 and j14 include 10, 12 and 14 activities respectively, while the problems from c15 and c21 have 16 activities.

The sets were solved using both the MMRTN1 and MMRTN2 formulations with the improvements and the computational results displayed in Table 3.9. The optimal solutions reported in Table 3.9, are taken from the website of PSPLIB (http://129.187.106.231/psplib/).

Notice that the MMRTN1 formulation performs better on smaller problem instances with 10 and 12 activities. As the number increases, the MMRTN2 model provides better results, due to the smaller number of binary variables.

Table 3.9. Computational Results for Multi-mode RCPSP test instances from PSPLIB

Instance	Model	CPU Time	Equations	Binary Variables	Continuous Variables	Nodes	Solution		
							Integrality Gap (%)	Final	Optimal
j10 2_2	MMRTN1	239.12	1583	600	536	59423	0	20	20
	MMRTN2	1000+	4555	225	3890	118685	5.97	20	
j12 2_8	MMRTN1	146.14	2205	864	738	16925	0	49	49
	MMRTN2	1000+	7023	318	6104	147486	12.24	49	
j14 1_8	MMRTN1	1000+	2967	1176	972	33931	31.63	34	34
	MMRTN2	1000+	10247	427	9027	51586	2.94	34	
c15 4_3	MMRTN1	1000+	3935	1536	1238	14420	44.44	36	34
	MMRTN2	1000+	14345	553	12746	11137	35.13	37	
c21 4_6	MMRTN1	1000+	3899	1536	1238	9058	50.00	38	36
	MMRTN2	1000+	14333	552	12754	7995	36.82	37	

3.6 Conclusions

New MILP models for the RCPSP and the MRCPSP are proposed, and used to solve various project scheduling problems found in the literature.

In the MMRTN2 formulation, the number of integer variables is reduced due to fewer defined binary variables y_{it} and $\overline{y}_{it}$ for fewer tasks. On the other hand, the number of continuous variables and constraints is higher. Overall, the MMRTN1 formulation performs better on smaller test instances, while the MMRTN2 model requires less computational effort for larger problems, due to its reduced number of computationally expensive binary variables. Extensive tests indicate that for large-scale problems (e.g. more than 30 activities) the proposed MILP models cannot lead to a global optimal solution in reasonable computational times.

In summary the main objective of this chapter is to establish a new framework for RCPSPs utilising techniques from the process scheduling area.

As such, the proposed models are suitable to be integrated with challenging process scheduling problems where project scheduling decisions are also important (e.g. pharmaceutical product design and scheduling process). In general the computational results and the similarities between process and project scheduling problems, such as initial and target inventories, required resource types and precedence relations, suggest that exchanging solution techniques between the two research fields is both possible and useful.

3.7 Nomenclature

Indices/ Sets

$i \in I$	activities
$r \in R$	resources
$t \in [1,...,T]$	time slots

Subsets

I_{ALT_X}	x^{th} set of alternative activities, 1 of which is to be executed, $I_{ALT_X} \subset I$
I_r	activities interacting with resource r, $I_r \subset I$
I_{sure}	activities that must be executed, $I_{sure} \subset I$

Parameters

$R_r^{initial}$	initial available quantity of resource r
$R_r^{\min} / R_r^{\max}$	minimum/maximum possible quantities for resource r
$R_{r,Final}^{\min} / R_{r,Final}^{\max}$	minimum/maximum excess of resource r at the end of the project
S_{ri}^{L} / S_{ri}^{U}	lower/upper bounds on the amount of resource r required at the beginning of activity i
$\overline{S}_{ri}^{L} / \overline{S}_{ri}^{U}$	lower/upper bounds on the amount of resource r produced at the end of activity i

$a_{imr}/\bar{a}_{imr}$	the quantities of resource r consumed/produced respectively, when activity i is performed in mode m.
θ_i	duration of activity i
θ_{mi}	duration of activity i when executed in mode m
v_{ri}/μ_{ri}	size dependent/independent coefficient of resource r consumption at the beginning of activity i
$\bar{v}_{ri}/\bar{\mu}_{ri}$	size dependent/independent coefficient of resource r production at the end of activity i
T	number of time slots
$\tau^{\min}/\tau^{\max}$	minimum/maximum slot duration

Binary Variables

y_{it}	1 if activity i starts at t or *0* otherwise
$\bar{y}_{it}$	1 if activity i is active over both t and $t-1$ or *0* otherwise
z_{mi}	1 if activity i is executed in mode m or *0* otherwise

Continuous Variables

H	time horizon
N_{it}	number of instances of activity i for time slot t
R_{rt}	excess quantity of resource r at the start of time slot t
S_{ri}	surplus of resource r consumed at the beginning of activity i (above basic consumption level)
$\bar{S}_{ri}$	surplus of resource r produced at the end of activity i (above basic production level)
$S_{rit}/\bar{S}_{rit}$	linearised surplus terms
τ_t	duration of time slot t
τlin_{it}	linearised duration term

Chapter 4

Four new Mathematical Formulations for the Resource-constrained Project Scheduling Problem

Two new binary integer programming discrete-time models and two novel precedence-based mixed integer continuous-time formulations are developed for the solution of resource-constrained project scheduling problems. The proposed discrete-time models are based on the definition of binary variables that describe the processing state of every activity between two consecutive time points, while the proposed continuous-time models are based on the concept of overlapping of activities, and the definition of a number of newly introduced sets. The four novel mathematical formulations are compared with four representative literature models using a total number of 2760 well-known open-accessed benchmark problem instances (j30 and j60 from the PSPLIB and 1800 problems generated by RanGen2). A detailed computational comparison study demonstrates the salient performance of the proposed mathematical models. The new continuous-time formulations feature the best overall performance. Finally, interesting observations are made through the computational study and future research lines are revealed.

4.1 Introduction

In the previous chapter, a new network representation technique and new mathematical formulations based on the RTN representation, were developed. The computational results displayed that the new models are computationally expensive for problems with more than 30 activities. In this chapter, we propose new discrete- and continuous-time mathematical

programming formulations for larger problems. Open-accessed test instances from ProGen and RanGen2 are considered in a computational comparison study of the performance of the newly proposed mathematical formulations and other models from the literature. Additionally, we make an attempt to interpret the results depending on the type of time-representation used, problem size and difficulty.

The rest of the chapter is organised as follows. In Section 4.2, the classical RCPSCP is formally stated along with some remarks on formulating mathematical models. Section 4.3 describes the preprocessing phase employed in this chapter followed by Section 4.4 which presents four representative mathematical models in the OR literature. In Section 4.5, two new discrete-time BIP and two new precedence-based continuous-time MIP formulations are described in detail. A brief description of the problem instance sets considered is presented in Section 4.6. Then, in Section 4.7 a comprehensive computational comparison study is presented, whereas final conclusions are drawn in Section 4.8.

4.2 Problem Statement

The standard RCPSP considers a project with a finite number of activities $i \in V := \{1,...,n\}$ with durations p_i. Preemption of activities is not allowed (i.e., the processing of activities cannot be interrupted). Precedence relations between some activities are present. These relations are given by defining sets of immediate predecessors E with pairs of activities (i, j), indicating that activity j cannot start before the completion of all its predecessor activities i. Additionally, each activity requires certain amounts r_{ik} of renewable resources $k \in \Re := \{1,...,m\}$, with specific maximum capacities R_k, to be processed. Renewable resources fully retrieve the occupied resource amount after the completion of each activity. In other words, the temporary availability of the renewable resources at every time is constrained. Moreover,

usually for modelling purposes, two dummy activities are added: (i) a start dummy activity *0* to represent the beginning of the project, and (ii) an end dummy activity $n+1$ corresponding to the completion of the project. Dummy activities have zero duration and zero resource requirements. Along with the real activities they comprise set $A = \{0,...,n+1\}$. The typical objective of the RCPSP is to find an optimal (or at least feasible) schedule, satisfying precedence and resource constraints, such that the total duration of the project (i.e., the makespan C_{max}) is minimised. In the standard RCPSP, all activities and renewable resources are available at the beginning of the project. Also, all information data are deterministic.

The standard RCPSP is denoted by $PS|prec|C_{max}$ in accordance with the notation proposed by Brucker et al. (1999), which follows the well-known three-field notation for machine scheduling problems introduced by Graham et al. (1979). More specifically, $\mathrm{PS}|prec|C_{max}$ notation specifies the single-mode project scheduling (PS) problem under precedence constraints between activities ($prec$) while minimizing the makespan of the project (C_{max}).

The classical RCPSP can be formulated as a mathematical programming model in several modelling ways, depending on the definition of decision variables and the construction of necessary constraints.

4.3 Preprocessing Phase

In this chapter, the critical-path method (Kelley, Jr and Walker, 1959) is employed to estimate ES_i and EF_i for each activity i. Additionally, we use the P*arallel Scheduling Scheme* (PSS) of Brooks, as presented by Kolisch (1996a), under two different rules: the minimum latest finishing time rule, and the minimum latest starting time rule, so as to find an upper bound to the time horizon.

The parallel method consists of a number stages that are at most equal to $|A|$, the number of all project activities (including dummy). In each stage, a set of activities (which might be empty) is scheduled. A unique feature of the parallel method is that each stage x is associated with a schedule time t_x, where $t_y \leq t_x$, for $y < x$ holds. On account of this schedule time, the set of scheduled activities is now divided into the following two subsets: Activities which were scheduled and are completed up to the schedule time are in the *complete set* $COMP_x$, whereas activities which were scheduled but which are still active at the schedule time, are in the *active set* ACT_x. Finally, we have the *decision set* DEC_x, which contains all yet unscheduled activities that are available for scheduling w.r.t. precedence *and* resource constraints.

The partial schedule of each stage is made up by the activities in the complete set and the active set. The schedule time of a stage equals the earliest completion time of activities in the active set of the ancestral stage. Each stage is made up of two steps:

(1) The new schedule time is determined and activities with a finish time equal to the (new) schedule time are removed from the active set and put into the complete set. This, in turn, may place a number of activities into the decision set.

(2) One activity from the decision set is selected with a priority rule (again, in case of ties the activity with the smallest label is chosen) and scheduled to start at the current schedule time. Afterwards, this activity is removed from the decision set and put into the active set.

Step (2) is repeated until the decision set is empty, i.e. until all activities are scheduled or are no longer available for scheduling w.r.t. the resource constraints. The parallel method terminates when all activities are in the complete or active set.

Given ACT_x, the active set, and $COMP_x$, the complete set, respectively, πR_k, the left over period capacity of the renewable resource k at the schedule time, and DEC_x, the decision set, are defined as follows:

$$\pi R_k := R_k - \sum_{j \in ACT_x} r_{jk} \ ,$$

$$DEC_x := \left\{ j \mid j \notin \left\{ COMP_x \cup ACT_x \right\}, P_j \subseteq COMP_x, r_{jk} \le \pi R_k \quad \forall k \in \Re \right\}$$

A formal description of the parallel scheduling scheme (PSS), given $v(j)$ a priority value of activity j, $j \in DEC_x$ representing the priority rule, is given below:

Initialisation: $n := 1$, $t_x := 0$, $DEC_x := \{1\}$, $ACT_x := COMP_x := 0$, $\pi R_k := R_k \quad \forall k \in \Re$, GOTO Step (1)

WHILE $\left| ACT_x \cup COMP_x \right| < |A|$ DO **Stage** x;

BEGIN

(1) $t_x := \min \left\{ FT_j \mid j \in ACT_{x-1} \right\}$;

$ACT_x := ACT_{x-1} \setminus \left\{ j \mid j \in ACT_{x-1}, FT_j = t_x \right\}$;

$COMP_x := COMP_{x-1} \cup \left\{ j \mid j \in ACT_{x-1}, FT_j = t_x \right\}$;

COMPUTE $\pi R_k \quad \forall k \in \Re \quad and \quad DEC_x$;

(2) $j^* := \min_{j \in DEC_x} \left\{ j \mid v(j) = \inf_{j \in DEC_x} \{ v(i) \} \right\}$;

$FT_{j^*} := t_x + p_{j^*}$;

$ACT_x := ACT_x \cup \left\{ j^* \right\}$;

COMPUTE $\pi R_k \quad \forall k \in \Re \quad and \quad DEC_x$;

IF $DEC_x \neq \varnothing$ GOTO Step (2) ELSE $x := x + 1$;

END;

Stop

The upper bound calculated by the PSS is then used to calculate LS_i and LF_i, again through the critical-path method. More specifically, the upper

bound on the time horizon is equal to the minimum time horizon found by the two rules applied. Note that the computational time of such a simple preprocessing phase is negligible, and activity time-window lengths can be significantly reduced.

4.4 Review of Existing Mathematical Formulations

In this section, some key and representative discrete- and continuous-time mathematical formulations for the RCPSP found in the literature are briefly presented. These mathematical models have been used for comparison purposes with the new mathematical formulations developed in this chapter.

There is a plethora of mathematical programming formulations for the RCPSP in the OR literature. Typically, the standard RCPSP can be formulated as: a Binary Integer Programming (BIP), involving only binary decision variables, or a linear Mixed Integer Programming (MIP) model, involving both binary and continuous decision variables. Discrete-time models use time-indexed binary decision variables, while continuous-time models usually rely on precedence-based or event-based decision variables. Also, time-indexed continuous-time models can be derived, if a variable time grid is applied. It should be noted that time-indexed BIP discrete-time models divide the scheduling horizon into equal-size time intervals, whereas precedence-based MIP continuous-time formulations are based on sequencing binary variables that indicate the processing priority between pairs of activities.

For the sake of clarity of the models presentation, we use the notation of Brucker et al. (1999). Additionally, we use lowercase Latin letters for decision variables and uppercase Latin letters for sets and subsets. The complete list of the notation used throughout this chapter is given in the Nomenclature section.

4.4.1 Discrete-time model by Pritsker [Pri-DT]

A very early mathematical model for the RCPSP was presented by Pritsker et al. (1969). This Binary Integer Programming (BIP) formulation is based on the definition of binary variables y_{it}, which specify if activity i starts processing at time t (i.e., $y_{it}=1$) or not (i.e., $y_{it}=0$). Activities $i \in V$ have a resource demand at the time point that they start processing, and no resource demand at the time point of their completion. Taking into account the above definition of binary variables y_{it}, the following mathematical model was proposed:

$$\min C_{\max} = \sum_{t=ES_{n+1}}^{LS_{n+1}} t \cdot y_{n+1,t} \tag{1}$$

$$\sum_{t=ES_i}^{LS_i} y_{it} = 1 \qquad \forall i \in (V \cup \{n+1\}) \tag{2}$$

$$y_{00} = 1 \tag{3}$$

$$\sum_{t=ES_i}^{LS_i} t \cdot y_{it} + p_i \le \sum_{t=ES_j}^{LS_j} t \cdot y_{jt} \qquad \forall (i,j) \in E \tag{4}$$

$$\sum_{i \in V} r_{ik} \sum_{t'=\max[ES_i, t-p_i+1]}^{\min[LS_i, t]} y_{it'} \le R_k \qquad \forall t \in T, k \in \Re \tag{5}$$

$$y_{it} \in \{0,1\} \qquad \forall i \in (V \cup \{n+1\}), t = ES_i, ..., LS_i \tag{6}$$

Equation (1) minimises the completion time of the dummy end activity $n+1$, and thus the makespan of the project. Constraints (2) ensure that each activity starts processing exactly once. Note that the starting time for every activity i is between its earliest and latest starting time (i.e., $t = ES_i, ..., LS_i$), and that the dummy start activity 0 begins at $t=0$ according to constraint (3). Constraints (4) and (5) represent the precedence and the renewable resource constraints, respectively. Finally, the decision variables domain is given by constraints (6).

The BIP formulation of Pritsker et al. (1969), consisting of constraints (1) - (6), involves $\sum_{i=1}^{n+1}(LS_i - ES_i)$ binary variables, and $|E| + (h+1)m + n + 1$ constraints. Where h represents the upper bound on the time horizon. Henceforth, this model will be called Pri-DT.

4.4.2 Discrete-time model by Christofides [Chri-DT]

Christofides et al. (1987) proposed a BIP formulation very similar to Pri-DT model, by disaggregating the precedence constrains of Pri-DT, as follows:

$$\sum_{t'=t}^{LS_i} y_{it'} + \sum_{t'=ES_j}^{\min[LS_j, t+p_i-1]} y_{jt'} \leq 1 \qquad \forall (i,j) \in E, t = ES_i, ..., LS_i \qquad (7)$$

That way constraints (7) replace constraints (4). Therefore, the BIP model of Christofides et al. (1987) consists of constraints (1)-(3), and (5)-(7). Henceforth, this model will be referred as Chri-DT. Notice that this formulation has the same number of binary variables as Pri-DT, but involves $\sum_{i=1}^{n+1}(LS_i - ES_i)$ more constraints than Pri-DT.

