Computers and Chemical Engineering 36 (2012) 369–385

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

MILP formulations for single- and multi-mode resource-constrained project scheduling problems

Thomas S. Kyriakidis[a], Georgios M. Kopanos[b], Michael C. Georgiadis[a,c,*]

[a] Department of Engineering Informatics & Telecommunications, University of Western Macedonia, Karamanli and Lygeris Street, Kozani 50100, Greece
[b] Department of Chemical Engineering, Universitat Politècnica de Catalunya, ETSEIB, Av. Diagonal 647, Barcelona 08028, Spain
[c] Department of Chemical Engineering, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece

ARTICLE INFO

Article history:
Received 25 January 2011
Received in revised form 25 May 2011
Accepted 6 June 2011
Available online 15 June 2011

Keywords:
Project scheduling
Single/multi-mode resource-constrained project scheduling
Mixed-integer linear programming
Resource-Task Network
Project management

ABSTRACT

This work 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 modeling 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 and different types of resources. Finally, the applicability of the proposed formulations is illustrated using several example problems under the most commonly addressed objective, the makespan minimization.

1. Introduction

Project scheduling involves the construction of a *precedence* and *resource* feasible time schedule which identifies the starting and completion 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. However, there may exist alternatives for some, which vary in aspects such as duration, cost and required set of resources. For instance, only one of two activities must be performed to enable the start of their successor (e.g. the activity "heating food" must be preceded by one of the activities "oven on" or "microwave on"). Since, resources are usually limited and impose restrictions to project scheduling, they must be included in the project description. Logical links and precedence constraints for activities must also be incorporated.

Most *resource-constrained project scheduling problems* (RCPSPs) assume a single execution mode per activity, with specific time and resource requirements. Other problems include activities with various possible execution modes. Such problems are called *multi-mode resource-constrained project scheduling problems* (MRCPSPs), where the mode determines the duration of the activity and the resource requirements.

Slowinski (1980) categorized 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 at 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, Drexl, and Kolisch (1996) limit the utilisation 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

* Corresponding author at: Department of Engineering Informatics & Telecommunications, University of Western Macedonia, Karamanli and Lygeris Street, Kozani 50100, Greece. Tel.: +30 2461056523.
E-mail addresses: mgeorg@otenet.gr, mgeorg@uowm.gr (M.C. Georgiadis).

0098-1354/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2011.06.007

Nomenclature

Indices/sets

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

Subsets

I_{ALT_X}	xth 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
$\bar{S}_{ri}^L/\bar{S}_{ri}^U$	lower/upper bounds on the amount of resource r produced at the end of activity i
$\alpha_{imr}/\bar{a}_{imr}$	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
$\bar{y}_{it}$	1 if activity i is active over both t and $t-1$
z_{mi}	1 if activity i is executed in mode m

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)
$S_{rit}/\bar{S}_{rit}$	linearized surplus terms
τ_t	duration of time slot t
τlin_{it}	linearized duration term

working time is limited by their contract. It has been shown by Böttcher, Drexl, Kolisch, and Salewski (1999) that both renewable and non-renewable resource categories can be depicted by partially renewable resources.

Two representations have been commonly used to capture project networks, the *Activity-on-Arc* (AoA), which is event-based, and the *Activity-on-Node* (AoN), which is activity based. An *Activity-on-Arc* (AoA) diagram is based on the idea that each activity is a transition between two events; its beginning 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 activities leading into it and the beginning of all activities pointing out. *Activity-on-Node* (AoN), also known as precedence diagramming method, is a network representation for activity sequencing. 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 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).

Blazewicz, Lenstra, and Rinnooy Kan (1983) have shown that the RCPSP is an NP-hard problem. When the solution has to additionally determine the assignment of modes (MRCPSP), further complexity is added since the solution search space is enlarged. Due to the high degree of complexity of RCPSPs and MRCPSPs, numerous heuristic and metaheuristic methods for them have been proposed in the literature. Although, heuristics may produce good solutions with few computational requirements and in a reasonable computation time, they ignore whether the solution can be proven to be correct. Contrary to exact algorithms which always produce an optimal solution. As this work focuses on exact mixed-integer linear programming (MILP) formulations for the RCPSP and MRCPSP, we refer to Hartmann and Kolisch (2000) and Kolisch and Hartmann (2006) for an elaborate study on state-of-the-art heuristic and metaheuristic methods. In our review we will examine exact state-of-the-art solution methods.

According to Herroelen (2005) computational results indicate that many of the 60-activity and most of the 90- and 120-activity instances from PSPLIB (Kolisch & Sprecher, 1996) are still beyond the solution capabilities of exact methods. Even recent papers using MILP models, such as Koné, Artigues, Lopez, and Mongeau (2011) deal with up to 25–35 activities. Pritsker, Watters, and Wolfe (1969) proposed a discrete-time mathematical formulation for multi-project scheduling which can also be used for the single-project case. 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, Alvarez-Valdés, and Tamarit (1987) developed a formulation similar to Pritsker et al., named CAT. The two formulations mainly differ in how they formulate the precedence constraints with the precedence constraints formulated by Christofides being disaggregated expressions of Pritsker's. The reported computational results are on randomly generated problems involving up to 25 activities and 3 resources. Two formulations with an exponential number of variables are those of Alvarez-Valdés and Tamarit (1993) and Mingozzi, Maniezzo, Ricciardelli, and Bianco (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. The second one by Mingozzi et al. (1998), is a *discrete-time* formulation based on *feasible subsets*, that is activities that can be simultaneously executed without violating resource or precedence constraints. 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 multistage 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 trials outcomes while taking into account the trading structure of the company. Recently, Koné et al. (2011) introduced two new MILP formulations: the *Start/End Event*-based (SEE) formulation (a variant of the event-based formulation proposed in Zapata, Hodge, & Reklaitis, 2008), and the *On/Off Event*-based (OOE) formulation. The variables in such formulations are fewer than in time-indexed ones, since they are not a function of the time horizon. They compared the event-based formulations with three other formulations issued from the literature, two of which (Christofides et al., 1987; Pritsker et al., 1969) use time-indexed variables, and a third formulation (Artigues, Michelon, & Reusser, 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.

