An Exact Algorithm for the Mode Identity Project Scheduling Problem

Behrouz Afshar Nadjafi[a,*], Amir Rahimi[b], Hamid Karimi[b]

[a] *Assistant Professor, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[b] *MSc, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

Received 2 July, 2011; Revised 19 December, 2011; Accepted 18 March, 2012

Abstract

In this paper we consider the non-preemptive variant of a multi-mode resource constrained project scheduling problem (MRCPSP) with mode identity, in which a set of project activities is partitioned into disjoint subsets while all activities forming one subset have to be processed in the same mode. We present a depth-first branch and bound algorithm for the resource constrained project scheduling problem with mode identity. The proposed algorithm is extended with some bounding rules to reduce the size of branch and bound tree. Finally, some test problems are solved and their computational results are reported.

Key Words: Project Scheduling, Branch and Bound, Mode-Identity, Multi-Mode, Resource Constrained.

1. Introduction

Project scheduling with constrained resources is a central field within operations research and management sciences. A project consists of activities, subject to precedence relations and constrained resources, which have a predetermined objective. In the classical resource-constrained project scheduling problems the project duration (or make span) is to be minimized. The multi-mode problem (MRCPSP) is a generalized version of the standard problem (RCPSP), where each activity can be executed in one of several modes representing a relation between resource requirements of the activity and its duration. The schedule has to be precedence and resource feasible and no activity should be interrupted. A project can be represented by an activity-on-node (AoN) or an activity-on-arc (AOA) network. In the first representation nodes correspond to activities and arcs to precedence constraints whereas in the second one, nodes correspond to time events and arcs to activities. The resources can be renewable, non-renewable, doubly constrained, and/or partially renewable, where the renewable resources are limited period-by-period, the non-renewable resources are limited for the entire project, the doubly constrained ones are limited for both each period and the entire project, and the availability of the partially renewable resources is defined for a specific time interval (a subset of periods). However, under discrete resources, the doubly constrained resources do not need to be taken into account explicitly since they can be incorporated by properly enlarging the sets of the first two types of resources. The objective is to find an assignment of modes to activities as well as precedence- and resource-Feasible starting times of all activities such that the make span of the project is minimized. The problem is strongly NP-hard being a generalization of the RCPSP. The RCPSP is strongly NP-hard as a generalization of the well-known job shop problem (Błazewicz, 1983). Moreover, for more than one non-renewable resource the problem of finding a feasible solution of the MRCPSP is already NP-complete (Kolisch, 1995).

The MRCPSP has been widely studied in the recent years. Several exact, meta-heuristics and heuristic approaches as well as some extensions have been developed. Some exact approaches are proposed by Talbot (1982), Speranza and Vercellis (1993), and Zhu et al. (2006). Almost all the approaches, except the one proposed by Zhu et al. (2006), are based on the B&B method and the idea to enumerate partial schedules.

An extension of the MRCPSP to its version with generalized precedence relations, denoted as the MRCPSP-GPR, is studied in several publications. Exact approaches based on the B&B method are proposed in (De Reyck and Herroelen, 1998; Dornorf, 2002; Heilmann, 2003). Some heuristic algorithms are developed by De Reyck and Herroelen (1999) who used a hybrid of tabu search and a truncated version of their B&B, by Heilmann (2001) who proposed a multi-pass priority rule approach with back planning which is based on an integration approach and embedded in random sampling, and by Calhoun et al. (2002) who implemented the tabu search. Moreover, Van Hove and Deckro (1998) proposed a B&B approach for the MRCPSP

* Corresponding author E-mail address: afsharnb@alum.sharif.edu

with minimal time lags only. Recently, Barrios et al. (2009) proposed for the MRCPSPGRP a so-called double genetic algorithm which outperforms other approaches in medium and large instances.