4.4.3 Continuous-time model by Artigues [Art-CT]

Artigues et al. (2003) proposed a continuous MIP formulation based on sequencing and resource flow variables. Starting time continuous variables s_i are defined for each activity i. In addition, sequencing binary variables x_{ij} are introduced to define the sequence between any pair of activities i and j. Specifically, if activity i is processed before activity j, binary variable $x_{ij} = 1$, otherwise is set to zero. Finally, resource flow continuous variables q_{ijk} are defined to denote the quantity of resource k directly transferred from activity i (at its completion) to activity j (at the beginning of its processing). Notice that in this mathematical formulation $r_{0,k} = r_{n+1,k} = R_k$, instead of setting them

to zero. That way, the start dummy activity *0* acts as a resource source while the end dummy activity *n+1* plays the role of a resource sink. In the current thesis and for comparison purposes we use the modified formulation presented recently in Koné et al. (2011). It should be noted that we removed the first two tightening constraints of the original formulation, because the model performance was inferior when they were included. The MIP model is formally stated by:

$$\min C_{\max} = s_{n+1} \tag{8}$$

$$s_j - s_i \geq \left(ES_j - LS_i\right) + \left(p_i + LS_i - ES_j\right)x_{ij} \qquad \forall (i,j) \in A^2 : i \neq j \tag{9}$$

$$q_{ijk} \leq \min\left\lfloor r_{ik}, r_{jk} \right\rfloor x_{ij} \qquad \forall (i,j) \in \left(V \cup \{0\} \times V \cup \{n+1\}\right), k \in \Re : i \neq j \tag{10}$$

$$\sum_{j \in A,\, j \neq i} q_{ijk} = r_{ik} \qquad \forall i \in A, k \in \Re \tag{11}$$

$$\sum_{i \in A,\, j \neq i} q_{jik} = r_{jk} \qquad \forall j \in A, k \in \Re \tag{12}$$

$$q_{ijk} = R_k \qquad \forall i := \{n+1\},\ j := \{0\}, k \in \Re \tag{13}$$

$$x_{ij} = 1 \qquad \forall (i,j) \in C \tag{14}$$

$$x_{ji} = 0 \qquad \forall (i,j) \notin C \tag{15}$$

$$s_0 = 0 \tag{16}$$

$$ES_i \leq s_i \leq LS_i \qquad \forall i \in \left(V \cup \{n+1\}\right) \tag{17}$$

$$\begin{aligned} q_{ijk} &\geq 0 && \forall (i,j) \in A^2, k \in \Re : i \neq j \\ x_{ij} &\in \{0,1\} && \forall (i,j) \in A^2 : i \neq j \end{aligned} \tag{18}$$

Objective (8) minimises the starting time of the dummy end activity $n + 1$, and therefore the makespan of the project. Constraints (9) are disjunctive constraints that prevent two activities linked through a resource unit flow from being scheduled simultaneously. Constraints (10) link resource flow

variables q_{ijk} and sequencing variables x_{ij}. Note that the maximum resource flow sent from i to j is set to $\min\lfloor r_{ik}, r_{jk} \rfloor$ if activity i precedes activity j, otherwise it is set to zero. Resource flow conservation is ensured by imposing constraints (11) - (13), while precedence relations are satisfied by constraints (14) - (15). The starting time of the start dummy activity is equal to zero, according to constraint (16). Lower and upper bounds on the starting times of the remaining activities are imposed by constraint (17). Finally, constraints (18) provide the domain of the remaining decision variables. Note that set C is the transitive closure of set E.

The MIP formulation of Artigues et al. (2003) consists of constraints (8) - (18), and involves $2|C|$ binary variables, $m(n+2)^2 + n + 1$ continuous variables, and $2|C| + (n+2)^2 + m(n+1)^2 + 2n - m - 1$ constraints. Henceforth, this model will be called Art-CT.

4.4.4 Continuous-time model by Koné [Kone-CT]

Recently, Koné et al. (2011) presented two new continuous time MIP formulations. In this work we compare our models with the On/off event-based formulation with preprocessing (OOE_prec), which according to their computational results presents the best performance. OOE_prec is based on the definition of event points e. The number of event points is equal to the number of non-dummy activities, i.e., $e \in J := \{1,...,n\}$. This MIP model is based on the definition of binary variables v_{ie} that denote if activity i starts processing at event point e, or if it still being processed immediately after event point e (i.e., $v_{ie} = 1$). Event timing continuous variables d_e are also introduced. The model proposed by Koné et al. (2011) is given below:

$$\min C_{\max} \geq d_e + (v_{ie} - v_{i,e-1}) p_i \qquad \forall i \in V, e \in J \qquad (19)$$

$$\sum_{e \in J} v_{ie} \geq 1 \qquad \forall i \in V \tag{20}$$

$$d_1 = 0 \tag{21}$$

$$d_{e+1} \geq d_e \qquad \forall e \in J \setminus \{n\} \tag{22}$$

$$d_{e'} \geq d_e + \left(\left(v_{ie} - v_{i,e-1}\right) - \left(v_{ie'} - v_{i,e'-1}\right) - 1\right) p_i \qquad \forall i \in V, (e,e') \in J^2 : e' > e \tag{23}$$

$$\sum_{e'=1}^{e-1} v_{ie'} \leq (e-1)\left(1 - \left(v_{ie} - v_{i,e-1}\right)\right) \qquad \forall i \in V, e \in J \setminus \{1\} \tag{24}$$

$$\sum_{e' \geq e} v_{ie'} \leq (n-e+1)\left(1 + \left(v_{ie} - v_{i,e-1}\right)\right) \qquad \forall i \in V, e \in J \setminus \{1\} \tag{25}$$

$$v_{ie} + \sum_{e'=1}^{e} v_{je'} \leq 1 + (1 - v_{ie})(e-1) \qquad \forall (i,j) \in E, e \in J \tag{26}$$

$$\sum_{i \in V} r_{ik} v_{ie} \leq R_k \qquad \forall k \in \Re, e \in J \tag{27}$$

$$v_{ie} ES_i \leq d_e \leq LS_i \left(v_{ie} - v_{i,e-1}\right) + LS_{n+1}\left(1 - \left(v_{ie} - v_{i,e-1}\right)\right) \qquad \forall i \in V, e \in J \tag{28}$$

$$ES_{n+1} \leq C_{\max} \leq LS_{n+1} \tag{29}$$

$$v_{ie} = 0 \qquad \forall i \in V, e \in \{1,...,\zeta_i\} \cup \{n - \theta_i + 1,...,n\} \tag{30}$$

$$\begin{aligned} d_e &\geq 0 && \forall e \in J \\ v_{ie} &\in \{0,1\} && i \in V, e \in J \end{aligned} \tag{31}$$

Objective (19) correlates the makespan to the timing of the event points, and thus minimises the makespan of the project. Constraints (20) ensure that each activity $i \in V$, is processed at least once, while constraints (21) and (22) express the sequencing of event points. Constraints (23) give the timing between event points by linking to it event-start variables v_{ie}, and constraints (24) and (25) ensure non-preemption of activities. Precedence relations and renewable resource limitations are expressed by constraints (26) and (27), respectively. Lower and upper bounds on the makespan objective are given by constraints (29). Additionally, constraints (30) set equal to zero all event

points wherein activity i cannot be in process due to its number of predecessors ζ_i and successors θ_i. The decision variables domain is given by constraints (31).

The MIP model of Kone et al. (2011) consists of constraints (19) - (31), and involves: n^2 binary variables, n continuous variables, and $(n-1)(3+|E|+m+n^2/2)+n^2+n$ constraints. Henceforth, this model will be called Kone-CT.

4.5 New Mathematical Formulations

In this section, two new time-indexed discrete-time BIP models and two new precedence-based continuous-time MIP formulations for the RCPSP are developed and described in detail.

4.5.1 Proposed discrete-time model 1 [KKG-DT1]

Here, a new BIP formulation is presented which is based on the definition of two types of binary variables. Specifically, we define binary variables y_{it}, which specify if activity i starts processing at time t (i.e., $y_{it}=1$) or not (i.e., $y_{it}=0$); likewise to Pri-DT and Chri-DT models. Moreover, we introduce variables w_{it}, which specify if activity i is being processed in the time interval between time points t and $t+1$ (i.e., $w_{it}=1$) or not (i.e., $w_{it}=0$). Figure 4.1 shows an illustrative example of how binary variables w_{it} work. Activities $i \in V$ have a resource demand at the time point that they start processing, and no resource demand at the time point of their completion. Considering the above definitions of binary variables y_{it} and w_{it}, the following mathematical formulation is proposed:

$$\min C_{\max} = \sum_{t=ES_{n+1}}^{LS_{n+1}} t \cdot y_{n+1,t} \tag{32}$$

$$\sum_{t=ES_i}^{LS_i} y_{it} = 1 \qquad \forall i \in (V \cup \{n+1\}) \tag{33}$$

$$y_{00} = 1 \tag{34}$$

$$\sum_{t=ES_i}^{LS_i} t \cdot y_{it} + p_i \le \sum_{t=ES_j}^{LS_j} t \cdot y_{jt} \qquad \forall (i,j) \in E \tag{35}$$

$$w_{it} = \sum_{t'=t-p_i+1}^{t} y_{it'} \qquad \forall i \in V, t = ES_i,...,LF_i - 1 \tag{36}$$

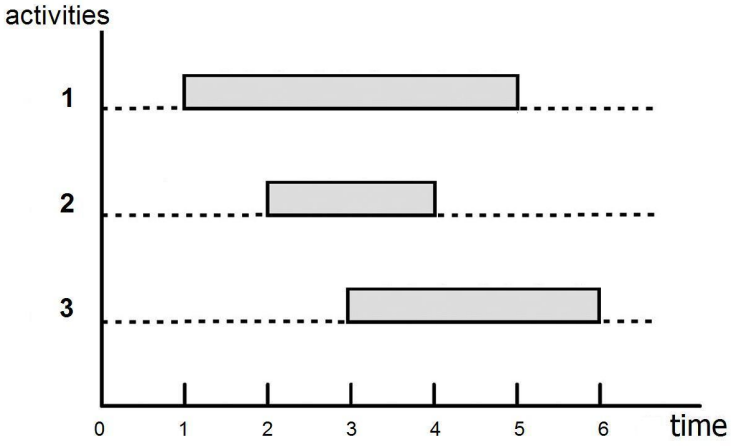
$$\sum_{t=ES_i}^{LF_i-1} w_{it} = p_i \qquad \forall i \in V \tag{37}$$

$$\sum_{\substack{i \in V, r_{ik}>0 \\ t \in \{ES_i, LF_i-1\}}} r_{ik} w_{it} \le R_k \qquad \forall t \in T, k \in \Re \tag{38}$$

$$\begin{aligned} y_{it} &\in \{0,1\} && \forall i \in (V \cup \{n+1\}), t = ES_i,...,LS_i \\ w_{it} &\in \{0,1\} && \forall i \in V, t = ES_i,...,LF_i - 1 \end{aligned} \tag{39}$$

Objective (32) minimises the completion time of the dummy end activity $n+1$, which corresponds to the makespan of the project. Constraints (33) ensure that each activity i starts processing exactly once. Obviously, the starting time for every activity i is between its earliest and latest starting time (i.e., $t = ES_i,...,LS_i$), and the dummy start activity *0* begins at $t = 0$ according to constraint (34). Precedence relations are guaranteed by constraints (35). Additionally, constraints (36) link binary variables y_{it} and w_{it} for every activity i and time points $t = ES_i,...,LF_i - 1$. Constraints (37) tighten the model. The definition of binary variables w_{it} allows us to model the renewable resource constraints in a very simple way, according to constraints (38). The main idea of modelling renewable resource constraints using binary variables w_{it} is illustrated in Figure 4.1 through a simple example considering

3 activities $\{1,2,3\}$ and a single resource k. In this example, $r_{ik}=1$ for all activities. Finally, the domain of the decision variables is given by constraints (39).



w_{1t}	0	1	1	1	1	0	0	$y_{11}=1$
w_{2t}	0	0	1	1	0	0	0	$y_{22}=1$
w_{3t}	0	0	0	1	1	1	0	$y_{33}=1$
$\sum_i r_{ik} w_{it}$	0	1	2	3	2	1	0	

Figure 4.1. Illustrative example: modelling of resource constrains through binary variables w_{it}

This BIP model consists of constraints (32)-(39) and involves $\sum_{i=1}^{n+1}(LS_i - ES_i)+\sum_{i=1}^{n}(LF_i - ES_i)$ binary variables, and $|E|+(h-1)m+n+1+\sum_{i=1}^{n}(LF_i - ES_i)$ constraints. Henceforth, this model will be called KKG-DT1.

4.5.2 Proposed discrete-time model 2 [KKG-DT2]

A slightly different BIP formulation can be obtained by disaggregating the precedence constrains of KKG-DT1, as follows:

$$\sum_{t'=t}^{LS_i} y_{it'} + \sum_{t'=ES_j}^{\min[LS_j,\,t+p_i-1]} y_{jt'} \le 1 \qquad \forall (i,j)\in E, t = ES_i,\ldots,LS_i \qquad (40)$$

By doing so, constraints (40) replace constraints (35), and that way we have a new BIP model that consists of constraints (32)-(34), and (36)-(40). Henceforth, this model will be referred as KKG-DT2. Note that the KKG-DT2 formulation has the same number of binary variables as KKG-DT1, but involves $\sum_{i=1}^{n+1}(LS_i - ES_i)$ more constraints than KKG-DT1.

4.5.3 Proposed continuous-time model 1 [KKG-CT1]

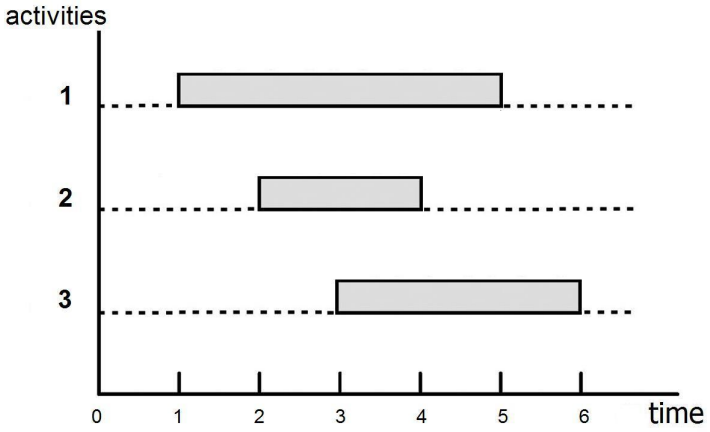
In this part, a new MIP formulation based on the definition of two types of continuous and binary variables is proposed. Specifically, starting and finishing time continuous variables s_i and f_i are defined for each activity $i \in (V \cup \{n+1\})$. Moreover, sequencing binary variables x_{ij} are introduced to define the sequence between any pair of activities i and j. For pairs of activities that *cannot be* executed in parallel (e.g., activities for which the sum of resource requirements exceeds the maximum availability, etc), $x_{ij} = 1$ if activity i is completed before activity j starts processing (i.e., $f_i \leq s_j$), otherwise is zero. However, for pairs of activities that *can be executed in parallel*, x_{ij} is used to define the relative sequencing between the activities' starting times. More specifically, $x_{ij} = 1$ if activity i begins processing before or exactly at the same time as activity j starts ($s_i \leq s_j$), otherwise it is set to 0.

To continue with, in order to model renewable resource constraints using sequence-based continuous-time formulations, it is necessary to identify the set of activities requiring the same resource k and being processed simultaneously. That can be done by extending the concept of overlapping activities (Marchetti and Cedrá, 2009). By definition, an activity j that is overlapping the starting time of activity i must satisfy the following conditions (see Figure 4.2):

(A) It should require some resource k also required by activity i (i.e., $r_{ik} > 0$ and $r_{jk} > 0$).

(B) It should start before or exactly at the time that activity i starts processing (i.e., $s_j \leq s_i$).

(C) It should end after the starting time of activity i (i.e., $f_j > s_i$).