Talbot (1982) proposed a model for multiple resource constraints where activities have *multiple modes* (MRCPSP). It 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. Zhu, Bard, and Ju (2006) proposed a branch-and-cut method based on Christofides et al. formulation (1987) for the MRCPSP, which gave very competitive results on benchmark problems. Sabzehparvar and Seyed-Hosseini (2008) presented a continuous-time formulation for the MRCPSP with Generalized Precedence Relations (MRCPSP-GPR) in which the minimal or maximal time lags between a pair of activities may vary depending on the chosen modes. 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. Each model handles the time domain in a different way.

Project scheduling problems have been studied under various types of objective functions, such as *maximizing the Net Present Value* (NPV), introduced by Russell (1970), m*inimizing resource availability costs* (Demeulemeester, 1995; Kimms, 1998), and multi-objective scheduling (Bomsdorf & Derigs, 2008; Nabrzyski & Weglarz, 1994). The most popular objective function discussed in the project scheduling literature is undoubtedly the time-based objective of *minimizing the project makespan*. Most often it is recognized as the most relevant objective in various review papers (Hartmann & Briskorn, 2010; Herroelen, 2005; Kolisch, 1996; Kolisch & Sprecher, 1996).

The main scope of this work is to extend and therefore apply modeling and network representation techniques used in the process industry to RCPSP and MRCPSP problems. The manuscript is organized as follows. In Section 2, a new representation of project scheduling problems based on the RTN concept is presented. In Sections 3 and 4, we propose new MILP formulations for RCPSPs and MRCPSPs, including precedence relations, different types of resources, time, and other constraints. Afterwards, the applicability of the proposed formulations to several project scheduling problems is demonstrated in Section 5. Finally, concluding remarks are drawn in Section 6.

2. A new network representation for the RCPSP

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 set of resources. Since resources impose restrictions to project scheduling, they must be included in the general project description. Logical links and precedence constraints for activities must also be incorporated.

2.1. Definition of a project

Traditional 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 generalized 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 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 characterized 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.

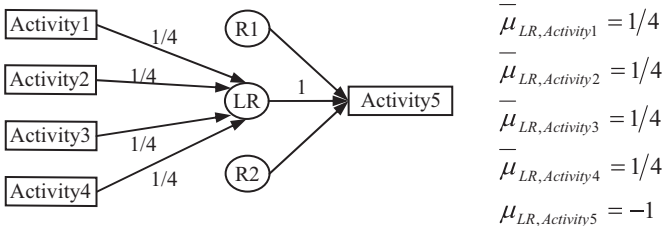


Fig. 1. Example of the use of logical resources.

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

2.2. Conversion of general projects to RTN form

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. 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.

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 modeled as Resources, with specific values on their following parameters:

1. the activity consumption and production coefficients (μ_{ri} and $\bar{\mu}_{ri}$) 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.

2.2.1. Precedence constraints

Projects consist of coordinated and controlled activities with start and finish times. To achieve the required execution precedence, precedence constraints on each activity are imposed. These constraints can be modeled indirectly 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 ($\bar{\mu}_{ri}$) for the preceding activities and the consumption coefficient (μ_{ri}) for the activity with the assigned logical resource, as shown in Fig. 1.

2.2.2. General resource management

Resource limitations can extend the project execution time. 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 us consider the example in Fig. 2, 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 were present, Cleaning and Settling could have been performed at the same time, making the Blender available for use sooner.

2.2.3. 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 modeled 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.

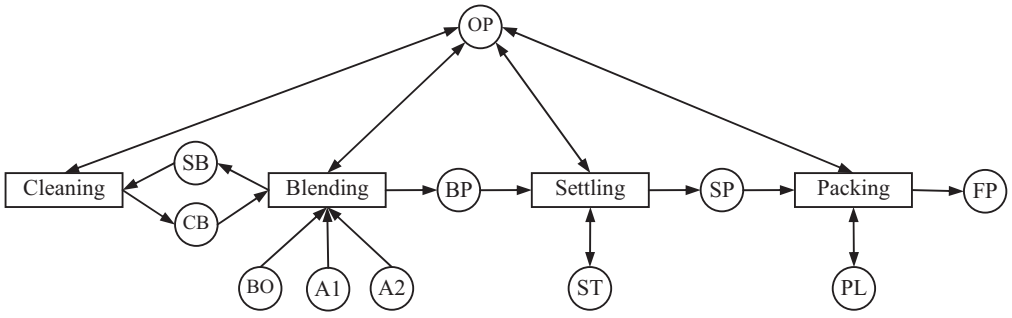


Fig. 2. Graphic representation of a resource task network.

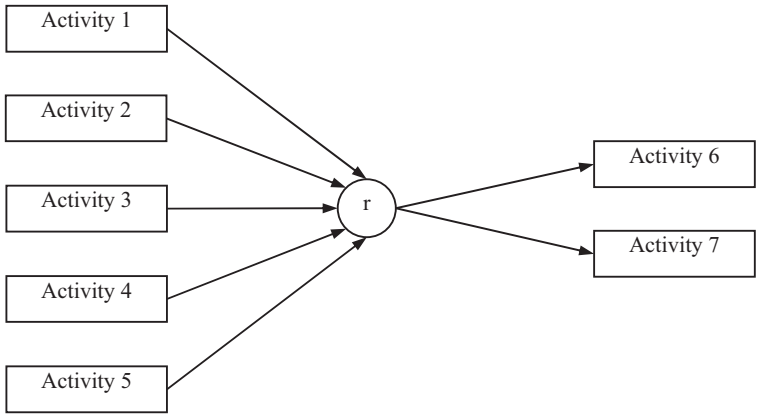


Fig. 3. Production of logical resource.

On the other hand, the maximum excess resource $R_r^{\max}$ and the activity consumption and production coefficients (μ_{ri} and $\bar{\mu}_{ri}$) 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 $\bar{\mu}_{ri} = 1/x, \forall i \in I_r^{in}$. 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 $\bar{\mu}_{ri} = 1/x, \forall i \in I_r^{in}$, 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 Fig. 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 $\bar{\mu}_{ri} = 1/3, \forall i \in I_r^{in}$. 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 $\bar{\mu}_{ri} = 1/3, \forall i \in I_r^{in}$, 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.