An extended version of the MRCPSP with the so-called mode identity constraints is considered by Salewski et al. (1997). The resulting problem is called the mode identity resource-constrained project scheduling problem (MIRCPSP), and is motivated by real-world situations where several activities should be performed in the same way, i.e. by allocating them the same resources. Practical examples of such a problem occur in audit staff scheduling, timetabling, course scheduling, etc. Formally, in this problem the set of all project activities is partitioned into several disjoint subsets, and all activities belonging to the same subset have to be performed by the same resources. The time and cost of executing activities from such a subset depend on the resources assigned. Moreover, for each activity a deadline, a ready time, and a set of mode-dependent finish to start time lags with direct predecessors are defined. A mathematical model of the problem is formulated, and the NP-hardness in the strong sense is proved. A two-phase heuristic is used to find a good feasible schedule. In phase one, for each subset of activities a mode is selected randomly. In phase two, a solution is built by scheduling randomly chosen activities from the eligible set.

The standard multi-mode resource constrained project scheduling problem involves the selection of an execution mode for each activity (mode assignment) and the determination of the activity start or finish times such that the precedence and resource constraints are met and the project duration is minimized. In the multi-mode case, all mode-activity assignments are mutually independent; i.e. assigning a mode to one activity i of a project consisting of a set of n nonpreemptable activities does not necessarily force any other activities to be processed in a specific mode. In practice, however, there may be situations in which certain activities belong together and must be executed in the same mode (Drexl et al., (1998)).

In the multi-mode case, some solution procedures have been developed (Coelho and Vanhoucke, 2011 and Kyriakidis et al., 2012).

Salewski et al. (1997) partitioned the set of all activities into disjoint subsets where all the activities forming one subset have to be performed by the same resources. The time and cost incurred by processing such a subset depend on the resources assigned to it. Salewski et al. (1997) refered to the resulting problem as the mode-identity problem, in which objective is to minimize the cost of processing. They prove that the mode identity problem is strongly NP-hard. This model is suitable for timetabling, course scheduling, audit staff scheduling and other assignments-type scheduling problems.

The literature on solution methods for the mode identity problem is scant. Salewski et al. (1997) developed a parallel regret-based biased random sampling approach, RAMSES, which consists of two stages. In the first stage, priority values are used to assign modes to subsets of activities. In the second stage, a schedule is built using a priority-based parallel scheduling scheme. This paper addresses the resource constrained project scheduling problem with mode identity, in which we consider a project consisting of activities to be scheduled subject to finish-start precedence relations with zero time lags and renewable resource constraints. The objective is to minimize the project duration.

The paper is organized as follows: In the next section, the problem description is presented and the terminology used is clarified. In Section 3, a branch and bound procedure is described. Following that, the computational results are reported in Section 4. Finally, Section 5 concludes the paper.

2. Problem Description

The mode identity and resource constrained project scheduling problem (MIRCPSP) involves the scheduling of project activities in order to minimize the project makespan. In this problem setting, the set of project activities is partitioned into U disjoint subsets while all activities forming one subset have to be processed in the same mode. The project is represented by an AON network where the set of nodes, N, represents activities and the set of arcs, A, represents finish-start precedence constraints with a time-lag of zero. The non-preemptable activities are numbered from a dummy start activity 1 to the dummy end activity n, and are topologically ordered. According to the classification scheme of Demeulemeester and Herroelen (2002), problem can be classified as $m,1/cpm,disc,id/reg,C_{max}$. We have the notations given in Table 1 for the MIRCPSP.

Table 1
Parameters of the MIRCPSP

Problem parameter	Definition
n	Number of activities indexed by j
K	Number of renewable resources indexed by k
d_{jm}	Time required to perform activity j in mode m
H_u	Specific nonempty subset u of activities
U	Number of disjoint subsets of activities, indexed by u
T	Number of time periods, indexed by t
r_{jmk}	Per-period usage of renewable resource k required to execute activity j in mode m
R_k	Per-period availability of renewable resource k
M_u	Number of modes of subset u, indexed by m
EFT_j	Earliest finish time of activity j
LFT_j	Latest finish time of activity j
f_u	The job with the smallest index of subset H_u
P_j	The set of immediate predecessors of activity j