Activity 1 is overlapping activity 2 & 3: $\begin{cases} s_1 < s_2,\ f_1 > s_2 \\ s_1 < s_3,\ f_1 > s_3 \end{cases}$

Activity 2 is overlapping activity 3: $s_2 < s_3,\ f_2 > s_3$

Figure 4.2. Illustrative example for overlapping conditions

In order to model condition (B) the definition of the sequencing binary variables x_{ij} is extended so that it also controls the sequencing of the starting times of parallel tasks i and j. Additionally, condition (C) can be modelled by defining overlapping binary variables z_{ji} which are equal to one whenever activity j is completed after activity i starts processing ($f_j > s_i$). Finally, three sets of activity pairs have been defined:

i) Set $\boldsymbol{B}$ containing activities $(i, j) \in V^2$ sharing at least one renewable resource,

ii) Set $\boldsymbol{G}$ containing $(i, j) \in B$ that cannot be processed simultaneously due to resource capacity limitations (i.e., $r_{ik} + r_{jk} > R_k$) and

iii) Set $\boldsymbol{D}$ containing activities $(i, j) \notin C$ for which according to the preprocessing phase, $LF_i \leq ES_j$ (i.e., activity j should be processed after the completion of activity i).

Before moving on to describe the various constraints we use the previously defined sets to create more composite ones. First, we can define Set $\boldsymbol{K} = (C \cup D)$ containing activities for which precedence is already known and thus, no x_{ij} variables need to be defined. Set $\boldsymbol{S} = G \setminus K$ contains activities that cannot overlap due to resource capacity limitations, excluding those with known precedence relations. For this set, only variables x_{ij} need to be defined. Finally, set $\boldsymbol{P} = B \setminus (G \cup K)$ contains activities that can overlap. For such activity pairs, both x_{ij} and z_{ji} variables need to be defined. The preprocessing time to determine sets B, C, D, G, K and P is negligible. Using the above definition of decision variables, sets and overlapping conditions, the following mathematical formulation is derived:

$$\min C_{\max} = f_{n+1} \tag{41}$$

$$f_i = s_i + p_i \qquad \forall i \in (V \cup \{n+1\}) \tag{42}$$

$$f_i \leq s_j \qquad \forall (i, j) \in (V \times (V \cup \{n+1\})) : (i, j) \in K \tag{43}$$

$$f_i \leq s_j + (LF_i - ES_j) x_{ji} \qquad \forall (i, j) \in S : i \neq j \tag{44}$$

$$x_{ij} + x_{ji} = 1 \qquad \forall (i, j) \in V^2 : (i, j) \notin K, i > j \tag{45}$$

$$s_j \leq s_i + (LS_j - ES_i) x_{ij} \qquad \forall (i, j) \in P : i > j \tag{46}$$

$$s_i + \lambda \leq s_j + (LS_i - ES_j + \lambda) x_{ji} \qquad \forall (i, j) \in P : i > j \tag{47}$$

$$f_j - s_i \leq (LF_j - ES_i) z_{ji} \qquad \forall (i, j) \in P : i \neq j \tag{48}$$

$$r_{ik} + \sum_{\substack{j \neq i : r_{jk} > 0 \\ (i,j) \in P}} r_{jk} (z_{ji} - x_{ij}) \leq R_k \qquad \forall i \in V, k \in \Re : r_{ik} > 0 \tag{49}$$

$$x_{ij} \leq z_{ji} \qquad \forall (i, j) \in P : i \neq j \tag{50}$$

$$x_{ji} = 1 \qquad \forall (i,j) \in P : i \neq j, LS_j \leq ES_i \qquad (51)$$

$$\begin{aligned} & s_i \geq 0,\ f_i \geq 0 && \forall i \in (V \cup \{n+1\}) \\ & x_{ji} \in \{0,1\} && \forall (i,j) \in (B \cup G) \setminus K : i \neq j \\ & z_{ji} \in \{0,1\} && \forall (i,j) \in P : i \neq j \end{aligned} \qquad (52)$$

Equation (41) minimises the completion time of the dummy end activity $n+1$, and therefore the duration of the project. Constraints (42) correlate the finishing and starting time for activities $i \in (V \cup \{n+1\})$. Constraints (43) set all the priori known precedence relations between activities $(i,j) \in K$. Constraints (44)-(47) define the sequencing constraints between any pair of activities $(i,j) \notin K$. The value of big M could strongly affect the performance of any model. For this reason, and in order not to use any arbitrary values for the big M parameters, we use proper differences between ES, LS and LF; similarly to the Art-CT model. It should be noted that constraints (46) and (47) relax (extend) the definition of sequencing binary variables x_{ij} to also control the starting times of the potential parallel tasks i and j. Also, notice that a small positive parameter λ is included in constraint (47) to deal with the case that activities i and j start processing at the same time. This way, when $s_i = s_j$ the activity with the higher index is assumed to start last (i.e., if $s_i = s_j$ and $i > j$, then $x_{ji} = 1$). In the case studies we used a value of *0.1* for parameter λ. Overlapping condition (C) is modelled by constraints (48), while renewable resource constraints are given by constraints (49). By using $(z_{ji} - x_{ij})$ we are able to detect activity overlapping and properly model the resource constraints.

Constraints (46) – (49) state that if activity i starts processing before activity j (i.e., $x_{ij} = 1$), then activity j has to finish after activity i starts (i.e., $z_{ji} = 1$). The only case that the difference $(z_{ji} - x_{ij})$ takes the value of *1* is when activity j overlaps i. That means, according to the definition of overlapping given above, that j finishes after i starts (i.e., $z_{ji} = 1$) and activity j starts

processing before i (i.e., $x_{ij} = 0$). Constraints (50) are tightening constraints that correlate directly sequencing and overlapping binary variables. It can be easily proven that for activities which could overlap $(i, j) \in P$, if $x_{ij} = 1$ that means that $s_i \leq s_j < f_j$, and consequently $f_j - s_i > 0$ which according to constraints (48) gives $z_{ji} = 1$.

Table 4.1 displays all possible combinations of variables z_{ji} and x_{ij} for activities $(i, j) \in P$ that could be executed in parallel. In the first two cases wherein $z_{ji} = 0$, no overlapping occurs since activity i starts after the completion of activity j (i.e., $s_j < f_j \leq s_i$). Especially, in the second case, notice that if $z_{ji} = 0$ and $x_{ij} = 1$ that would mean that $f_j \leq s_i < s_j$, which obviously is impossible. For this reason, if $z_{ji} = 0$ then always $x_{ij} = 0$. In the last two cases wherein $z_{ji} = 1$, overlapping occurs if and only if activity j begins processing before activity i starts (i.e., $x_{ij} = 0$), because in that case holds $s_j < s_i < f_j$. Further, the sequencing binary variables for activities $(i, j) \in P$ that $LS_j \leq ES_i$ can be fixed due to the preprocessing phase, according to constraints (51). Finally, the domain of the decision variables is given by constraints (52).

Table 4.1. Modelling of binary variables z_{ji} and x_{ij} for activities $(i, j) \in P$

Constraints (48)		Constraints (46) - (47)		Constraints (49)
$f_j - s_i$	z_{ji}	$s_i - s_j$	x_{ij}	$z_{ji} - x_{ij}$
≤ 0	0	> 0	0	0
≤ 0	0	< 0	1	impossible
> 0	1	< 0	1	0
> 0	1	> 0	0	1

The proposed continuous-time MIP model consists of constraints (41) - (52), and henceforth, this model will be referred as KKG-CT1.

4.5.4 Proposed continuous-time model 2 [KKG-CT2]

Here, another continuous-time MIP formulation is proposed, which uses the same types of binary and continuous variables as the KKG-CT1 model, while modelling the sequencing and resource constraints differently, as follows:

$$f_j \le s_i + \left(LF_j - ES_i\right)\left(1 - x_{ji}\right) \qquad \forall (i,j) \in S : i > j \tag{53}$$

$$f_i \le s_j + \left(LF_i - ES_j\right)x_{ji} \qquad \forall (i,j) \in S : i > j \tag{54}$$

$$s_j \le s_i + \left(LS_j - ES_i\right)\left(1 - x_{ji}\right) \qquad \forall (i,j) \in P : i > j \tag{55}$$

$$s_i + \lambda \le s_j + \left(LS_i - ES_j + \lambda\right)x_{ji} \qquad \forall (i,j) \in P : i > j \tag{56}$$

$$r_{ik} + \sum_{\substack{j<i : r_{jk}>0 \\ (i,j)\in P}} r_{jk}\left(z_{ji} + x_{ji} - 1\right) + \sum_{\substack{j>i : r_{jk}>0 \\ (i,j)\in P}} r_{jk}\left(z_{ji} - x_{ij}\right) \le R_k \qquad \forall i \in V, k \in \Re : r_{ik} > 0 \tag{57}$$

$$x_{ji} \le z_{ij} \qquad \forall (i,j) \in P : i > j \tag{58}$$

$$1 - x_{ji} \le z_{ji} \qquad \forall (i,j) \in P : i > j \tag{59}$$

$$x_{ji} = 1 \qquad \forall (i,j) \in P : i > j, LS_j \le ES_i \tag{60}$$

$$\begin{aligned} &s_i \ge 0,\ f_i \ge 0 && \forall i \in \left(V \cup \{n+1\}\right) \\ &x_{ji} \in \{0,1\} && \forall (i,j) \in \left(B \cup G\right) \setminus K : i > j \\ &z_{ji} \in \{0,1\} && \forall (i,j) \in P : i \ne j \end{aligned} \tag{61}$$

Constraints (53)-(56) define the sequencing for any pair of activities $(i,j) \notin K$, when $i > j$. The relative sequencing, regarding the starting times, of parallel activities i, j is given by constraints (55) and (56), and constraints (57) express resource constraints. Moreover, constraints (58) and (59) are tightening constraints which directly correlate sequencing and overlapping binary variables; similarly to constraints (50) of the KKG-CT1 model.

Additionally, sequencing binary variables x_{ji} can be fixed for activities $(i, j) \in P$ that $i > j$ and $LS_j \leq ES_i$; i.e., it is known a priori that activity j starts before i. Finally, the domain of the decision variables is given by constraints (61).

The new proposed continuous-time MIP model consists of constraints (41) - (43), (48) and (53) - (61). Henceforth, this model will be referred as KKG-CT2. Note that the KKG-CT2 formulation has the same number of constraints, continuous variables, overlapping binary variables and the half of sequencing variables than KKG-CT1.

4.6 Description of Problem Instance Sets

In this section, we present the project generators, the parameters they toggle to create test instances, and the problem sets available for benchmarking project scheduling solution techniques. Specifically, the computational performance of all formulations has been tested extensively using a large number of test instances, generated by the parameter-driven generators ProGen and RanGen2.

In this work, problem sets with 30 and 60 activities, utilizing 4 renewable resources are used from the PSPLIB. Each set comprises of 480 test problem instances, with the following parameters values:

NC = {1.50, 1.80, 2.10},
RF = {0.25, 0.50, 0.75, 1.00}
RS = {0.20, 0.50, 0.70, 1.00}

Henceforth, the 480 problem instance test sets with 30 and 60 activities will be referred to as *j30* and *j60*, respectively.

We also use the 1800 problem instances generated by RanGen2, which are available online at http://www.projectmanagement.ugent.be/rangen.html.

This set of problem instances contains single-mode RCPSP with 30 activities and 4 renewable resources and will be referred to as *RanGen2*, henceforth. For the 1800 problem set the Resource-Constrainedness (RC) (Patterson, 1976), which is defined per resource type as the average quantity demanded by all activities divided by its availability, has a fixed value of 0.4. Another parameter, the Resource Use (RU) (Demeulemeester and Herroelen, 2002), which is equal to the number of resource types required by each activity, is set to 3. The 1800 test instances are divided in 5 different sets of problems (Set 1 to Set 5). Set 1 was generated using nine different values for the I_2 indicator, thus generating 100 instances per setting resulting in a total of 900 instances. The other sets were generated by varying the I_2 indicator (by three values) and one other indicator (I_3, I_4, I_5 or I_6). It should be noted that 10 instances per setting were generated, resulting in 240 instances for Sets 3, 4 and 5. Set 2 contains only 180 instances because networks with high I_2 values and I_3 values lower than 0.75 could not be generated or simply do not exist.

4.7 Computational Comparison Study

This section summarises the computational comparison between some existing representative formulations and the four new mathematical formulations presented in Sections 4.4 and 4.5, respectively. Here, we present the computational results for all formulations, and further analyse and discuss the computational performance of the models considered. Notice that our experimental computational study involves a total number of 2760 RCPSP problems, considering the 3 benchmark RCPSP problem sets j30, j60 and RanGen2 described in the previous section. All mathematical formulations have been solved on an *Intel Core i5 CPU M430 2.27GHz with 4GB RAM* using *CPLEX 11.1.1* via a *GAMS 22.8.1* (Rosenthal, 2012) *WIN 6007.6015 VIS* interface under standard configurations. A maximum resource time limit of *600 CPU s* has been set for all problem instances.

A description of the notation used in the computational results follows.

Feasible (%) is the percentage of instances that gave an integer solution (i.e., optimal, suboptimal, or not-proven optimal) within the predefined time limit.

Good (%) is the percentage of instances that gave a good integer solution (i.e., optimal, or suboptimal with a gap lower than 3% from the optimal solution) within the predefined time limit.

Optimal (%) represents the percentage of (proven) optimal solutions found within the time limit.

Gap (%) is the average gap of the integer non-proven optimal solutions from the real optimal solution, or the best known solution if optimal solution is not available (some problems in j60 and RanGen2).

ΔHEUR (%) stands for the average deviation of the optimal solution from the upper bound calculated by the preprocessing stage, taking into account only instances that were solved to optimality.

Similarly, *ΔCPM (%)* is the average deviation of the optimal solution found from the lower bound calculated through the critical-path method.

Finally, *Optimal CPU (s)* is the average CPU time required for instances solved to optimality.

4.7.1 Overall Computational Results

Table 4.2 presents the overall mathematical models performance ranking (based on Good (%)) for the total number of 2760 problem instances considered. It is observed that the two continuous-time MIP formulations proposed in this work, KKG-CT1 and KKG-CT2, outperform the remaining models. Moreover, it should be noted that the best-performing discrete-time model is KKG-DT2 which outperforms the models of Chri-DT and Pri-DT. Also, notice that KKG-DT2 found the highest number of optimal solutions within the predefined time limit. Chri-DT is slightly better than KKG-DT1. As Table 4.2 clearly shows, the Art-CT and especially the Kone-CT formulations perform rather poor.

Table 4.2. Overall mathematical models ranking for the 2760 problems considered

Ranking	Model	Good (%)	Optimal	Feasible (%)
1	KKG-CT1	74.82	69.78	87.32
2	KKG-CT2	74.10	69.06	86.60
3	KKG-DT2	71.09	70.25	86.16
4	Chri-DT	70.00	68.91	84.38
5	KKG-DT1	69.86	68.23	82.72
6	Pri-DT	65.76	63.95	74.89
7	Art-CT	50.00	45.40	56.89
8	Kone-CT	39.38	26.12	61.63

In the literature, discrete-time models have been considered to be better and more appropriate than their continuous-time counterparts for dealing with the RCPSP. This fact is also reflected in the plethora of discrete-time models developed so far, in contrast to the few approaches using a continuous-time representation. Our experimental study demonstrates that continuous-time formulations (such as the ones proposed in this work) can perform better than state-of-the-art discrete-time models. We tend to believe that the definition of the decision variables is the most significant stage in the overall modelling process, and that further improvement in the modelling phase seems a particularly challenging and promising research direction.

Table 4.3 presents the computational results for all mathematical formulations per problem set. Note that RanGen2 problem set has been divided in RanGen2-A (containing Set 1 of the RanGen2 library) and RanGen2-B (containing Set 2 to 5 of RanGen2 library) subsets, because there are no available optimal (neither best) solutions for Set 1 (i.e., RanGen2-A) from Vanhoucke et al. (2008). For this reason, in RanGen2-A, Good (%) is set equal to Optimal (%). The best performance values are marked in bold.

The proposed continuous-time models, KKG-CT2 and especially KKG-CT1 feature the best performance in the j30, RanGen2-A, and RanGen2-B problem sets that involve 30 activities. However, it is worth noticing that the discrete-time model KKG-DT2 reported the highest number of optimal solutions in the j30 problem set.