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 Fig. 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 Fig. 4. The production of each of these resources is similar to what was described previously.

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 Figs. 5 and 6.

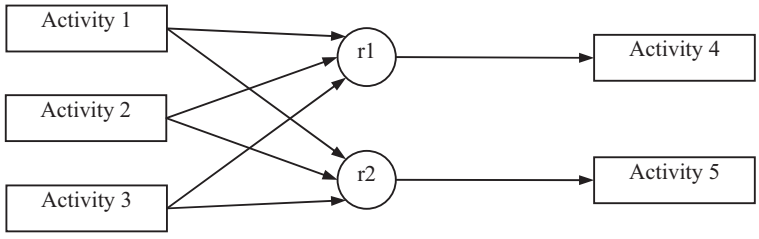


Fig. 4. Production of logical resources for several output activities.

Table 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	Fig. 5
At least x, $1 \le x \le IN_r$	Exactly 1	$1/x$	-1	IN_r/x	Fig. 5
Exactly 1	At least 1	1	$-1/OUT_r$	1	Fig. 5
All	1 Immediate	$1/IN_r$	-1	$1-(1/IN_r)$	Fig. 5
Exactly x, $1 < x \le IN_r$	At least 1	$1/x$	-1	1	Fig. 6
At least x, $1 \le x \le IN_r$	At least 1	$1/x$	-1	IN_r/x	Fig. 6

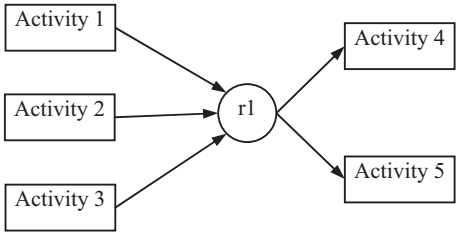


Fig. 5. Conditions Exactly *x*/At least *x*–Exactly one and Exactly one–At least one.

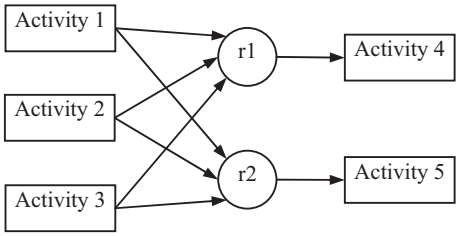


Fig. 6. Conditions Exactly *x*/At least *x*–At least one.

A decision box with *output* condition *Exactly 1*, is represented using one only one new logical resource. This case corresponds to Fig. 5.

A decision box with *input* condition *Exactly 1*, can also be represented as depicted in Fig. 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.

For the other cases we define OUT_r logical resources, as in Fig. 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 *All*/*Exactly 1*). It can be achieved by not allowing the excess resource of r to reach values greater or equal to 1 ($R_r^{\max} = 1 - (1/IN_r)$). 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 1.

There are two special subcases that require additional mechanisms to enforce the required logical dependencies. The first such case occurs when $x < IN_r$. 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 Fig. 7.

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 Fig. 8, with logic At least 2–Exactly 1.

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 easily be overcome by making each output activity $i \in I_r^{out}$ produce a unit amount of a new resource r' (i.e. $\bar{\mu}_{r'i} = 1, \forall i \in I_r^{out}$) 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.

2.3. 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 modeled 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.

Fig. 9 depicts an example of the simple case where the project end requires the execution of a single final activity.

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

$$R_{PE}^{initial} = 0$$
$$R_{PE}^{\min} = 0, \quad R_{PE}^{\max} = 1$$
$$R_{PE,Final}^{\min} = R_{PE,Final}^{\max} = 1$$
$$\bar{\mu}_{PE,D} = 1$$

2.3.1. 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 (an example with 4 activities is illustrated in Fig. 10).

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

$$\bar{\mu}_{PE,i} = \frac{1}{NF}, \quad \forall i \in F$$

By demanding the minimum and maximum final excess resources of PE to be 1, all input activities are enforced to be

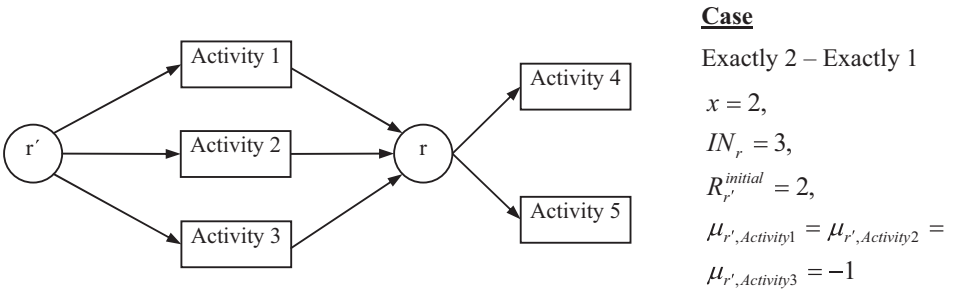


Fig. 7. Special case 1 example Exactly x with $x < IN_r$.

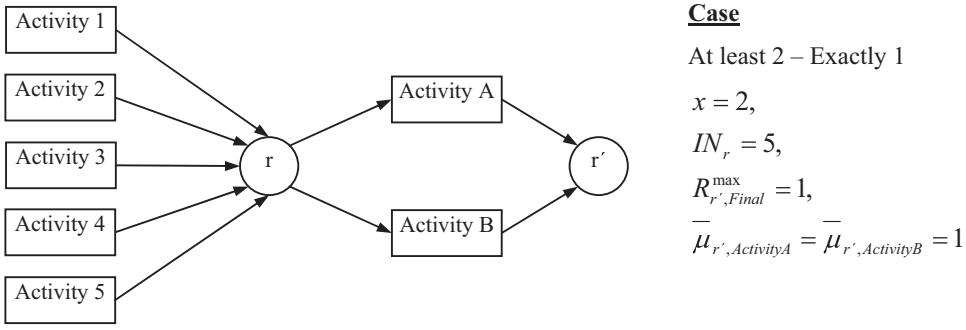


Fig. 8. Special case 2 example At least x–Exactly 1 with $IN_r > 2x$.