Table 4.3. Computational results per problem set

Problem Set	Model	Feasible (%)	Good (%)	Optimal (%)	Gap (%)	ΔHEUR (%)	ΔCPM (%)	Optimal CPU s
j30	Pri-DT	87.71	83.54	80.42	5.19	2.00	4.03	8.19
	Chri-DT	94.17	87.29	84.79	6.52	2.32	5.62	13.58
	KKG-DT1	91.25	88.33	85.21	5.49	2.34	5.52	18.49
	KKG-DT2	94.58	89.38	88.13	8.24	2.49	6.63	21.65
	Art-CT	81.04	75.21	67.50	3.04	1.53	3.47	10.57
	Kone-CT	62.92	48.75	32.08	3.73	0.43	0.09	12.94
	KKG-CT1	98.13	95.21	85.42	1.97	2.61	9.82	16.78
	KKG-CT2	96.88	93.54	85.42	2.39	2.61	9.57	17.79
j60	Pri-DT	76.25	73.33	72.50	6.68	1.93	0.82	5.10
	Chri-DT	77.71	75.42	75.42	6.77	2.18	1.13	8.62
	KKG-DT1	77.29	75.63	75.21	5.60	2.17	1.13	10.74
	KKG-DT2	77.50	76.04	75.21	4.46	2.18	1.15	8.91
	Art-CT	63.13	59.58	55.42	3.96	1.32	0.58	28.50
	Kone-CT	2.71	2.08	2.08	4.80	0.00	0.00	106.62
	KKG-CT1	71.67	67.50	64.58	4.81	1.87	1.34	19.56
	KKG-CT2	70.83	67.29	62.71	4.25	1.77	1.27	20.41
RanGen2-A	Pri-DT	71.33	56.56	56.56	_	3.48	4.04	17.94
	Chri-DT	84.11	62.00	62.00	-	3.86	5.80	22.57
	KKG-DT1	82.89	61.00	61.00	-	3.78	5.51	29.74
	KKG-DT2	87.44	63.33	63.33	-	3.99	6.21	24.00
	Art-CT	49.56	36.89	36.89	-	1.96	3.46	26.47
	Kone-CT	75.44	32.00	32.00	-	2.09	1.10	38.98
	KKG-CT1	88.00	66.67	66.67	-	4.09	9.32	17.78
	KKG-CT2	87.56	65.56	65.56	-	4.00	9.02	13.15
RanGen2-B	Pri-DT	70.89	61.44	58.00	9.60	3.89	5.92	23.80
	Chri-DT	83.00	65.89	63.89	9.90	4.11	7.47	16.97
	KKG-DT1	80.89	65.78	62.67	10.96	4.10	7.20	33.75
	KKG-DT2	85.00	66.44	65.00	11.61	4.15	7.93	21.44
	Art-CT	48.00	44.56	36.78	2.63	1.56	6.98	25.12
	Kone-CT	78.56	61.67	29.89	3.39	2.42	2.12	42.40
	KKG-CT1	89.22	76.00	67.33	5.42	4.05	12.09	20.52
	KKG-CT2	88.56	76.11	67.22	5.27	4.07	11.63	17.56

In the problem set j60, which involves a total number of 60 activities, discrete-time models perform better than continuous-time formulations. More specifically, KKG-DT2 and Chri-DT feature the best performance among the mathematical models considered. These results indicate that the performance of continuous-time formulations probably tends to reduce as the total number of activities increases. The worst-performing model, and probably not surprisingly, is that of Kone-CT, mainly due to its very big size. This model was able to solve solely problem instances whose optimal solutions are very

near to the solution found by the CPM method or to the solution found by the preprocessing stage (i.e., low ΔHEUR and ΔCPM values).

Another interesting observation is that continuous-time models report lower Gap (%), in comparison with the discrete-time models. In other words, a suboptimal integer solution found by a continuous-time formulation is more probable to be a good quality solution (i.e., close to the optimal). This characteristic of the continuous-time models could be of great importance in real-life RCPSP wherein near-optimal solutions, within reasonable computational time, are often acceptable. In order to highlight this point more, consider problem set j30, and observe that KKG-DT2 features a 88.13 Optimal (%), whereas KKG-CT1 reports a lower 85.42 Optimal (%) value. Nevertheless, KKG-CT1 results in a 95.21 Good (%) value, due to a very low 1.97 Gap (%), while KKG-DT2 reaches a 89.38 Good (%) value. Therefore, KKG-CT1 clearly overwhelms KKG-DT2 in the j30 problem instance. Similar, observations can be made for the remaining problem sets.

4.7.2 Computational Results: Detailed Analysis

In the detailed analysis of the computational results some further notation is introduced. More specifically, *Subopt. sol.* corresponds to the number of instances producing a feasible but non-proven optimal solution. *Actually opt.* is the number of suboptimal solutions that are actually optimal, but the solver did not manage to prove it due to the time limit. *Actually opt. (%)* is the percentage of suboptimal solutions that are actually optimal, *Subopt. gap <3%* is the number of suboptimal solutions with a deviation from the optimal solution that is less than 3% (excluding the Actually opt.), *Good subopt. (%)* is the percentage of suboptimal solutions that are actually optimal or have a gap <3%. *Dif (%)* is the percentage of improvement difference between Optimal (%) and Good (%).

Problem Set j30

The detailed analysis of the suboptimal solutions from each mathematical model for problem set j30 is presented in Table 4.4, which demonstrates the higher quality of the suboptimal solutions derived with continuous-time models in contrast with those found by discrete-time formulations. For instance, observe that 31.15% of the suboptimal solutions reported by KKG-CT1 are actually (i.e., non-proven) optimal, and furthermore 77.05% of the suboptimal solutions are either non-proven optimal or feature a gap <3% from the optimal solution. The corresponding percentages are relatively low for the discrete-time models, in accordance with Table 4.4.

Table 4.4. Detailed analysis of suboptimal solutions found for problems sets j30 and j60

Problem Set	Model	Subopt. sol.	Actually opt.	Subopt. gap<3%	Actually opt. (%)	Good subopt. (%)	Optimal (%)	Good (%)	Dif (%)
j30	Pri-DT	35	5	10	14.29	42.86	80.42	83.54	3.88
	Chri-DT	45	6	6	13.33	26.67	84.79	87.29	2.95
	KKG-DT1	29	5	10	17.24	51.72	85.21	88.33	3.67
	KKG-DT2	31	4	2	12.90	19.35	88.13	89.38	1.41
	Art-CT	65	20	17	30.77	56.92	67.50	75.21	11.42
	Kone-CT	148	45	35	30.41	54.05	32.08	48.75	51.96
	KKG-CT1	61	19	28	31.15	77.05	85.42	95.21	11.46
	KKG-CT2	55	16	23	29.09	70.91	85.42	93.54	9.51
j60	Pri-DT	18	2	2	11.11	22.22	72.50	73.33	1.15
	Chri-DT	11	0	0	0.00	0.00	75.42	75.42	0.00
	KKG-DT1	10	0	2	0.00	20.00	75.21	75.63	0.55
	KKG-DT2	11	0	4	0.00	36.36	75.21	76.04	1.11
	Art-CT	37	7	13	18.92	54.05	55.42	59.58	7.52
	Kone-CT	3	0	0	0.00	0.00	2.08	2.08	0.00
	KKG-CT1	34	3	11	8.82	41.18	64.58	67.50	4.52
	KKG-CT2	39	8	14	20.51	56.41	62.71	67.29	7.31

Table 4.5 shows a detailed analysis of good quality solutions for different RS and RF parameter settings for problem set j30. The maximum value for each cell is 30 instances. This also applies to Tables 4.6 and 4.7. According to this table, problem instances with low RS value (i.e., 0.20) and high RF values (i.e., 0.75, and 1.00) are the most difficult RCPSP to solve by the mathematical models considered. Note that only the proposed KKG-CT1 and KKG-CT2 models have managed to solve a significant percentage of these problem instances. The most difficult problem instances for KKG-CT1 and

KKG-CT2 are those with RS={0.20, 0.50} and RF=1.00. It should be noted that discrete-time models solved almost all problems with RS=0.50 and RF=1.00, but manage to solve just a very small number of problem instances with RS=0.20 and RF={0.75, 1.00}. The majority of the remaining problem instances in problem set j30 have been solved by the mathematical models considered except for the Kone-CT model, which performed poorly. The proposed KKG-CT1 formulation found the highest quality solutions, resulting in a 95.21% good solutions percentage.

Table 4.5. Detailed analysis of good quality solutions per RS and RF parameter setting for j30

Parameter		Mathematical Formulation							
RS	RF	Pri-DT	Chri-DT	KKG-DT1	KKG-DT2	Art-CT	Kone-CT	KKG-CT1	KKG-CT2
0.20	0.25	30	30	30	30	30	16	30	30
0.20	0.50	14	23	27	30	8	8	30	30
0.20	0.75	0	4	6	7	0	1	29	27
0.20	1.00	0	6	3	2	0	2	23	18
0.50	0.25	30	30	30	30	30	19	30	30
0.50	0.50	30	30	30	30	29	11	30	30
0.50	0.75	30	30	30	30	23	6	28	28
0.50	1.00	27	26	28	30	4	2	18	19
0.70	0.25	30	30	30	30	30	26	30	30
0.70	0.50	30	30	30	30	30	14	30	30
0.70	0.75	30	30	30	30	30	12	30	30
0.70	1.00	30	30	30	30	27	10	29	27
1.00	0.25	30	30	30	30	30	29	30	30
1.00	0.50	30	30	30	30	30	28	30	30
1.00	0.75	30	30	30	30	30	24	30	30
1.00	1.00	30	30	30	30	30	26	30	30
Good solutions		401	419	424	429	361	234	457	449
Good (%)		83.54	87.29	88.33	89.38	75.21	48.75	95.21	93.54

Problem Set j60

Table 4.4 also shows a detailed analysis of the suboptimal solutions found from each mathematical model for problem set j60. Once again, we observe the higher quality of the suboptimal solutions found from continuous-time formulations in comparison with those reported by discrete-time models. For example, 20.51% of the suboptimal solutions reported by KKG-CT2 are non-proven optimal, and 56.41% of the suboptimal solutions are either non-proven optimal or have a gap from the optimal solution lower than 3%.

Again, it should be noted that the corresponding percentages are relatively low for the discrete-time formulations. The detailed analysis of good quality solutions for different RS and RF parameter settings for problem set j60 is summarised in Table 4.6.

Table 4.6. Detailed analysis of good quality solutions per RS and RF parameter setting for j60

Parameter		Mathematical Formulation							
RS	RF	Pri-DT	Chri-DT	KKG-DT1	KKG-DT2	Art-CT	Kone-CT	KKG-CT1	KKG-CT2
0.20	0.25	23	29	29	30	17	0	30	30
0.20	0.50	0	0	0	0	0	0	1	0
0.20	0.75	0	0	0	0	0	0	0	0
0.20	1.00	0	0	0	0	0	0	0	0
0.50	0.25	30	30	30	30	30	0	30	30
0.50	0.50	27	28	28	28	8	0	19	19
0.50	0.75	20	22	23	24	2	0	4	6
0.50	1.00	12	13	13	13	1	0	2	2
0.70	0.25	30	30	30	30	30	0	30	30
0.70	0.50	30	30	30	30	30	0	30	30
0.70	0.75	30	30	30	30	27	0	29	29
0.70	1.00	30	30	30	30	21	0	29	27
1.00	0.25	30	30	30	30	30	6	30	30
1.00	0.50	30	30	30	30	30	2	30	30
1.00	0.75	30	30	30	30	30	1	30	30
1.00	1.00	30	30	30	30	30	1	30	30
Good solutions		352	362	363	365	286	10	324	323
Good (%)		73.33	75.42	75.63	76.04	59.58	2.08	67.50	67.29

It is demonstrated clearly that the hardest RCPSPs to solve by the mathematical models considered are those with low RS value (i.e., 0.20) and medium-to-high RF values (i.e., 0.50, 0.75, and 1.00); notice the inability of all models to solve these problem instances. Additionally, problem instances with RS=0.50 and RF={0.75, 1.00} are hard to solve for continuous-time models, however discrete-time models perform considerably better. The majority of the remaining problem instances, in problem set j60, has been solved by the mathematical models considered except for the Kone-CT model, which again had an extremely poor performance featuring just a 2.08% good solutions percentage. The proposed KKG-DT2 formulation found the largest number of good quality solutions, resulting in a 76.04% of good solutions.

Problem Set RanGen2

Table 4.7 presents the computational analysis for Sets 2 to 5 of RanGen2. Note that the computational analysis for Set 1 can be found in Table 4.4 (see RanGen2-A problem set). With respect to Good (%) values, the KKG-CT1 performs best in Set 1, 3 and 5, and KKG-CT2 in Set 2, whereas in Set 4 they both report a percentage of 70 % of good solutions. It is worth noticing that the continuous-time models proposed (KKG-CT1 and KKG-CT2) perform better than any discrete-time model considered in all sets of the RanGen2 problem set.

Once again, lower Gap (%) values are observed for the continuous-time formulations in comparison with those of the discrete-time models. For instance, in Set 4, KKG-DT2 and KKG-CT2 report a 65.42% and 62.50% of optimal solutions, respectively. Nevertheless, KKG-CT2 features a higher Good (%) percentage (70.00%) than that of KKG-DT2 (67.08%), due to the fact that KKG-CT2 reports many good suboptimal solutions which are either non-proven optimal or have a gap from the optimal solution lower than 3%.

According to the computational results in Table 4.7, Sets 1, 2, and 4 seem to include the hardest RCPSP problem instances for the mathematical models considered. Also, note that KKG-CT1 and KKG-CT2 solved very successfully almost all problem instances of Set 3.

It is worth mentioning that the KKG-DT2 model outperforms the Chri-DT model in all sets apart from Set 5, and the Pri-DT model is the worst discrete-time model in all sets. On average, Kone-CT performs better than Art-CT in contrast with the previous problem sets j30 and j60.

Table 4.7. Computational analysis for problem Set 2 to 5 for RanGen2

Problem Set	Model	Feasible (%)	Good (%)	Optimal (%)	Gap (%)	ΔHEUR (%)	ΔCPM (%)	Optimal CPU s
Set 2	Pri-DT	64.44	52.22	49.44	13.02	6.46	5.77	34.14
	Chri-DT	77.78	54.44	54.44	11.79	6.39	7.48	19.50
	KKG-DT1	78.33	56.11	52.78	12.06	6.48	6.97	37.96
	KKG-DT2	83.89	56.67	55.56	12.81	6.40	7.93	30.42
	Art-CT	26.11	23.33	16.11	2.69	2.33	8.91	31.02
	Kone-CT	67.78	46.67	18.89	4.51	5.18	2.11	105.26
	KKG-CT1	80.56	61.67	53.33	6.44	5.92	10.08	17.23
	KKG-CT2	81.67	64.44	55.00	7.06	6.24	9.79	27.86
Set 3	Pri-DT	67.92	64.17	58.75	3.88	0.66	10.71	13.81
	Chri-DT	81.25	71.67	67.92	5.50	0.91	13.17	18.76
	KKG-DT1	74.58	69.58	65.83	5.71	0.87	12.89	36.39
	KKG-DT2	81.25	72.50	70.00	7.50	1.02	14.06	28.61
	Art-CT	70.00	67.08	62.92	2.30	0.73	11.76	22.86
	Kone-CT	91.25	79.17	33.75	1.73	0.25	4.69	13.06
	KKG-CT1	99.17	94.58	81.25	2.01	1.69	22.16	22.64
	KKG-CT2	98.33	94.17	81.25	2.44	1.72	21.72	20.15
Set 4	Pri-DT	75.00	62.50	60.00	9.68	4.88	3.58	27.06
	Chri-DT	86.25	66.67	64.17	9.77	5.29	4.30	16.13
	KKG-DT1	86.25	67.50	63.33	11.54	5.26	4.13	30.47
	KKG-DT2	86.67	67.08	65.42	11.54	5.37	4.64	16.65
	Art-CT	42.92	39.17	30.00	2.94	2.03	1.81	23.36
	Kone-CT	76.25	57.92	32.08	4.02	3.12	1.05	52.59
	KKG-CT1	83.75	70.00	63.33	6.14	4.94	5.65	21.38
	KKG-CT2	83.75	70.00	62.50	5.30	4.79	4.68	12.43
Set 5	Pri-DT	74.58	64.58	61.67	10.57	4.47	3.74	23.91
	Chri-DT	85.42	67.92	66.67	11.41	4.84	4.71	14.40
	KKG-DT1	83.75	67.50	66.25	11.63	4.79	4.62	31.75
	KKG-DT2	87.92	67.08	66.67	12.67	4.84	4.71	13.02
	Art-CT	47.50	43.33	32.92	2.50	2.45	1.86	28.89
	Kone-CT	76.25	59.17	32.08	3.97	2.79	0.51	35.53
	KKG-CT1	91.25	74.17	67.92	6.52	4.96	7.22	19.13
	KKG-CT2	88.75	72.92	67.08	5.84	4.90	7.03	12.87

4.8 Conclusions

This chapter presents two BIP discrete-time models (KKG-DT1, KKG-DT2) and two MIP continuous-time formulations (KKG-CT1, KKG-CT2) followed by a detailed comparison with four existing literature models using 2760 RCPSPs. Overall, KKG-CT1 and KKG-CT2 have been found to be the best models for the problems addressed. The detailed and comprehensive computational analysis reveals some interesting features of the problems solved and the mathematical models considered. More specifically, it has been observed that

problem instances with low RS values and high RF values are the hardest to solve, according to the computational results in problem sets j30 and j60. Moreover, it seems that increasing the number of activities has a direct negative effect on the computational performance of continuous-time models since the number of sequencing binary variables is increased dramatically, while discrete-time models are affected less due to the fact that the number of the decisions variables depends mainly on the time points. Notice that, in contrast to continuous-time models, the performance of discrete-time models is strongly affected (positively) from the bounds imposed on the preprocessing phase. In addition, continuous-time formulations feature lower Gap (%) values compared with discrete-time models. Also, the event-based Kone-CT formulation generally performs poor and becomes inappropriate for large number of activities due to the huge model size. Finally, it has been demonstrated that continuous-time models, which have received little attention in the literature so far, could deal very successfully and efficiently with RCPSPs.