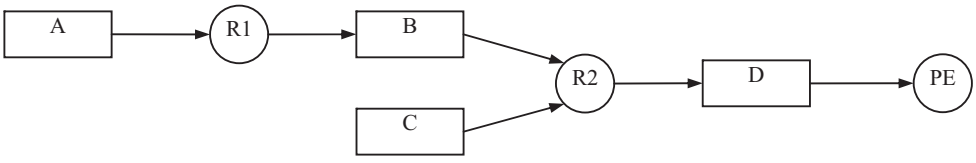


Fig. 9. An example of an RTN representation of a project.

executed. The rest of data values are the same as the case with one activity:

$$R_{PE}^{initial} = 0$$
$$R_{PE}^{\min} = 0$$
$$R_{PE}^{\max} = 1$$
$$R_{PE,Final}^{\min} = 1$$
$$R_{PE,Final}^{\max} = 1$$
$$\bar{\mu}_{PE,i} = \frac{1}{NF}, \quad \forall i \in F$$

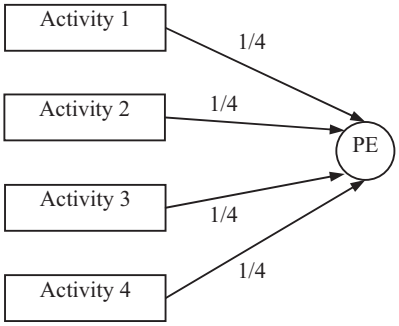


Fig. 10. Example of project end with 4 activities.

2.3.2. Project end with alternative final activities

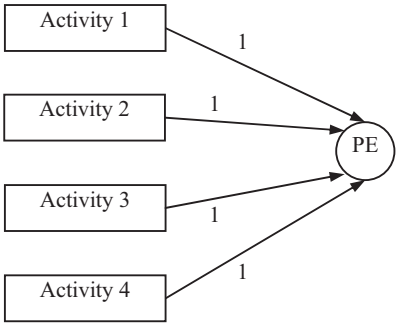
Having examined the case of multiple required final activities, we move on to another one. It is possible to have alternatives among final activities, meaning that project completion can be achieved by performing only one of them. This is equivalent to the input condition *Exactly 1* in a decision box.

Therefore, a logical final resource PE, with no output conditions is introduced. Each alternate input activity is set to produce $\bar{\mu}_{PE,i} = 1$ quantity of PE, and the maximum final excess resource is set to 1, $R_{PE,Final}^{\max} = 1$. 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 $R_{PE,Final}^{\min} = 1$. For an illustrative example see Fig. 11.

2.3.3. 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 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 modeled as a circle, is the condition for project completion. An example of such a tree is shown in Fig. 12.



Case

4 alternate activities

$$R^{\max}_{PE,Final} = 1,$$
$$R^{\min}_{PE,Final} = 1$$
$$\bar{\mu}_{PE,Activity1} = \bar{\mu}_{PE,Activity2} = \bar{\mu}_{PE,Activity3} = \bar{\mu}_{PE,Activity4} = 1$$

Fig. 11. Example of project end with 4 alternate activities.

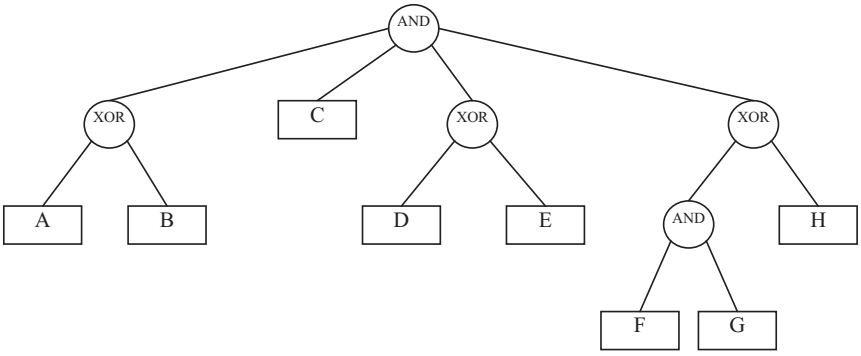


Fig. 12. Example of logical tree.

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

- complete either A or 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 Fig. 12 is depicted in Fig. 13.

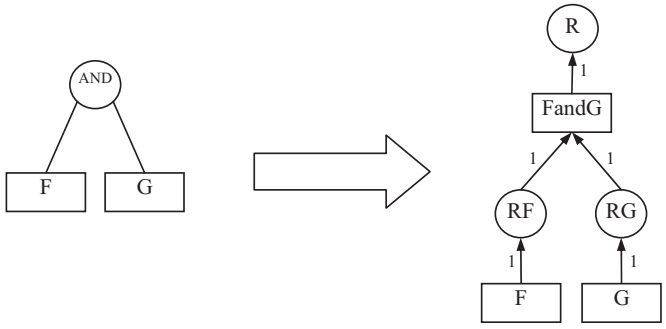


Fig. 13. Transformation of conjunctions to RTN.

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

- For activity F:

$$\tilde{\mu}_{RF,F} = 1$$
$$R^{initial}_{RF} = 0$$
$$R^{\min}_{RF} = 0$$
$$R^{\max}_{RF} = 1$$

- For activity G:

$$\tilde{\mu}_{RG,G} = 1$$
$$R^{initial}_{RG} = 0$$
$$R^{\min}_{RG} = 0$$
$$R^{\max}_{RG} = 1$$

- For activity FandG:

$$\mu_{RF,FandG} = -1$$
$$\mu_{RG,FandG} = -1$$
$$\tilde{\mu}_{R,FandG} = 1$$
$$R^{initial}_{R} = 0$$
$$R^{\min}_{R,Final} = 1$$
$$R^{\max}_{R,Final} = 1$$

Disjunctions are much easier to represent, as they are, essentially, a decision box with an *Exactly 1* input logic. Fig. 14 illustrates an example of such a disjunction of two activities.

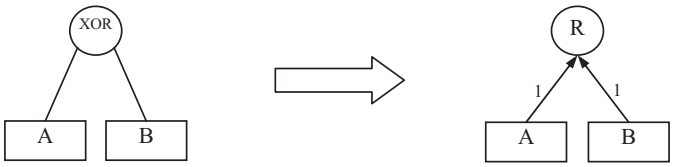


Fig. 14. Transformation of disjunctions to RTN.

The data set for this transformation is:

$\bar{\mu}_{A,R} = 1$
$\bar{\mu}_{B,R} = 1$
$R_R^{initial} = 0$
$R_R^{\min} = 0$
$R_R^{\max} = 1$
$R_{R,Final}^{\min} = 1$
$R_{R,Final}^{\max} = 1$

Using these transformations we can convert the logical tree of Fig. 12, to its equivalent RTN of Fig. 15, with all arrows corresponding to unit consumption/production of resource.

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 synchronized 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 Pantelides (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, Barbosa-Póvoa, and Matos (2001), that relaxes the non-linear timing constraint producing an easier to solve, pure MILP problem and also the work of Castro, Barbosa-Póvoa, Matos, and Novais (2004), where a new set of timing constraints further reduces the computational cost. The main objective of the manuscript is to establish a new modeling framework for RCPSPs representations 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 Pantelides (1996).

We set the number of time slots T equal to the number of activities, in case the worst scenario is realized, with only one activity executed at each slot. We assume that no more than one instance of an activity can be executed at any time point. This assumption converts the N_{it} variable, representing the number of 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, $\upsilon_{ri} = \bar{\upsilon}_{ri} = 0$.

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.

3.1. Constraints

Schilling and Pantelides (1996) distinguish four types of constraints: timing, slot, excess resource balances and excess resource capacity constraints. We will modify these constraints for 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 binary variables, y_{it}, replaces the RTN variable 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 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) results in a simpler mathematical model with far fewer binary variables than those required by the original RTN formulation (Schilling & Pantelides, 1996).

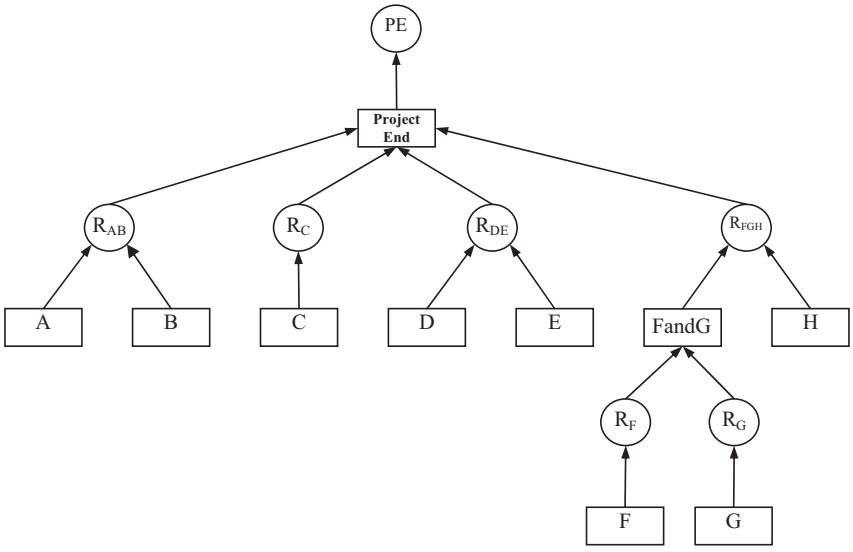


Fig. 15. Equivalent RTN for logical tree in Fig. 12.

Table 2
Combination of values for $\bar{y}_{i,t-1} + y_{i,t-1} - \bar{y}_{it}$..

$\bar{y}_{i,t-1}$	$y_{i,t-1}$	$\bar{y}_{it}$	$\bar{y}_{i,t-1} + y_{i,t-1} - \bar{y}_{it}$	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

3.1.1. 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 & Pantelides, 1996) are transformed to:

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

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 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 $\tau lin_{it} \equiv (y_{it} + \bar{y}_{it})\tau_t$. This definition can be effected using the following linear constraints:

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

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

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:

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

3.1.2. 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:

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

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 in Table 2.

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

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

3.1.3. 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:

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

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.

3.1.4. 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:

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

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

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

3.1.5. Objective function

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

Minimize H

Table 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

3.2. Improvement to the formulation

The proposed mathematical model although it is simple and compact, can be improved to achieve better computational performance. Considering the various aspects of project scheduling, we can set tighter bounds to variables and reduce the number of elements for sets that participate in constraints.

3.2.1. 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 Fig. 16. Activity C is preceded by A and cannot be executed before A finishes. This sets a lower slot bound to C's starting time slot to 2.

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

$$tl_i = 1 + PA_i, \quad \forall i$$
$$tu_i = T - SA_i, \quad \forall i$$

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 Fig. 16 is $T = 5$. Table 3 contains the appropriate lower and upper bounds for this case.

This improvement reduces the number of y_{it}, $\bar{y}_{it}$ 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* $\times T$. This number of variables ($6 \times 5 = 30$ for this example) could be reduced to 14, as given in Table 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* $\times (T-1) = 6 \times 4 = 24$, 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 $\sum_i (tu_i - tl_i) = 8$.

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

3.2.2. 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:

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

Exploiting, constraint (11), we can also transform (2) to:

$$\sum_{t=1}^{T} (y_{it} + \bar{y}_{it})\tau_t = \theta_i, \quad \forall i \in I_{sure} \tag{12}$$

3.2.3. Alternative activities

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

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

This results in a reduction of $(N_{ALT_x} - 1) \times T$ for the constraint, with N_{ALT_x} denoting the number of activities in set ALT_x. In the given example, this is a reduction of $(N_{ALT_x} - 1) \times T = (2 - 1) \times 5 = 5$ with only one alternative and its two members D and E.