At this point, it should be emphasised that in discrete-time approaches: (a) the scheduling horizon is divided into a finite number of time intervals with predefined duration, and therefore (b) activities can start or end only at the boundaries of these time periods. For this reason, in real-world problems where the duration of the activities may not be integer numbers, a finer division of the scheduling horizon (resulting in bigger model sizes), and/or duration time rounding (sacrificing optimality) should be employed. An inherent special advantage of continuous-time approaches is that timing decisions can be represented explicitly. However, the modelling of resource limitations needs more complicated constraints involving many big-M terms, which tends to increase the model complexity.

4.9 Nomenclature

Indices/Sets

$i, j \in A := \{0,...,n+1\}$	activities (including start *0* and end $n+1$ dummy activities)
$k \in \Re := \{1,...,m\}$	renewable resources
$t \in T := \{0,...,h\}$	time points
$e \in J := \{1,...,n\}$	event points

Subsets

B	activity pairs $(i, j) \in V^2$ sharing at least one renewable resource
C	transitive closure of subset E
D	pairs of activities $(i, j) \notin C$ where according to the preprocessing phase $LF_i \leq ES_i$
E	pairs of activities (i, j) where j is an immediate successor of activity i; $E \subseteq C$
G	pairs of activities $(i, j) \in V^2$ that cannot be processed simultaneously due to resource capacity limitations, $G \subseteq B$
K	pairs of activities with known precedence relations, $K = (C \cup D)$
P	pairs of activities that could overlap, $P = B \setminus (G \cup C \cup D)$
S	pairs of activities that cannot overlap due to resource capacity limitations, excluding those with known precedence relations, $S = G \setminus (C \cup D)$
$V := \{1,...,n\}$	set of non-dummy activities, $V \subset A$

Parameters

EF_i	earliest finishing time of activity i
ES_i	earliest starting time of activity i

LF_i latest finishing time of activity i

LS_i latest starting time of activity i

r_{ik} renewable resource k requirements for activity i

R_k maximum capacity of renewable resource k

ζ_i number of predecessors of activity i

θ_i number of successors of activity i

λ a small number (= 0.1 in this work)

p_i duration of activity i

Continuous Variables

s_i starting time of activity i

f_i finishing time of activity i

d_e time of event point e

q_{ijk} quantity of resource k transferred from activity i (at the end of its processing) to activity j (at the beginning of its processing)

Binary Variables

v_{ie} = 1, if activity i starts at event point e, or if it is still being processed immediately after event point e

w_{it} = 1, if activity i is being processed in the time interval between time points t and $t+1$

x_{ij} = 1, if activity i is completed before activity j starts (remark: in KKG-CT1 and KKG-CT2, for activities $(i, j) \in P$ this definition is relaxed, and $x_{ij} = 1$ if activity i starts before activity j)

y_{it} = 1, if activity i starts at time point t

z_{ij} = 1, if activity i is overlapped by activity j

Chapter 5

Mathematical Formulation for the Resource-Constrained Project Scheduling Problem with Generalised Precedence Relations

This chapter presents a new precedence-based continuous-time formulation for a challenging extension of the standard single-mode resource-constrained project scheduling problem that also considers minimum and maximum time lags (RCPSP/max), under the objective of minimizing the project makespan. The proposed linear mixed integer programming model is based on the definition of two types of binary variables to express: (i) the relative sequencing of activities, and (ii) the overlapping conditions. Two types of continuous variables for activity starting and finishing times are also used. Additionally, a number of new activity subsets are introduced for modelling purposes, in order to reduce the model size and tighten the proposed model. The new mathematical formulation is used to conduct an extensive computational study on a total of 2,250 well-known and open-accessed benchmark problem instances from the literature. Various problem sizes are considered in the test sets involving 10, 20, 30, 50 and 100 activities. The computational results illustrate the efficient performance of the proposed mathematical formulation. Finally, interesting observations are made through the computational study and potential future research lines are proposed.

5.1 Introduction

The classical single-mode RCPSP uses finish-start precedence relations with zero time-lags, which means that an activity can only start as soon as all its predecessor activities have finished. In this chapter we study a challenging

extension of the RCPSP in which activity starting times are constrained by *generalised precedence relations* (also called *temporal constraints* or *minimum and maximum time lags* or *time windows*). This extension is denoted as *RCPSP/max* or $PS \mid temp \mid C_{max}$, using the notation of Brucker et al. (1999). More specifically, the $PS \mid temp \mid C_{max}$ notation specifies the single-mode project scheduling problem (PS) under general temporal constraints given by minimum and maximum start-start time lags between activities ($temp$) while minimizing the makespan of the project (C_{max}).

From a modelling point of view, time lags are necessary to represent partial or total overlapping of activities, release or ready times, project milestones and deadlines, time windows or time-varying resource requirements. For even more applications of GPRs to modelling real-life project scheduling problems, we refer the reader to Neumann and Schwindt (1995).

The addition of maximal time lags to the classical RCPSP significantly increases the complexity of the problem. Moreover, the generation of problem instances could be problematic because infeasible problems might be generated. The RCPSP/max is an NP-hard problem and even the theoretically easier problem of checking its feasibility is NP-complete (Hartmann and Briskorn, 2010).

The main contribution of this chapter is that it efficiently solves the RCPSP/max using a pure mathematical programming approach. It is an extension of the KKG-CT2 model introduced in the previous chapter, and to the best of our knowledge, it is the only continuous-time approach for the RCPSP/max in the OR literature. The only other mathematical model for the RCPSP/max was proposed by Bianco and Caramia (2012b), but it is a discrete-time one. The formulation is used to calculate a lower bound through a Langrangean relaxation and a branch-and-bound algorithm which exploits both.

The rest of the chapter is organised as follows. In Section 5.2, the RCPSP/max problem is formally stated. Section 5.3 describes the preprocessing phase employed in this work. Section 5.4 describes in detail the new continuous-time Mixed-Integer Programming (MIP) formulation. A brief description of the problem instance sets considered and a comprehensive computational study are presented in Section 5.5. Finally, concluding remarks are drawn in Section 5.6.

5.2 Problem statement

The single-mode *RCPSP/max* consists of a set of n activities $V = \{1,...,n\}$, which have to be processed without interruption, under the objective of *minimising the project makespan*. For modelling purposes, dummy activities *0* and $n+1$ are defined, to represent the beginning and completion of the project. Set $A = \{0,1,...,n,n+1\}$ defines the new complete set of activities. The processing time of activity i is denoted by p_i, (for dummy activities $p_0 = p_{n+1} = 0$).

The project utilises a set of renewable resource types $k \in \Re := \{1,...,m\}$. Each activity i requires r_{ik} units of resource k during each time period of its execution ($r_{0k} = r_{n+1,k} = 0$). The maximum available capacity of each resource k is R_k units. Renewable resources fully retrieve the occupied resource amount after the completion of each activity. In other words, the temporary availability of the renewable resources at every time is constrained.

The modelling of activities with a piecewise constant resource requirement $r_{ik}(t)$ (i.e., activity i requires an amount $r_{ik}(t)$ of resource $k \in \Re$, during the t th time unit it is in process) can be easily implemented by replacing them with several other activities which are "tied together" through temporal constraints. Similarly, time-varying resource availability $R_k(t)$ (i.e., a specific

quantity $R_k(t)$ of resource k is available at time t) can be modelled by introducing dummy jobs with fixed start times which consume the excessive resource amount. Therefore, it is safe to assume that $r_{ik}(t)$ and $R_k(t)$ are constant over time without any loss of generality (Fest et al., 1998).

Time lags d_{ij} between activity pairs (i, j) are included in set E and take integer values (i.e., $d_{ij} \in Z$). Time lags can be presented as an activity-on-node (AON) network $G(A, E, d)$, where nodes correspond to the set of activities A and set E represents the arcs with weight d. Minimal lags are usually represented as forward arcs and maximal lags as backward arcs. Given that the various types of minimum and maximum time lags (SS, SF, FS, FF) can be transformed to a single type using the rules by Bartusch et al. (1988), from now on we will consider them to be of type *start-start*. Time lags can be represented in the following general form:

$$s_j - d_{ij} \geq s_i \qquad \forall (i, j) \in E$$

where s_i (s_j) is the starting time of activity i (j). The case $d_{ij} \geq 0$ corresponds to a *minimum time lag* of d_{ij} units, stating that activity j has to start at least d_{ij} time units after the start time of activity i. The case $d_{ij} \leq 0$ corresponds to *maximum time lag* of d_{ij} units, stating that activity i has to start at the latest $-d_{ij}$ time units after the start time of activity j. By exploiting time lags, we can define a set L containing pairs of activities (i, j) that have *real precedence relations* (i.e., the finishing time of activity i is smaller or equal to the starting time of j). Specifically, when $d_{ij} \geq p_i$ holds, activity j cannot begin earlier than the finish of activity i ($s_i + p_i = f_i$, with f_i the finishing time of activity i).

A solution for the RCPSP/max is a list of start times or schedule $S = (s_0, s_1, \ldots, s_n, s_{n+1})$, where $s_0 = 0$, and s_{n+1} corresponds to the makespan of the project.

5.3 Preprocessing

In this section some fast computable bounds are considered, however other bounding techniques could be used. First, a trivial *upper bound UB* equal to $\sum_{i \in A} \max\left(p_i, \max_{(i,j) \in E} \left(d_{ij} \right) \right)$ is calculated.

Defining an *earliest starting time* (ES_i) for each activity i is also a common preprocessing step. It can be computed using the first of the Special Implementations of the *Modified Label Correcting Algorithm* of Ahuja et al. (1993); which is of $O\left(|A||E| \right)$ time complexity. A label correcting algorithm is iterative and assigns tentative distance labels π_j to nodes at each step. The distance labels are estimates of (i.e., lower bounds on) the longest path distances and are considered as temporary until the final step where π_j is the longest path length from the source node 1 to node j. In each pass, arcs $(i, j) \in E$ are scanned one by one, and condition $\pi_j < \pi_i + d_{ij}$ is checked. If the arc satisfies this condition, $\pi_j = \pi_i + d_{ij}$ is updated. The algorithm stops when no distance label changes during an entire pass. Finally, we set $ES_j = \pi_j$. The algorithm runs as displayed in Algorithm 1.

Through the ES_i, we can easily calculate the earliest finishing times EF_i by adding the activity duration ($EF_i = ES_i + p_i$). We can also set a trivial *lower bound* on the time horizon, equal to the earliest finishing time of the dummy end activity ($LB = EF_{n+1}$). It should be noted that other preprocessing methods could be used, potentially yielding better bounds at the expense of longer CPU time. Here, we have chosen a relatively simple method that requires negligible computational effort.

Algorithm 1. Modified label-correcting algorithm

algorithm *modified label-correcting;*
begin
$\pi(0) := 0$ and $pred(0) := 0$;
$\pi(j) := -\infty$ for each node $j \in A - \{0\}$;
$nochange := false$;
while $nochange \neq true$ **do**
begin
$nochange := true$;
for each $arc(i, j) \in E$ **do**
begin
if $\pi_j < \pi_i + d_{ij}$ **then**
begin
$\pi_j = \pi_i + d_{ij}$;
$nochange := false$;
end;
end;
end;
end;

5.4 The mathematical model

The proposed MIP mathematical formulation is an extension of the KKG-CT2 *continuous-time* model introduced in the previous chapter. It also relies on the definition of *binary sequencing variables* to represent activity ordering (x_{ij}) and overlapping (z_{ij}), and *continuous variables* for activity starting (s_i) and finishing (f_i) times.

However, we now introduce a number of different sets, containing activity pairs for modelling purposes:

(i) set E contains activity pairs (i, j) with generalised precedence relations, such that $s_j - d_{ij} \geq s_i$,

(ii) set L contains activity pairs with *real precedence relations*, calculated using the time lags (i.e. $(i,j): d_{ij} \geq p_i$),

(iii) set C is the transitive closure of immediate predecessor subset L,

(iv) set B contains all pairs of activities $(i,j) \in V^2$ sharing at least one renewable resource and

(v) set G contains all pairs of activities $(i,j) \in B$ that cannot be processed simultaneously due to resource capacity limitations (i.e., $r_{ik} + r_{jk} > R_k$).

Two more composite sets are defined by using the previously defined sets of activities. The first set $S = G \setminus C$ contains all pairs of activities that cannot overlap due to resource capacity limitations, excluding those with known precedence relations. Notice that for this set only variables x_{ij} need to be defined. The second set $P = B \setminus (G \cup C)$ contains all pairs of activities that could overlap. For such activity pairs, both x_{ij} and z_{ji} variables need to be defined. The preprocessing time to determine sets B, C, G, L, P and S is negligible. Using the above definition of sets and decision variables, the following mathematical formulation is proposed:

$$\min C_{\max} = f_{n+1} \tag{1}$$

$$s_0 = 0 \tag{2}$$

$$f_i = s_i + p_i \qquad \forall i \in (V \cup \{n+1\}) \tag{3}$$

$$f_i \leq s_j \qquad \forall i \in V, \forall j \in (V \cup \{n+1\}): (i,j) \in C \tag{4}$$

$$s_j - d_{ij} \geq s_i \qquad \forall (i,j) \in E \tag{5}$$

$$f_j \leq s_i + (M - ES_i) \cdot (1 - x_{ji}) \qquad \forall (i,j) \in S : i > j \tag{6}$$

$$f_i \leq s_j + (M - ES_j) \cdot x_{ji} \qquad \forall (i,j) \in S : i > j \tag{7}$$

$$s_j \leq s_i + (M - ES_i) \cdot (1 - x_{ji}) \qquad \forall (i,j) \in P : i > j \tag{8}$$

$$s_i + \lambda \leq s_j + (M - ES_j) \cdot x_{ji} \qquad \forall (i,j) \in P : i > j \tag{9}$$

$$f_j - s_i \leq (M - ES_i) \cdot z_{ji} \qquad \forall (i,j) \in P : i \neq j \tag{10}$$

$$r_{ik} + \sum_{\substack{j<i:r_{jk}>0 \\ \forall(i,j)\in P}} r_{jk}\left(z_{ji} + x_{ji} - 1\right) + \sum_{\substack{j>i:r_{jk}>0 \\ \forall(i,j)\in P}} r_{jk}\left(z_{ji} - x_{ij}\right) \le R_k \qquad \forall i \in V, k \in R : r_{ik} > 0 \qquad (11)$$

$$x_{ji} \le z_{ij} \qquad \forall(i,j) \in P : i > j \qquad (12)$$

$$1 - x_{ji} \le z_{ji} \qquad \forall(i,j) \in P : i > j \qquad (13)$$

$$x_{ji} \in \{0,1\} \qquad \forall(i,j) \in (P \cup S) : i > j$$

$$z_{ji} \in \{0,1\} \qquad \forall(i,j) \in P : i \neq j \qquad (14)$$

$$s_i \ge ES_i,\ f_i \ge EF_i \qquad \forall i \in (V \cup \{n+1\})$$

Equation (1) minimises the completion time of dummy end activity $n+1$, representing the project completion. Note that the makespan is lower than or equal to the upper bound UB calculated in the preprocessing phase. The starting time of the dummy start activity is equal to zero, according to constraint (2). Constraints (3) correlate the starting and finishing times for activities $i \in (V \cup \{n+1\})$ throughout their execution. Constraints (4) set all a priori known precedence relations between activities $(i,j) \in C$, through their starting and finishing times. Minimum and maximum time lags d_{ij} are imposed by constraints (5).

Big-M constraints (6) - (9) define the sequencing constraints between any pair of activities $(i,j) \notin C$ when $i > j$. For the big M parameters a value equal to $UB+1$ is used, with UB being the upper bound to the time horizon calculated in the preprocessing phase. To reduce the solution search space, the value of the big M parameters is properly adjusted by the ES of the corresponding activity.

Constraints (8) and (9) extend the definition of sequencing binary variable x_{ij} to also control the starting times of the parallel tasks i and j. In the case studies, parameter $\lambda = 0.1$. Overlapping condition (C) is modelled by constraints (10), and renewable resource constraints are given by constraints (11). By splitting the resource demand into two separate summations

containing $(z_{ji} + x_{ji} - 1)$ when $j < i$ and $(z_{ji} - x_{ij})$ when $j > i$, we are able to detect overlapping between activity pairs $(i, j) \in P$ and properly model the resource constraints.

Constraints (8) - (11) state that if activity i starts processing before activity j (i.e., $x_{ij} = 1$), then activity j has to finish after activity i starts (i.e., $z_{ji} = 1$). The only case that one of the parentheses $(z_{ji} + x_{ji} - 1)$ and $(z_{ji} - x_{ij})$ in the resource constraints (10) takes the value of *1* is when activity j overlaps i (i.e., $z_{ji} = 1$) and activity j starts processing before i (i.e., $x_{ji} = 1$). It can be easily proven that for activities which could overlap $(i, j) \in P$, if $x_{ij} = 1$ then $s_i \leq s_j < f_j$, and consequently $f_j - s_i > 0$ which according to constraints (10) gives $z_{ji} = 1$. Table 5.1 displays all possible combinations for variables z_{ji} and x_{ij} for activities $(i, j) \in P$ that could be executed in parallel. In the first two cases wherein $z_{ji} = 0$, no overlapping occurs since activity i starts after the completion of activity j (i.e., $s_j < f_j \leq s_i$). Especially, in the second case notice that if $z_{ji} = 0$, $x_{ij} = 1$ and $x_{ji} = 0$ that implies $f_j \leq s_i < s_j$, which is obviously impossible. For this reason, if $z_{ji} = 0$ then always $x_{ij} = 0$. In the last two cases wherein $z_{ji} = 1$, overlapping occurs *if and only if* activity j begins processing before activity i starts (i.e., $x_{ij} = 0$), because in that case holds $s_j < s_i \leq f_j$.

Table 5.1 Modelling of binary variables z_{ji} and x_{ij} for activities $(i, j) \in P$

Constraints (10)		Constraints (8) – (9)			Constraints (11)	
$f_j - s_i$	z_{ji}	$s_i - s_j$	x_{ij}	x_{ji}	$(z_{ji} + x_{ji} - 1)$	$(z_{ji} - x_{ij})$
≤ 0	0	> 0	0	1	0	0
≤ 0	0	< 0	1	0	impossible	impossible
> 0	1	< 0	1	0	0	0
> 0	1	> 0	0	1	1	1

Constraints (12) and (13) are tightening constraints that correlate directly sequencing and overlapping binary variables. Finally, the domain of the decision variables is given by constraints (14).

In summary, the proposed continuous-time MIP model for the RCPSP/max consists of constraints (1) - (14).

5.5 Computational Results

In this section we present the problem sets solved by the proposed mathematical formulation and an extended computational study. All problem instances have been solved on an *Intel Core i5 CPU M430 2.27GHz with 4GB RAM* using *CPLEX 11.1.1* via a *GAMS 22.8.1* (Rosenthal, 2012) *WIN 6007.6015 VIS* interface under standard configurations. A maximum resource time limit of *600 CPU s* has been set for all problem instances.

5.5.1 Description of Problem Sets

Three widely-used test sets consisting of a total of 2,250 problem instances have been solved by the proposed mathematical formulation. These sets were generated by ProGen/max that uses two generating methods: DIRECT, which directly generates entire projects, and CONTRACT, which first generates cycle structures, upon which the (acyclic) contracted project network is generated. At this point, we point out that the generator does not always produce instances with a feasible solution. These instances are included in the computational study, since proving their infeasibility is an NP-complete problem.

The *first set* was taken from Schwindt (1998a) and contains 270 problems for each network size of $n = \{10,20,30\}$ activities. These problem sets will be referred to as *j10*, *j20* and *j30*, for $n = \{10,20,30\}$ respectively. The *second set*, from Franck et al. (2001) includes 90 instances for each network size with

$n = \{10,20,50,100\}$. From now on, these test sets will be referred to as *ubo10*, *ubo20*, *ubo50* and *ubo100*, depending on the number of activities. Finally, the *third set* was generated by Schwindt (1996) and it contains 1080 instances with 100 activities. We will refer to this set as *CD* in our study. The three sets are summarised in Table 5.2.

Table 5.2 Test instance characteristics

Set	# of activities	# of instances	# of infeasible instances*	% of infeasible instances*
j10	10	270	83	30.74
j20	20	270	86	31.85
j30	30	270	85	31.48
ubo10	10	90	17	18.89
ubo20	20	90	20	22.22
ubo50	50	90	17	18.89
ubo100	100	90	12	13.33
CD	100	1,080	21	1.94
Total		2,250	341	15.16

* problems that are reported to be infeasible

5.5.2 Computational Study Results

Before presenting the results, we briefly discuss the notation used. *Solved instances* (%) reports the percentage of problems that gave a feasible solution or were proven infeasible (*Feasible* and *Proven Infeasible*). *Optimally Solved Instances* (%) represents the percentage of proven solutions using the suggested formulation, optimal or infeasible (*Proven Optimal* and *Proven Infeasible*). *Feasible* (%) is the percentage of instances for which a feasible solution was found (optimal or not). *Proven Optimal (%)* reports the percentage of problems for which an optimal solution was found within the predefined time limit. *Proven Infeasible (%)* is the percentage of reported infeasible problems that were proven by the proposed model. *Average CPU* (s) is the average computational time in CPU seconds for all instances in the

set, while *Optimal CPU* (s) considers only the Optimally Solved Instances. Table 5.3 presents the computational results for all test sets solved.

Table 5.3 Computational results for all instances including reported infeasible, within predefined time limit 600 CPU s

Set	Solved Instances (%)	Optimally Solved Instances (%)	Feasible (%)	Proven Optimal (%)	Proven Infeasible (%)	Average CPU (s)	Optimal CPU (s)
j10	100.00	100.00	69.26	69.26	30.74	0.06	0.06
j20	99.63	95.56	67.78	63.70	31.85	31.24	4.79
j30	97.04	85.56	65.56	54.07	31.48	99.60	15.11
ubo10	100.00	100.00	81.11	81.11	18.89	0.06	0.06
ubo20	100.00	100.00	77.78	77.78	22.22	0.59	0.59
ubo50	93.33	78.89	74.44	60.00	18.89	186.44	75.74
ubo100	44.44	22.22	31.11	8.89	13.33	483.53	70.54
CD	54.26	43.33	53.43	42.50	0.83	425.73	197.54

All reported infeasible instances were proven, except those of the *CD* test set, wherein 9 out of 21 (0.83%) where identified (see Tables 5.2 and 5.3). All instances in sets *j10*, *ubo10*, *ubo20* and most (95.56%) in *j20* are easily solved to optimality with few computational requirements. As the number of activities increases in sets *j30* and *ubo50*, we observe that the formulation still finds a high number of optimally solved instances (second column of Table 5.3), with percentages 85.56% and 78.89% respectively. When taking into account all feasible solutions (first column of Table 5.3), the corresponding percentages rise to 97.04% and 93.33%. Finally, for the larger sets *ubo100* and *CD*, which involve 100 activities, the percentage of solved instances drops to 44.44% and 54.26%, respectively. For the *CD* set, the number of optimal solutions (43.33%) is almost twice as that of the *ubo100* set (22.22%). It is worth noticing that the *Optimal CPU* time which refers to optimally solved instances is quite lower than the total *Average CPU* time. This is because instances for which a solution was not found or non-proven deplete the 600 CPU s time limit.

Table 5.4 displays the computational results normalised to the reported feasible instances (i.e., excluding the reported infeasible instances). *Feasible instances solved* (%) represents the percentage of feasible solutions found.

Good solutions (%) reports the percentage of optimal and non-proven solutions with a deviation from the reported optimal or best solution less than 3%. *Optimal* (%) are the optimal solutions found and *Average gap* (%) is the average gap reported by the solver, for non-proven solutions only.

Table 5.4 Results for feasible instances

Set	Feasible instances solved (%)	Good solutions (%)	Optimal (%)	Average gap (%)
j10	100.00	100.00	100.00	0.00
j20	99.46	94.02	93.48	10.58
j30	95.68	84.32	78.92	17.40
ubo10	100.00	100.00	100.00	0.00
ubo20	100.00	100.00	100.00	0.00
ubo50	91.78	83.56	73.97	12.91
ubo100	35.90	30.77	10.26	10.43
CD	54.49	49.67	43.34	7.18

Notice that for sets with up to 50 activities, although the percentage of optimal solutions decreases as the number of activities increases, the number of feasible instances solved and good solutions remains high.

Through the computational study, the proposed formulation found a number of solutions that are better than the ones reported in the online Project Scheduling Problem Library (PSPLIB). In Table 5.5, we present the total number of problems for which we found a better solution (*# of better solutions*), the number of optimal solutions that were lower than the best upper bound achieved (*# of better optimal*), the number of problems for which the upper bound is now proven to be the optimal solution (*# of extra proven optimal solutions*) and in the final column we display the number of *better non-proven optimal solutions* found.

For sets *j10* and *ubo10* all optimal solutions are reported, so no better results are reported. Set *ubo20* contained 4 instances without a reported optimal solution. Those solutions were either found or proven by our formulation. For

project networks with 30 or more activities, a substantial number of better solutions has been found. Especially in set *ubo50*, that number consists of about 1/3 of the problems (29 out of 90).

Table 5.5 Number of solutions better than the ones reported in literature

Set	# of better solutions	# of better optimal	# of extra proven optimal solutions	# of better non-proven solutions
j10	0	0	0	0
j20	15	1	13	1
j30	36	4	24	8
ubo10	0	0	0	0
ubo20	4	2	2	0
ubo50	29	11	11	7
ubo100	10	1	0	9
CD	5	0	2	3

Table 5.6 presents an analysis of the instances for which an optimal solution was found, which was lower than the reported upper bound, along with the corresponding CPU time.

Table 5.6 Optimally solved instances with solution lower than the reported upper bound

Set	Instance #	Our solution	Best reported	CPU Time
j20	150	46	47	289.12
ubo20	15	45	46	12.18
	20	65	66	1.65
j30	32	113	114	3.1
	139	88	89	2.82
	170	95	96	7.63
	195	55	58	598.06
ubo50	14	153	168	159.81
	18	163	164	45.99
	31	302	308	24.24
	37	229	232	7.04
	42	147	148	256.56
	45	181	187	223.66
	49	145	153	504.86
	52	137	139	78.12
	57	132	133	186.92
	67	243	246	11.17
	68	275	278	4.02
ubo100	68	538	540	45.43

As an illustrative example, the Gantt chart and resource profiles for instance 195 of the j30 problem set in displayed in Figure 5.2.

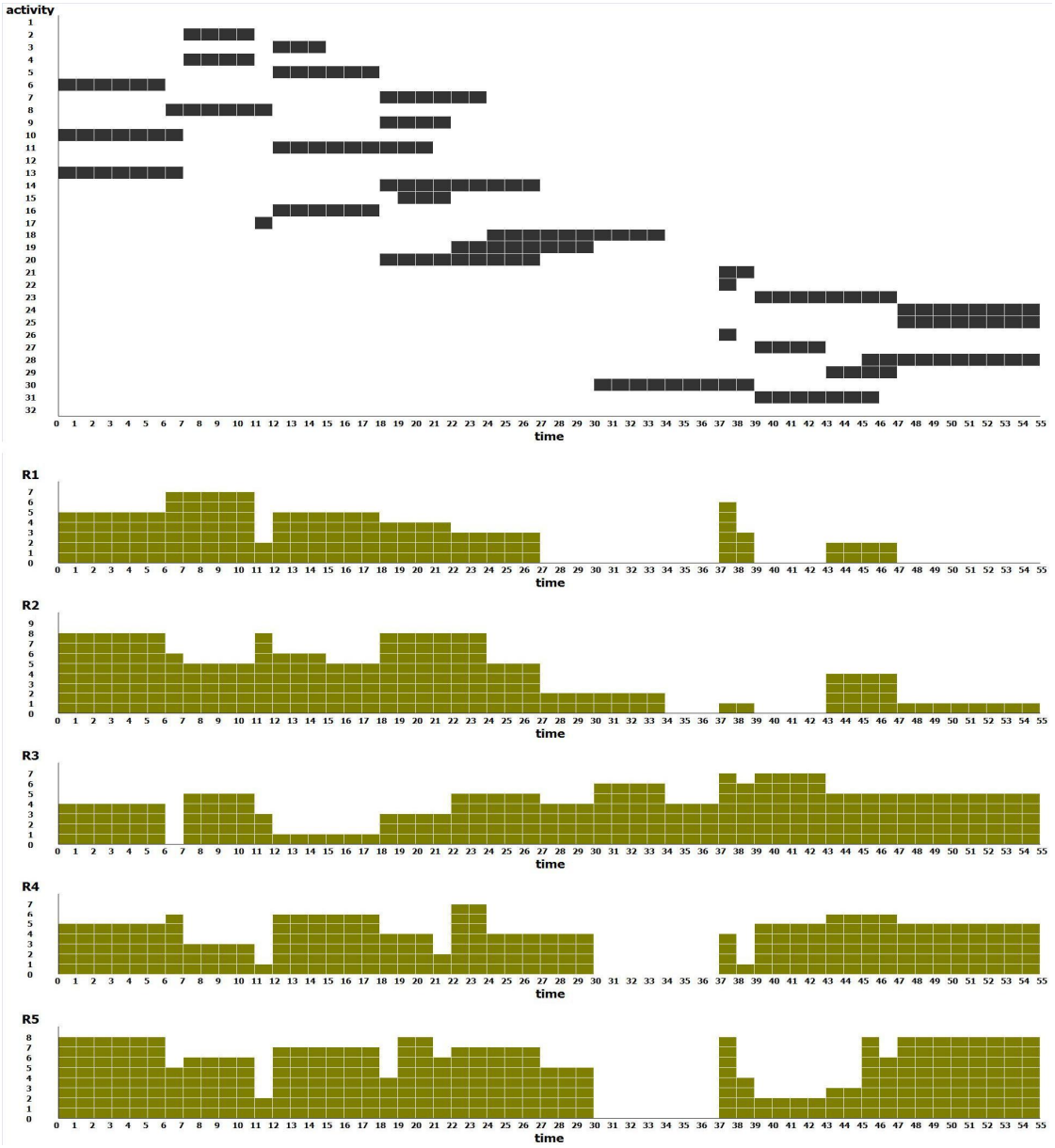


Figure 5.1. Illustrative example for better optimal solution (Set *j30*, Instance 195)

Table 5.7 lists the instances for which a better solution than the best reported upper bound was found. Note that in some cases the new solution is significantly better than the reported one. Instance 33 of set j30 and instance 67 of ubo100 displayed the greatest improvement in percentage on the upper bound with 15.56% and 10.24%, respectively. The improvement percentage is calculated by *(Reported UB – New UB)/Reported UB*.