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

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 modeled using decision boxes with multiple inputs, exactly one of which is to be executed. An example of how activities with multiple modes are modeled is shown below, in Fig. 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 ($R_{LRA}^{initial} = 1$).

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) and (3)–(13) and the objective function represents the total project duration.

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 (Demeulemeester & Herroelen, 2002).

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.

A number of new parameters and variables are introduced, to account for the special features of the MRCPSP:

Parameters

S^L_{ri}/S^U_{ri}	represent the minimum/maximum quantities of resource r required by the various modes m at the beginning of activity i
$\bar{S}^L_{ri}/\bar{S}^U_{ri}$	are the minimum/maximum quantities of resource r produced by the various modes m 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.

Variables

z_{mi}	1 if activity i is executed in mode m, 0 otherwise.
S_{ri}	the surplus of resource r consumed at the beginning of activity i (above basic consumption level S^L_{ri}),
$\bar{S}_{ri}$	the surplus of resource r produced at the end of activity i (above basic production level $\bar{S}^L_{ri}$).

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 $t = 0$.

4.1.1. 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:

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

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 3.1 We define new variables $\tau lin_{it} \equiv (y_{it} + \bar{y}_{it})\tau_t$, through the linear constraints:

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

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

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 $\theta^{\max}_{mi}$. After applying this linearization technique, constraint (15) becomes linear:

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

4.1.2. Slot constraints

As in the RCPSP formulation, 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. These constraints can be combined to the following inequality:

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

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:

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

and if so, only one activity mode is performed:

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

4.1.3. Excess resource balances

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

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

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

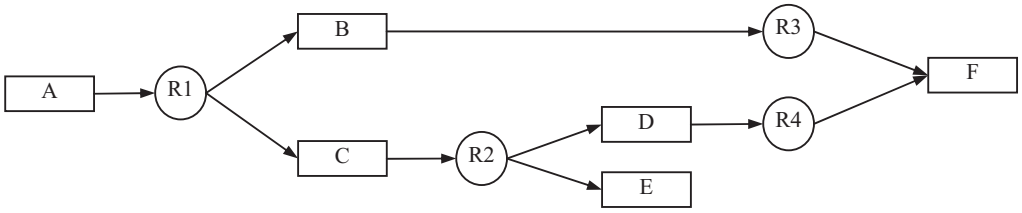
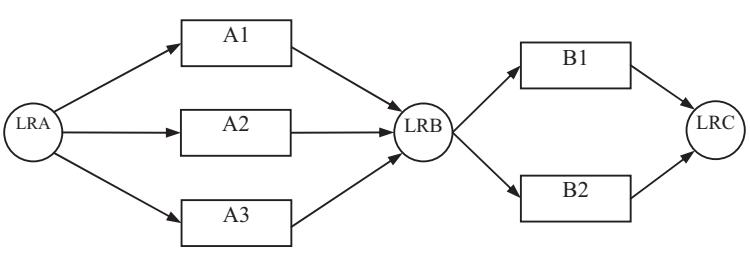


Fig. 16. Slot boundaries example.



$$R^{initial}_{LRA} = 1,$$
$$\mu_{LRA,A1} = \mu_{LRA,A2} = \mu_{LRA,A3} = -1$$
$$\bar{\mu}_{LRB,A1} = \bar{\mu}_{LRB,A2} = \bar{\mu}_{LRB,A3} = 1$$

Fig. 17. Modeling activities with multiple modes.

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

$$S^L_{ri} = \min(a_{imr}) \quad \text{and} \quad S^U_{ri} = \max(a_{imr})$$

Similarly, the values of the minimum/maximum quantities of resource r produced by the various modes m at the end of activity i, $\bar{S}^L_{ri}/\bar{S}^U_{ri}$ are given by:

$$\bar{S}^L_{ri} = \min(\bar{a}_{imr}) \quad \text{and} \quad \bar{S}^U_{ri} = \max(\bar{a}_{imr})$$

Surplus variables S_{ri} and $\bar{S}_{ri}$ depend on the selected mode m, are bounded by constraints (30) and (31) and calculated through (32) and (33), as presented in Section 4.1.5.

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

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

The first variable S_{rit} is defined:

$$0 \le S_{rit} \le (S^U_{ri} - S^L_{ri})y_{it}, \quad \forall t, (ri) \tag{23}$$

$$\sum_t S_{rit} = S_{ri}, \quad \forall (ri) \tag{24}$$

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

$$0 \le \bar{S}_{rit} \le (\bar{S}^U_{ri} - \bar{S}^L_{ri})(\bar{y}_{i,t-1} + y_{i,t-1} - \bar{y}_{it}), \quad \forall t, (ri) \tag{25}$$

$$\sum_t \bar{S}_{rit} = \bar{S}_{ri}, \quad \forall (ri) \tag{26}$$

Now we can replace (22) with:

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

4.1.4. 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:

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

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

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

4.1.5. Task operation constraints

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

$$0 \le S_{ri} \le S^U_{ri} - S^L_{ri}, \quad \forall (r, i) \tag{30}$$

$$0 \le \bar{S}_{ri} \le \bar{S}^U_{ri} - \bar{S}^L_{ri}, \quad \forall (r, i) \tag{31}$$

For logical resources S^L_{ri} and $\bar{S}^L_{ri}$ are equal to 0.

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

$$\sum_m (a_{imr} - S^L_{ri}) z_{mi} = S_{ri}, \quad \forall r, i \tag{32}$$

$$\sum_m (\bar{a}_{imr} - \bar{S}^L_{ri}) z_{mi} = \bar{S}_{ri}, \quad \forall r, i \tag{33}$$

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.

4.1.6. Objective function

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

4.2. Improvement to the formulation

The extended formulation proposed in this section can achieve better computational performance, using the improvement techniques in Section 3.2. To account for the new variables introduced, we need to clarify a few points, regarding each constraint.

4.2.1. 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}, $\bar{y}_{it}$, τ_{it}, S_{rit}, and $\bar{S}_{rit}$ variables and related constraints.