Table 5.7 Non-proven optimal instance with a better solution than the reported upper bound

Set	Instance #	New solution			Reported solution		UB Improvement (%)
		LB	UB	Gap (%)	LB	UB	
j20	167	49.00	50	2	42	52	3.85
j30	4	94.05	101	6.88	84	104	2.88
	33	94.00	114	17.54	81	135	15.56
	38	76.22	90	15.31	63	93	3.23
	40	100.70	114	11.67	80	120	5.00
	65	110.08	162	32.05	104	163	0.61
	78	58.91	63	6.49	56	64	1.56
	123	121.43	150	19.05	91	151	0.66
	175	68.10	70	2.71	59	71	1.41
ubo50	6	203.99	216	5.56	202	232	6.90
	9	165.01	217	23.96	170	230	5.65
	11	140.00	145	3.45	139	146	0.68
	34	193.34	228	15.2	170	232	1.72
	36	171.00	207	17.39	149	218	5.05
	54	88.00	90	2.22	88	91	1.10
	65	177.23	199	10.94	169	215	7.44
ubo100	8	350.92	430	18.39	351	435	1.15
	36	354.01	437	18.99	364	457	4.38
	37	369.86	450	17.81	372	453	0.66
	38	476.00	477	0.21	473	483	1.24
	62	498.82	531	6.06	500	540	1.67
	64	523.73	536	2.29	522	538	0.37
	67	352.59	412	14.42	335	459	10.24
	73	391.99	398	1.51	394	414	3.86
	87	362.89	367	1.12	361	368	0.27
CD	760	631.74	679	6.96	661	685	0.88
	817	550.96	569	3.17	558	575	1.04
	992	577.98	615	6.02	578	616	0.16

5.6 Conclusions

This chapter proposes a new continuous-time formulation for the RCPSP/max. The suggested mathematical model is based on the concept of overlapping conditions, which is modelled through sequencing and overlapping variables. In our approach, the model performance is enhanced through simple (i.e., with low computational effort) preprocessing techniques that include the calculation of upper/lower bounds and earliest starting/finishing times. The proposed formulation was tested on 2,250 well-known test instances from the

literature, with problems sizes varying from 10 to 100 activities. The computational results revealed a number of solutions that are better than the ones reported in PSPLIB.

All reported infeasible instances were proven, with the exception of the *CD* test set, wherein 9 out of 21 where identified. For projects with up to 30 activities, the proposed formulation solved optimally most problems. Problem sizes of 50 activities are also dealt efficiently with a feasible solution for 91.78% and a good solution for 83.56% of the solvable instances. Finally, for the larger sets *ubo100* and CD that involve 100 activities, the reported results were quite satisfactory for a solution method that relies on a mathematical formulation taking into consideration the problem sizes. This deteriorating model performance is caused by the larger number of activities, which leads to a polynomial increase in the number of sequencing and overlapping variables.

With the increasing computational power available, even in personal computers, mathematical solvers are becoming capable of dealing with larger problems than in the past. Given that the RCPSP/max is an offline problem, we can easily accept to increase the time limit of the solver and obtain optimal or good suboptimal solutions.

5.7 Nomenclature

Indices/Sets

$i, j \in A := \{0,1,...,n+1\}$	activities
$k \in \Re := \{1,...,m\}$	renewable resources

Subsets

B	set of activities $(i, j) \in V^2$, sharing at least one renewable resource
C	transitive closure of subset L
E	pairs of activities with temporal constraints

G	set of activities $(i, j) \in V^2$ that cannot be processed simultaneously (due to resource capacity limitations); $G \subseteq B$
L	activity pairs (i, j) where j is an immediate successor of activity i; $L \subseteq C$
P	activity pairs that could overlap; $P = B \setminus (G \cup C)$
S	activity pairs that cannot overlap due to resource capacity limitations, excluding those with known precedence relations; $S = G \setminus C$
$V := \{1,...,n\}$	set of non-dummy activities i; $V \subset A$

Parameters

r_{ik}	renewable resource k requirements for activity i
R_k	maximum capacity of renewable resource k
λ	a small number (= *0.1*)
M	a large number (= *UB+1*)
ES_i	earliest starting time of activity i defined as $\max_{(i,j)\in E}(ES_j + l_{ij})$
EF_i	earliest finishing time of activity i defined as $ES_i + p_i$
l_{ij}	time lag between activities $(i, j) \in E$
p_i	processing time of activity i
UB	upper bound on time horizon
LB	lower bound on time horizon

Continuous Variables

s_i	starting time of activity i
f_i	finishing time of activity i

Binary Variables

x_{ij}	= 1, if activity i is completed before activity j starts, (remark: for activities $(i, j) \in P$ this definition is relaxed and $x_{ij} = 1$ if activity i starts before activity j)
z_{ij}	= 1, if activity i finishes after activity j starts

Chapter 6

Conclusions and future work

In this chapter, we summarise the main contributions of this dissertation and discuss possible future research directions that seem promising.

6.1 Conclusions

The objective of this dissertation is to establish a new representation and solution framework for RCPSPs utilising techniques from the process scheduling area. For this reason, a new network representation method has been developed, based on the well-known RTN, as well as a number of mathematical programming formulations for the RCPSP and its variants. The computational results displayed the efficient performance of the formulations, which outperform state-of-the-art models from the literature.

- In Chapter 3, we proposed a new network representation method and new MILP models for the RCPSP and the MRCPSP, based on the well-known RTN process representation from the process scheduling industry. The formulations were used to solve some project scheduling problems from the online PSPLIB. The computational results of this initial approach, indicated that for problems involving more than 30 activities, the proposed MILP models cannot lead to a global optimal solution in reasonable computational times. However, the proven similarities between process and project scheduling problems suggested that exchanging solution techniques between the two research fields is both possible and useful. In summary, the main objective of establishing a new framework for RCPSPs utilising techniques from the process scheduling area was achieved.

- In Chapter 4, four very efficient discrete- and continuous-time formulations were developed, followed by a detailed comparison with four state-of-the-art models from the literature, using 2760 RCPSPs. Also, a number of computationally inexpensive preprocessing steps were taken to enhance and simplify the formulations. Overall, the continuous-time models have been found to be the best models for the problems addressed. The detailed and comprehensive computational analysis revealed some interesting features of the problems solved and the mathematical models considered. In particular, it has been observed that problem instances with low RS values and high RF values are the most difficult to solve, according to the computational results in problem sets j30 and j60. Moreover, it seems that increasing the number of activities directly affects negatively the computational performance of continuous-time models since the number of sequencing binary variables is increased dramatically, while discrete-time models are affected less due to the fact that the number of the decisions variables principally depends on the time points. Notice that, in contrast to continuous-time models, the performance of discrete-time models is strongly affected (positively) from the bounds imposed on the preprocessing phase. In addition, continuous-time formulations feature lower Gap (%) values compared with discrete-time models.

- In Chapter 5, a new continuous-time formulation for the RCPSP/max was proposed. The suggested mathematical model is an extension of the KKG-CT2 model introduced in Chapter 4. The model performance was enhanced through simple (i.e., with low computational effort) preprocessing techniques that include the calculation of upper/lower bounds and earliest starting/finishing times. The proposed formulation was tested on 2,250 well-known test instances from the literature, with problems sizes varying from 10 to 100 activities. The computational results revealed a number of solutions that are better than the ones reported in PSPLIB. All reported infeasible instances were proven, with the

exception of the *CD* test set, wherein 9 out of 21 where identified. For projects with up to 30 activities, the proposed formulation solved optimally most problems. Problem sizes of 50 activities are also dealt efficiently with a feasible solution for 91.78% and a good solution for 83.56% of the solvable instances. Finally, for the larger sets *ubo100* and CD that involve 100 activities, the reported results were quite satisfactory for a solution method that solely relies on a mathematical formulation taking into consideration the problem sizes.

6.2 Future Work

The results of this dissertation revealed a range of issues and pointed to several interesting directions for future work:

- Further research in devising other continuous- and discrete-time mathematical modelling approaches seems promising and highly challenging.
- Of particular interest would be the comparison of the proposed formulations with more mathematical models from the literature and hopefully some new ones, using the same and possibly larger RCPSP problems (i.e. sets j90 and j120). A more extensive computational study could provide us with further insight regarding the performance of the available mathematical frameworks, and could inspire the development of new modelling approaches.
- Since the new continuous-time models proposed in chapters 4 and 5, have proven to be efficient for solving the RCPSP and the RCPSP/max, a possible extension to the multi-mode RCPSP could be considered.
- The proposed models could be the core element of new decomposition methods, such as the one proposed by Kopanos et al. (2010), or part of hybrid methods, in an attempt to reduce the

computational burden of the complicated problems solved and allow us to deal with larger problems.

- Given that the RCPSP is an offline problem, we can easily accept an increase on the time limit of the solver and obtain more optimal and/or good suboptimal solutions.
- The RCPSP has a number of interesting variants, that could be addressed. For example, since project scheduling is dynamic in nature, the consideration of *uncertainty* also arises as a challenging future research task. Three main categories of approaches can be distinguished: proactive, reactive and stochastic.
- Another interesting extension is the *Resource Levelling Problem* (RLP), which arises whenever it is expedient to reduce the fluctuations in patterns of resource utilizations over time, while maintaining compliance with a prescribed project completion time. In particular, in cases where even slight variations in resource needs represent financial burden or heightened risks of accidents, a resource levelling approach helps to schedule the project activities so that the resource utilization is as smooth as possible over the entire planning horizon. Under resource levelling, no resource limits are typically imposed. Therefore, only the time lags between individual activities form the project constraints are necessary.
- Finally, if we reverse the scope of this dissertation, we could transfuse solution techniques and concepts from project scheduling literature to process scheduling.

Hopefully, this work will motivate further research in developing better and/or improved modelling frameworks for the standard RCPSP.

Publications

This is a list of the works carried out so far within the scope of interest of this thesis, in reversed chronological order. The list has been divided in papers submitted to international refereed journals and published in conference proceedings:

Scientific Journals

Manuscript published

[1] Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C., 2012. MILP formulations for single- and multi-mode resource-constrained project scheduling problems, Computers & Chemical Engineering 36, 369–385. DOI: http://dx.doi.org/10.1016/B978-0-444-53711-9.50176-0

Manuscripts submitted

[1] Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C., 2012. Mathematical Formulation for Resource-Constrained Project Scheduling Problems with Generalized Precedence Relations, OR Spectrum (September 2012, under review).

[2] Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C., 2012. New Continuous-time and Discrete-time Mathematical Formulations for Resource-constrained Project Scheduling Problems, European Journal of Operational Research (January 2012, 1st revision completed).

Book Chapter

[1] Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C., MILP Formulation for Resource-Constrained Project Scheduling Problems, In E.N.

Pistikopoulos, M. C. Georgiadis, & A. C. Kokossis (Eds.), Computer Aided Chemical Engineering, Vol. 29: 880-884. Amsterdam: Elsevier.

DOI: http://dx.doi.org/10.1016/j.compchemeng.2011.06.007

Refereed Conference Proceeding Articles

The work carried out in this dissertation has been also presented and published in different international specialised refereed conferences. The RCPSP/max formulation proposed in Chapter 5 is very recent and has not been submitted to any conference yet. A list of publications in conferences proceedings follows:

[1] Kyriakidis, T.S., Georgiadis, M.C., Solving resource-constrained project scheduling problems with new mathematical programming formulations, 25th European Conference on Operational Research (EURO 2012), 8-11 July 2012, Vilnius, Lithuania.

[2] Kopanos, G.M., Kyriakidis, T.S., Georgiadis, M.C., New Mathematical Programming Formulations for Resource-Constrained Project Scheduling Problems, 9th International Conference on Computational Management Science (CMS 2012), 18-20 April 2012, London, United Kingdom.

[3] Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C., MILP Formulation for Resource-Constrained Project Scheduling Problems, 21st European Symposium on Computer Aided Process Engineering (ESCAPE-21), 29 May-01 June 2011, Porto Carras, Chalkidiki, Greece.

References

Ahuja, R., Magnanti, T., Orlin, J., 1993. Network Flows. Prentice Hall, Englewood Cliffs.

Achuthan, N.R., Hardjawidjaja, A., 2005. Project scheduling under time dependent costs - A branch and bound algorithm. Annals of Operations Research 108 (1-4), 55-74.

Agrawal, M.K., Elmaghraby, S.E., Herroelen, W.S., 1996. DAGEN: A generator of testsets for project activity nets. European Journal of Operational Research 90. 376–82.

Alcaraz, J., Maroto, C., Ruiz, R., 2004. Improving the performance of genetic algorithms for the RCPS problem. Proceedings of the Ninth International Workshop on Project Management and Scheduling, Nancy, 40–3.

Al-Fawzana, M.A., Haouari, M., 2005. A bi-objective model for robust resource-constrained project scheduling. International Journal of Production Economics 96, 175–87.

Alcaraz, J., Maroto, C., Ruiz, R., 2003. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. Journal of the Operational Research Society 54 (6), 614–26.

Alvarez-Valdes, R., Tamarit, J.M., 1993. The project scheduling polyhedron: Dimension, facets and lifting theorems. European Journal of Operational Research 67 (2), 204-20.

Alvarez-Valdes, R., Tamarit, J.M., 1989. Heuristic algorithms for resource-constrained project scheduling: a review and an empirical analysis. In: Slowinski, R., Węglarz, J., editors. Advances in project scheduling, p. 113–34. Elsevier, Amsterdam.

Artigues, C., Michelon, P., Reusser, S., 2003. Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research 149 (2), 249-67.

Ballestin, F., Barrios, A., Valls, V., 2011. An evolutionary algorithm for the resource-constrained project scheduling problem with minimum and maximum time lags. Journal of Scheduling 14, 391-406.

Bard, J.F., Kontoravdis, G., Yu, G., 2002. A branch-and-cut procedure for the vehicle routing problem with time windows. Transportation Science 36, 250–69.

Bartusch, M., Möhring, R.H., Radermacher, F.J., 1988. Scheduling project networks with resource constraints and time windows. Annals of Operations Research 16, 201–40.

Bianco, L., Caramia, M., 2012a. A new formulation for the project scheduling problem under limited resources. Flexible Services and Manufacturing Journal 2012. published online.

Bianco, L., Caramia, M., 2012b. An exact algorithm to minimize the makespan in project scheduling with scarce resources and generalized precedence relations. European Journal of Operational Research 219, 73-85.

Blazewicz, J., Lenstra, J. K., Rinnooy Kan, A. H. G., 1983. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5 (1), 11-24.

Bomsdorf, F., Derigs, U., 2008. A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. OR Spectrum 30 (4), 751–72.

Böttcher, J., Drexl, A., Kolisch, R., 1996. A Branch-and-Bound Procedure for Project Scheduling with Partially Renewable Resource Constraints. Proceedings of the Fifth Workshop on Project Management and Scheduling, Poznan, April 11-13, 48-51.

Böttcher, J., Drexl, A., Kolisch, R., Salewski, F., 1999. Project scheduling under partially renewable resource constraints. Management Science 45(4), 543–59.

Browning, T.R., Yassine, A.A., 2007. A Random Generator of Resource-constrained Multi-project Scheduling Problems, Technical Report, Texas Christian University, M.J. Neeley School of Business.

Brucker, P., Drexl, A., Mohring, R., Neumann, K., Pesch, E., 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112 (1), 3-41.

Castillo, E., Conejo, A.J., Pedregal, P., García, R., Alguaci, N., 2001. Building and Solving Mathematical Programming Models in Engineering and Science. New York, USA: John Wiley & Sons.

Castro, P., Barbosa-Póvoa, A.P., Matos, H.A., 2001. An Improved Continuous-Time Formulation for the Short-term Scheduling of Multipurpose Batch Plants. Industrial and Engineering Chemistry Research 40, 2059-68.

Castro, P.M., Barbosa-Póvoa, A.P., Matos, H.A., Novais, A.Q., 2004. Simple Continuous-Time Formulation for Short-Term Scheduling of Batch and Continuous Processes. Industrial and Engineering Chemistry Research 43, 105-18.

Cesta, A., Oddi, A., Smith, S., 2002. A constraint based method for project scheduling with time windows. Journal of Heuristics 8(1), 109-36.

Christofides, N., Alvarez-Valdes, R., Tamarit, J. M., 1987. Project scheduling with resource constraints: a branch-and-bound approach. European Journal of Operational Research 29 (3), 262-73.

Cicirello, V.A., Smith, S.F., 2004. Heuristic selection for stochastic search optimization: Modeling solution quality by extreme value theory. In: Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming. CP 2004, Toronto, Canada, 197-211.

Coelho, J., Vanhoucke, M., 2011. Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. European Journal of Operational Research 213, 73–82.

Colvin, M., Maravelias, C.T., 2008. A stochastic programming approach for clinical trial planning in new drug development. Computers and Chemical Engineering 32, 2626-42.

Colvin, M., Maravelias, C.T., 2009. Scheduling of testing tasks and resource planning in new product development using stochastic programming. Computers and Chemical Engineering 33 (5), 964-76.

Colvin, M., Maravelias, C.T., 2010. Modeling methods and a branch and cut algorithm for pharmaceutical clinical trial planning using stochastic programming. European Journal of Operational Research 203, 205-15.

De Reyck, B., Herroelen, W.S., 1998. A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations. European Journal of Operational Research 111 (1), 152–74.

Debels, D., De Reyck, B., Leus, R., Vanhoucke, M., 2006. A hybrid scatter search/Electromagnetism meta-heuristic for project scheduling. European Journal of Operational Research 169 (2), 638-53.

Debels, D., Vanhoucke, M., 2008. A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem. Operations Research 55 (3), 457–69.

Demassey, S., Artigues, C., Michelon, P., 2005. Constraint propagation based cutting planes: an application to the resource-constrained project scheduling problem. INFORMS Journal on Computing 17 (1), 52-65.

Demeulemeester, E.L., 1995. Minimizing Resource-Availability Costs in Time-Limited Project Networks, Management Science 41, 1590-8.

Demeulemeester, E.L., Herroelen, W.S., 2002. Project Scheduling: A Research Handbook. Kluwer Academic Publishers, Boston.

Demeulemeester, E.L., Herroelen, W.S., 1997a. New benchmark results for the resource-constrained project scheduling problem. Management Science 43 (11), 1485–92.

Demeulemeester, E.L., Herroelen, W.S., 1997b. A branch-and-bound procedure for the generalized resource-constrained project scheduling problem. Operations Research 45, 201–12.

Demeulemeester, E.L., Herroelen, W.S., 1992. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management Science 38 (12), 1803–18.

Demeulemeester, E.L., Vanhoucke, M., Herroelen, W.S., 2003. RanGen: A random network generator for activity-on-the-node networks. Journal of Scheduling 6 (1), 17-38.

Dorndorf, U., Pesch, E., Phan-Huy, T., 2000. A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. Management Science 46, 1365-84.

Drexl, A., 1997. Local Search Methods for Project Scheduling under Partially Renewable Resource Constraints, Paper presented at the INFORMS San Diego Spring Meeting, May 4-7, 1997.

Eisner, H., 1962. A generalized network approach to the planning and scheduling of a research project. Operations Research 10 (1), 115-125.

Elmaghraby, S.E., Kamburowski J., 1992. The Analysis of Activity Networks under Generalized Precedence Relations (GPRs). Management Science 38 (9), 1245-63.

Elmaghraby, S.E., 1977. Activity Networks – Project Planning and Control by Network Models, John Wiley & Sons Inc., New York.

Ferris, M., 1999. MATLAB and GAMS: Interfacing optimization and visualization software. Technical report, University of Wisconsin.

Fest, A., Möhring, R.H., Stork, F., Uetz, M., 1998. Resource-constrained project scheduling with time windows: a branching scheme based on dynamic release dates. Technical Report 596, Technical University of Berlin. (Revised in 1999).

Franck, B., Schwindt, C., 1995. Different Resource-Constrained Project Scheduling Models with Minimal and Maximal Time-Lags, Technical Report WIOR-450, Universität Karlsruhe, Germany.

Franck, B., Neumann, K., 1998. Resource Constrained Project Scheduling Problems with Time Windows - Structural Questions and Priority-Rule Methods. Technical Report WIOR-492, University of Karlsruhe, Germany.

Franck, B., Neumann, K., Schwindt, C., 2001. Truncated branch and bound, schedule construction, and schedule improvement procedures for resource constrained project scheduling. OR Spectrum 23, 297–324.

GAMS Development Corporation (2007). GAMS/Cplex 11 User Notes.

Glover, F., 1975. Improved linear integer programming formulations of nonlinear integer problems. Management Science 22 (4), 455–60.

Gomory, R.E., 1960. An algorithm for the mixed integer problem. Technical Report RM-2597, The Rand Corporation.

Gomory, R.E., 1958. Outline of an Algorithm for Integer Solutions to Linear Programs. Bulletin Of the American Mathematical Society 64, 275–8.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimisation and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5 (2), 287-326.

Hartmann, S., 2002. A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics 49, 433–48.

Hartmann, S., 2001. Project scheduling with multiple modes: a genetic algorithm. Annals of Operations Research 102 (1–4), 111–35.

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

Hartmann, S., Drexl, A., 1998. Project scheduling with multiple modes: a comparison of exact algorithms. Networks 32, 283–97.

Hartmann, S., Kolisch, R., 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research 127 (2), 394-407.

Herroelen, W.S., 2005. Project Scheduling - Theory and Practice. Production and Operations Management 14 (4), 413-32.

Herroelen, W.S., De Reyck, B., Demeulemeester, E..L. 1998. A Classification Scheme for Project Scheduling, In Węglarz, J. (Ed.), Handbook of Recent Advances in Project Scheduling, Kluwer Academic Publishers, 1-26.

International Organization for Standardization, 2003. ISO 10006: Guideline for Quality in Project Management. ISO Press, Geneva.

Jain, V., Grossmann, I.E., 1999. Resource-constrained scheduling of tests in new product development. Industrial and Engineering Chemistry Research 38, 3013-26.

Jarboui, B., Damak, N., Siarry, P., Rebai, A., 2008. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Applied Mathematics and Computation 195, 299–308.

Kaplan, L.A., 1988. Resource-constrained project scheduling with preemption of jobs. Ph.D. thesis, University of Michigan.

Karmarkar, N., 1984. A New Polynomial-Time Algorithm for Linear Programming. Combinatorica 4, 373-95.

Kelley, Jr, J. E., Walker, M. R., 1959. Critical-path planning and scheduling. In: Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference. IRE-AIEE-ACM '59 (Eastern). ACM, New York, NY, USA, pp. 160-73.

Khachiyan, L. G., 1979. A polynomial algorithm in linear programming, Dokl. Akad. Nauk SSSR 244, no. 5, 1093{1096. MR 522052 (80g:90071).

Kimms, A., 1998. Optimization Guided Lower and Upper Bounds for the Resource Investment Problem, Research Report 481, Universitat Kiel, Germany.

Klein, R., 2000. Scheduling of resource-constrained projects. Kluwer Academic Publishers, Boston.

Kochetov, Y., Stolyar, A., 2003. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. Proceedings of the 3rd International Workshop of Computer Science and Information Technologies, Russia.

Kolisch, R., 2000. Integrated scheduling, assembly area- and part-assignment for large-scale, make-to-order assemblies. International Journal of Production Economics 64 (1-3), 127-42.

Kolisch, R., 1996a. Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. European Journal of Operational Research 90 (2), 320-33.

Kolisch, R., 1996b. Efficient Priority Rules for the Resource-Constrained Project Scheduling Problem. Journal of Operations Management 14, 179-192.

Kolisch, R., 1995. Project Scheduling under Resource Constraints – Efficient Heuristics for Several Problem Classes, Physica-Verlag, Heidelberg.

Kolisch, R., Hartmann, S., 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174 (1), 23-37.

Kolisch, R., Padman, R., 2001. An integrated survey of deterministic project scheduling. Omega 29, 249-72.

Kolisch, R., Sprecher, A., 1996. PSPLIB - a project scheduling problem library. European Journal of Operational Research 96 (1), 205-16.

Kolisch, R., Schwindt, C., Sprecher A., 1998. Benchmark Instances for Project Scheduling Problems. In J. Węglarz, editor, Handbook on Recent Advances in Project Scheduling. Kluwer.

Kolisch, R., Sprecher, A., Drexl, A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 41 (10), 1693-703.

Kone, O., Artigues, C., Lopez, P., Mongeau, M., 2011. Event-based MILP models for resource-constrained project scheduling problems. Computers & Operations Research 38 (1), 3-13.

Kopanos, G.M., Mendez, C.A., Puigjaner, L., 2010. MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. European Journal of Operational Research 207 (2), 644-55.

Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C., 2012a. Solving Resource-Constrained Project Scheduling Problems through new Mathematical Programming Formulations. Proceedings of the European Conference on Operational Research (EURO 2012), Vilnius, Lithuania, 240.

Kyriakidis, T. S., Kopanos, G. M., Georgiadis, M. C., 2012b. MILP formulations for single- and multi-mode resource-constrained project scheduling problems. Computers & Chemical Engineering 36, 369-85.

Lancaster, J., Ozbayrak, M., 2007. Evolutionary algorithms applied to project scheduling problems - a survey of the state-of-the-art. International Journal of Production Research 45 (2), 425–50.

Land, A.H., Doig, A. G., 1960. An automatic method for solving discrete programming problems. Econometrica 28, 497–520.

Levis, A.A., Papageorgiou, L.G., 2004. A hierarchical solution approach for multi-site capacity planning under uncertainty in the pharmaceutical industry. Computers and Chemical Engineering 28, 707-25.

Malcolm, D.G., Roseboom, J. H., Clark, C.E, Fazar W., 1959. Application of a technique for research and development program evaluation. Operations Research 7(5), 646-69.

Maravelias, C.T., Grossmann, I.E., 2004. Optimal resource investment and scheduling of tests for new product development. Computers and Chemical Engineering 28, 1021-38.

Marchetti, P.A., Cerda, J., 2009. A general resource-constrained scheduling framework for multistage batch facilities with sequence-dependent changeovers. Computers & Chemical Engineering 33 (4), 871-86.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L., 1998. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Management Science 44 (5), 715-29.

Minieka, E., 1978. Optimization algorithms for Networks and Graphs. Marcel Dekker Inc., New York.

Mitchell, J.E., 2001. Branch-and-cut algorithms for integer programming. In the Encyclopedia of Optimization, Volume II, pages 519-25, Kluwer Academic Publishers.

Möhring, R.H., 1984. Minimizing Costs of Resource Requirements in Project Networks Subject to a Fixed Completion Time, Operations Research 32, 89-120.

Möhring, R.H., Schulz, A., Stork, F., Uetz, M., 2003. Solving project scheduling problems by minimum cut computations. Management Science 49 (3), 330-50.

Nabrzyski, J., Węglarz, J., 1994. A Knowledge-Based Multiobjective Project Scheduling System. Revue des Systemes de Decision, 3, 185-200.

Nemhauser, G. L., Wolsey, L. A., 1999. Integer and Combinatorial Optimization. Discrete Mathematics and Optimization. Wiley - Interscience, New York.

Neumann, K., Schwindt, C., 1995. Projects with Minimal and Maximal Time Lags: Construction of Activity-on-Node Networks and Applications. Technical Report WIOR-447. Institüt fur Wirtschaftstheorie und Operations Research, University of Karlsruhe.

Neumann, K., Schwindt, C., Zimmermann, J., 2003. Project Scheduling with Time Windows and Scarce Resources. Springer, Berlin.

Neumann, K., Schwindt, C., Zimmermann, J., 2002. Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions. Springer, Berlin.

Neumann, K., Zimmermann, J., 2000. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. European Journal of Operational Research 127, 425-43.

Neumann, K., Zimmermann, J., 1999. Heuristic methods for resource-constrained project scheduling with regular and nonregular objective functions and schedule-dependent time windows. In: Węglarz J (ed) Project Scheduling: Recent Models, Algorithms, and Applications. Kluwer Academic Publishers, Boston, pp 261-87.

Pantelides, C.C., 1994. Unified frameworks for optimal process planning and scheduling. In: Proceedings of the Second Conference on Foundations of Computer-Aided Process Operations. CACHE, pp. 253-74.

Papageorgiou, L.G., Rotstein, G.E., Shah, N., 2001. Strategic supply chain optimization for the pharmaceutical industries. Industrial and Engineering Chemistry Research 40, 275-86.

Paraskevopoulos, D.C., Tarantilis, C.D., Ioannou, G., 2012. Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. Expert Systems with Applications 39, 3983–94.

Patterson, J.H., 1976. Project scheduling: The effects of problem structure on heuristic scheduling. Naval Research Logistics 23 (1), 95-123.

Pinto, J.M., Grossmann, I.E., 1995. A continuous time MILP model for short term scheduling of batch plants with pre-ordering constraints. Industrial & Engineering Chemistry Research 34 (9), 3037–51.

Pritsker, A.A.B., Watters, L.J., Wolfe, P.M., 1969. Multiproject scheduling with limited resources: a zero-one programming approach. Management Science 16 (1), 93-108.

Rieck, J., Zimmermann, J., Gather, T., 2012. Mixed-integer linear programming for resource leveling problems. European Journal Operational Research 221, 27-37.

Rosenthal, R.E., 2012. GAMS - A User's Guide. GAMS Development Corporation, Washington, Washington, DC, USA.

Russell, J.R.M., 1970. Cash Cows in networks Management Science 16 (5), 357-73.

Sabzehparvar, M., Seyed-Hosseini, S.M., 2008. A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. Journal of Supercomputing 44 (3), 257-73.

Salewski, F., Schirmer, A., Drexl, A., 1997. Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application. European Journal Operational Research 102, 88-110.

Schirmer, A., Drexl, A., 1996. Partially Renewable Resources – A Generalisation of Resource-Constrained Project Scheduling, Paper presented at the IFORS Triennial Meeting, Vancouver, B.C., July 8-12.

Schilling, G., Pantelides, C.C., 1996. A simple continuous-time process scheduling formulation and a novel solution algorithm. Computers and Chemical Engineering 20, S1221-S1226.

Schmidt, C.W., Grossmann, I.E., 1996. Optimization models for the scheduling of testing tasks in new product development. Industrial and Engineering Chemistry Research 35, 3498-510.

Schutt, A., Feydy, T., Stuckey, P. J., & Wallace, M. G., 2009. Why cumulative decomposition is not as bad as it sounds. In I. P. Gent (Ed.), LNCS 5732. Proceedings of principles and practice of constraint programming—CP 2009, pp. 746–761), Springer, Berlin.

Schutt, A., Feydy, T., Stuckey, P. J., & Wallace, M. G., 2012. Solving RCPSP/max by lazy clause generation. Journal of Scheduling 2012.

Schwindt, C., 1996. Generation of resource-constrained scheduling problems with minimal and maximal time Lags. Technical Report WIOR-489. Institüt fur Wirtschaftstheorie und Operations Research, University Karlsruhe, Germany.

Schwindt, C., 1998a. A branch-and-bound algorithm for the resource-constrained project duration problem subject to minimum and maximum time lags. Technical Report WIOR-544. Institüt fur Wirtschaftstheorie und Operations Research, University of Karlsruhe, Germany.

Schwindt, C., 1998b. Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungs problems mit planungsabhängigen Zeitfenstern. Shaker Verlag, Aachen, Germany.

Slowinski, R., 1980. Two approaches to problems of resource allocation among project activities - a comparative study. The Journal of the Operational Research Society 31 (8), 711-23.

Smith, T., 2004. Windows-based project scheduling algorithms. Dissertation, University of Oregon.

Sprecher, A., Hartmann, S., Drexl, A., 1997. An exact algorithm for project scheduling with multiple modes. OR Spektrum 19 (3), 195–203.

Stinson, J.P., Davis, E.W., Khumawala, B.M., 1978. Multiple resource-constrained scheduling using branch and bound. AIIE Transactions 10, 252–9.

Talbot, F.B., 1982. Resource-constrained project scheduling with time-resource tradeoffs: the nonpreemptive case. Management Science 28 (10), 1197-1210.

Tavares, L. V., Ferreira, J. A., Coelho, J. S., 1999. The risk of delay of a project in terms of the morphology of its network. European Journal of Operational Research 119 (2), 510-37.

The Mathworks, Inc., 1998. MATLAB: The Language of Technical Computing: MATLAB Notebook User's Guide. Natick, MA, USA: The Mathworks Inc.

Valls, V., Ballestin, F., Quintanilla, M.S., 2003. A hybrid genetic algorithm for the RCPSP. Technical report, Department of Statistics and Operations Research, University of Valencia.

Valls, V., Ballestin, F., Quintanilla, M.S., 2005. Justification and RCPSP: A technique that pays. European Journal of Operational Research 165, 375-86.

Van Peteghem, V., Vanhoucke, M., 2010. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. European Journal of Operational Research 201, 409–18.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., Tavares, L. V., 2008. An evaluation of the adequacy of project network generators with systematically sampled networks. European Journal of Operational Research 187 (2), 511-24.

Węglarz, J., 1979. Project scheduling with discrete and continuous resources. IEEE Transactions on Systems, Man and Cybernetics 9 (10), 644–50.

Węglarz, J., Jozefowska, J., Mika, M., Waligora, G., 2011. Project scheduling with finite or infinite number of activity processing modes - A survey. European Journal of Operational Research 208, 177–205.

Yang, X.S., 2008. Introduction to Computational Mathematics. World Scientific Publishing, Singapore.

Zapata, J.C., Hodge, B.M., Reklaitis, G.V., 2008. The Multimode Resource Constrained Multiproject Scheduling Problem: Alternative Formulations. AICHE 54(8), 2101-19.

Zhu, G., Bard, J.F., Ju, G., 2006. A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. INFORMS Journal on Computing 18 (3), 377-90.

Zimmermann, J., 1997. Heuristics for Resource-Leveling Problems in Project Scheduling with Minimum and Maximum Time-Lags, Technical Report WIOR-491, Universität Karlsruhe, Germany.