4.2.2. 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:

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

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

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

4.2.3. Alternative activities

An MRCPSP can contain sets I_{ALT_X} 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:

$$\sum_{I_{ALT_X}} \sum_t y_{it} \le 1, \quad \forall i \tag{13}$$

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

5. Computational results

In this section, we consider a typical MRCPSP with all typical modeling 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:

Table 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	–

- **SMRTN** – the formulation used for RCPSPs, presented in Section 3,
- **MMRTN1** – the RCPSP formulation with the modifications in Section 3.3 used for MRCPSPs, and
- **MMRTN2** – the formulation used for MRCPSPs, presented in Section 4.

All formulations were solved on an Intel Core 2 Quad Q8300 @ 2.5 GHz and 4GB RAM using CPLEX 11.1.1 (GAMS Development Corporation, 2007) via a GAMS 22.8.1 (Rosenthal, 2008) WIN 6007.6015 VIS interface.

5.1. Example problem

In this section, we consider a typical MRCPSP from PSPLIB (Kolisch & Sprecher, 1996). The test instance used is j10 2_2 with 10 activities, each having 3 possible execution modes. Table 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.

Activity durations and resource requests for each mode are provided in Table 5. The maximum project duration is 86, and we calculate it by summing the maximum mode duration per activity.

This project uses 2 renewable (R1 and R2) and 2 non-renewable resources (N1 and N2) and their availabilities are displayed in Table 6. For the renewable resources, the availabilities are per period.

The activities network and coefficients using our representation is shown in Fig. 18. The resources L2–L12 are logical resources.

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 modeled using decision boxes with multiple inputs, exactly one of which is to be executed, as in Fig. 19.

Table 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

Table 6
Resource availabilities.

R1	R2	N1	N2
9	4	29	40

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

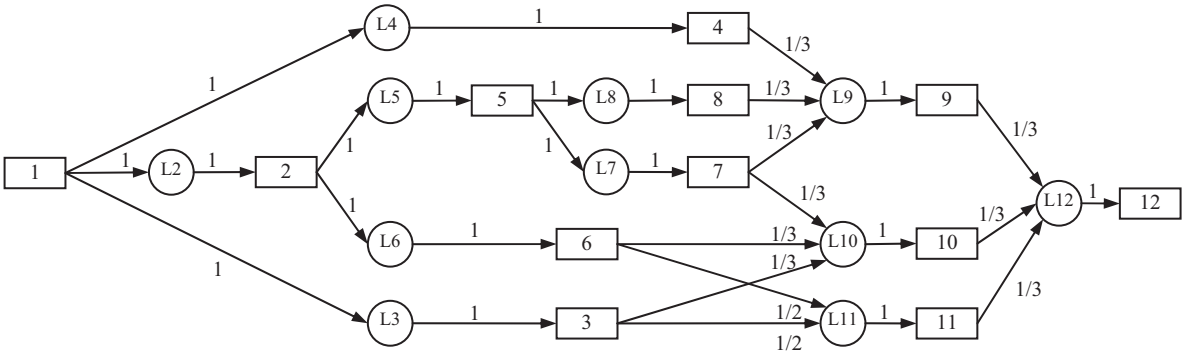
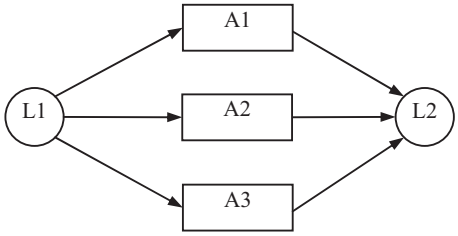


Fig. 18. Activity network for example problem.



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

Fig. 19. Modeling activities with multiple modes.

Table 7
S_{ri}^{L}, $\bar{S}_{ri}^{L}$, S_{ri}^{U} and $\bar{S}_{ri}^{U}$ parameter values.

S_{ri}^{L}

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

S_{ri}^{U}

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

$\bar{S}_{ri}^{L}$

Activity	Resource	
	R1	R2
1	–	–
2	–	–
3	–	–
4	6	–
5	–	–
6	–	–
7	–	–
8	3	–
9	1	–
10	–	–
11	–	1
12	–	–

$\bar{S}_{ri}^{U}$

Activity	Resource	
	R1	R2
1	–	–
2	6	6
3	7	4
4	10	–
5	2	9
6	2	8
7	5	7
8	6	–
9	2	–
10	4	2
11	–	2
12	–	–

completion, activity A produces 1 unit of L2 ($\bar{\mu}_{L2,A1} = \bar{\mu}_{L2,A2} = \bar{\mu}_{L2,A3} = 1$).

The S_{ri}^{L}, $\bar{S}_{ri}^{L}$, S_{ri}^{U} and $\bar{S}_{ri}^{U}$ parameter values when using the MMRTN2 formulation, are shown in Table 7.

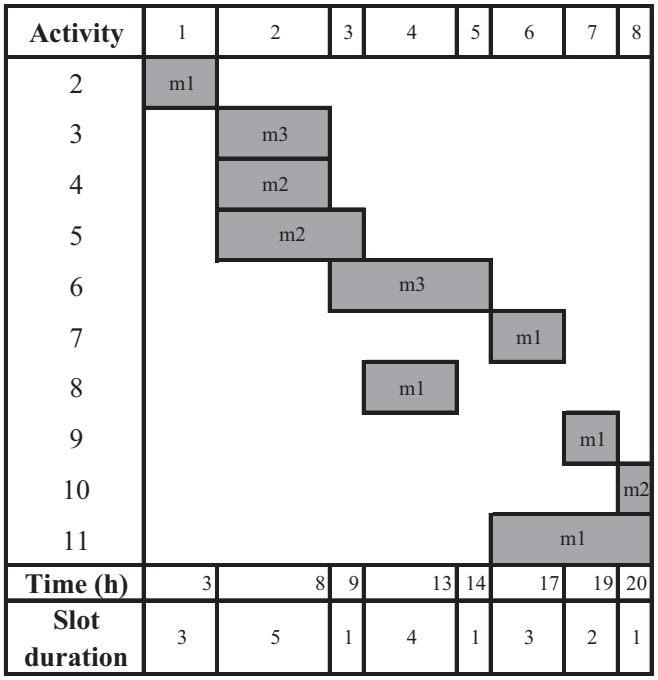


Fig. 20. GANTT chart of optimal solution for the example problem.

The MMRTN1 formulation obtained the optimal makespan (20 time units) in 239.12 CPU s (see Table 9) and the optimal project schedule is shown in Fig. 20. Activities 1 and 12 are not included in the GANTT chart since they are dummy and have zero duration.

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, utilising up to 4

Table 8
Computational results for various single-mode RCPSP test instances.

Activities	Renewable resources	Resource complexity	CPU time	Number of equations	Binary variables	Continuous 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

Table 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	59,423	0	20	20
	MMRTN2	1000+	4555	225	3890	118,685	5.97	20	
j12 2_8	MMRTN1	146.14	2205	864	738	16,925	0	49	49
	MMRTN2	1000+	7023	318	6104	147,486	12.24	49	
j14 1_8	MMRTN1	1000+	2967	1176	972	33,931	31.63	34	34
	MMRTN2	1000+	10247	427	9027	51,586	2.94	34	
c15 4_3	MMRTN1	1000+	3935	1536	1238	14,420	44.44	36	34
	MMRTN2	1000+	14345	553	12746	11,137	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	

renewable resources. The computational results are presented in Table 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.

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, and 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 are displayed in Table 9. The optimal solutions reported in Table 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.

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 less defined binary variables y_{it} and $\bar{y}_{it}$ for less tasks. Meanwhile, the number of continuous variables and constraints are 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. However, the proposed formulations seem promising for the efficient modeling of specific complex project scheduling problems such as construction projects.

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

Acknowledgment

Georgios Kopanos would like to acknowledge financial support from the Spanish Ministry of Education (FPU Grant to GMK).

References

Alvarez-Valdés, R., & Tamarit, J. M. (1993). The project scheduling polyhedron: Dimension, facets and lifting theorems. *European Journal of Operational Research*, *67*(2), 204–220.

Artigues, C., Michelon, P., & Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, *149*(2), 249–267.

Blazewicz, J., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, *5*, 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), 751–772.

Böttcher, J., Drexl, A., & Kolisch, R. (1996). Proceedings of the fifth workshop on project management and scheduling. *A Branch-and-bound procedure for project scheduling with partially renewable resource constraints*, 48–51.

Böttcher, J., Drexl, A., Kolisch, R., & Salewski, F. (1999). Project scheduling under partially renewable resource constraints. *Management Science*, *45*(4), 543–559.

Castro, P., Barbosa-Póvoa, A. P. F. D., & Matos, H. (2001). An improved continuous-time formulation for the short-term scheduling of multipurpose batch plants. *Industrial and Engineering Chemistry Research*, *40*, 2059–2068.

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*, 105–118.

Christofides, N., Alvarez-Valdés, R., & Tamarit, J. M. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, *29*, 262–273.

Colvin, M., & Maravelias, C. T. (2008). A stochastic programming approach for clinical trial planning in new drug development. *Computers and Chemical Engineering*, *32*, 2626–2642.

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), 964–976.

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*, 205–215.

Demeulemeester, E. L. (1995). Minimizing resource-availability costs in time-limited project networks. *Management Science*, *41*, 1590–1598.

Demeulemeester, E. L., & Herroelen, W. (2002). *Project scheduling—A research handbook. Volume 49 of International series in Operations research & management science*. Boston, MA: Kluwer Academic Publishers.

Eisner, H. (1962). A generalized network approach to the planning and scheduling of a research project. *Operations Research*, *10*(1), 115–125.

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), 455–460.

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *207*, 1–14.

Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *127*, 307–394.

Herroelen, W. S. (2005). Project scheduling—Theory and practice. *Production and Operations Management*, *14*(4), 413–432.

Jain, V., & Grossmann, I. E. (1999). Resource-constrained scheduling of tests in new product development. *Industrial and Engineering Chemistry Research*, *38*, 3013–3026.

Kimms, A. (1998). *Optimization guided lower and upper bounds for the resource investment problem*. Research Report 481. Universitat Kiel, Germany.

Kolisch, R. (1996). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, *14*, 179–192.

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of Heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, *174*, 23–37.

Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega*, *29*, 249–272.

Kolisch, R., & Sprecher, A. (1996). PSPLIB—A project scheduling problem library. *European Journal of Operational Research*, *96*, 205–216.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers and Operation Research*, *38*(1), 3–13.

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*, 707–725.

Maravelias, C. T., & Grossmann, I. E. (2004). Optimal resource investment and scheduling of tests for new product development. *Computers and Chemical Engineering*, *28*, 1021–1038.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1998). An exact algorithm for the multiple resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, *44*(5), 714–729.

Minieka, E. (1978). *Optimization algorithms for networks and graphs*. New York: Marcel Dekker Inc.

Nabrzyski, J., & Weglarz, J. (1994). A knowledge-based multiobjective project scheduling system. *Revue des Systemes de Decision*, *3*, 185–200.

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; Computer Aided Chemical Engineering (pp. 253–274). CACHE Publications.

Papageorgiou, L. G., Rotstein, G. E., & Shah, N. (2001). Strategic supply chain optimization for the pharmaceutical industries. *Industrial and Engineering Chemistry Research*, *40*, 275–286.

Pritsker, A., Watters, L., & Wolfe, P. (1969). Multi-project scheduling with limited resources: A zero-one programming approach. *Management Science*, *16*, 93–108.

Rosenthal, R. E. (2008). *GAMS-A user's guide*. Washington, DC: GAMS Development Corporation.

Russell. (1970). Cash cows in networks. *Management Science*, *16*(5), 357–373.

Sabzehparvar, M., & Seyed-Hosseini, S. M. (2008). A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent timelags. *Journal of Supercomputing*, *44*(3), 257–273.

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*, 3498–3510.

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), 711–723.

Talbot, F. B. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science*, *28*(10), 1197–1210.

Zapata, J. C., Hodge, B. M., & Reklaitis, G. V. (2008). The multimode resource constrained multiproject scheduling problem: Alternative formulations. *AIChE*, *54*(8), 2101–2119.

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), 377–390.