Defining variables xjmt is as follows:

$$x_{jmt} = \begin{cases} 1 & \text{If activity } j \text{ is performed in mode } m \text{ and} \\ & \text{completed in period } t\,;0 \quad \text{Otherwise} \end{cases} \qquad (1)$$

This allows formulating the mode identity and resource constrained project scheduling problem (MIRCPSP) under the minimum project make span objective as follows: (derived from Salewski et al. 1997's formulation).

$$\min\ Z = \sum_{t=EFT_n}^{LFT_n} t x_{n1t} \tag{2}$$

s.t:

$$\sum_{m=1}^{M_u} \sum_{t=EFT_{f_u}}^{LFT_{f_u}} x_{f_u mt} = 1 \qquad (1 \le u \le U) \tag{3}$$

$$\sum_{t=EFT_{f_u}}^{LFT_{f_u}} x_{f_u mt} = \sum_{t=EFT_j}^{LFT_j} x_{jmt} \tag{4}$$

$$(1 \le u \le U, \forall j \in H_u \setminus \{f_u\}, 1 \le m \le M_u, |H_u| > 1)$$

$$\sum_{m=1}^{M_{u'}} \sum_{t=EFT_i}^{LFT_i} t x_{imt} \le \sum_{m=1}^{M_u} \sum_{t=EFT_j}^{LFT_j} (t - d_{jm}) x_{jmt} \tag{5}$$

$$(1 \le u' \le U, \forall i \in H_{u'}, 1 \le u \le U, \forall j \in H_u, \forall i \in P_j)$$

$$\sum_{u=1}^{U} \sum_{m=1}^{M_u} \sum_{j \in H_u} r_{jmk} \sum_{q=\max\{t, EFT_j\}}^{\min\{t+d_{jm}-1, LFT_j\}} x_{jmq} \le R_k \tag{6}$$

$$1 \le k \le K, 1 \le t \le T) \ x_{jmt} \in \{0,1\} \tag{7}$$

$$(1 \le u \le U, \forall j \in H_u, 1 \le m \le M_u, EFT_j \le t \le LFT_j)$$

The objective in equation 2 minimizes the project duration. It is assumed that the dummy start node and dummy end node can only be processed in a single mode with duration equal to zero. The constraints in equation 3 assure that each activity is assigned exactly one mode and exactly one finish time. The constraint set in equation 4 maintains the mode identity constraints, in which all activities forming one subset have to be performed in the same mode. Equation 5 denotes the precedence relations-constraints. Equation 6 ensures that the per-period availability of the renewable resources is not violated. Finally, equation 7 imposes binary values on the decision variables.

Here, we demonstrate the MIRCPSP with the project network (Fig.1). There are 5 activities (and two dummy activities).

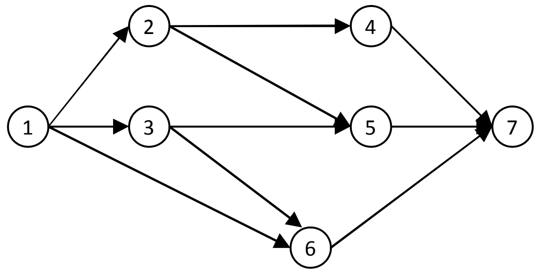

Fig. 1. An example of the network

An example of the mode identity structure for this network is presented in Table 2. The set of activities is partitioned into four disjoint subsets. The second subset, for example, consists of activities 2, 3 and 5, for which two possible modes are specified. All three activities, however, must be executed in the same mode. If mode 2 is selected, the three activities 2, 3 and 5 are executed in the second mode.

Table 2
Partitioning the set of all activities

Disjoint subset number	Activities in the subset	Modes	Selected mode
1	1	1	1
2	2,3,5	1,2	2
3	4,6	1,2,3,4	4
4	7	1	1

3. The Branch and Bound Algorithm (Precedence Tree)

The precedence tree based approach was originated by Patterson et al. (1989) to solve the RCPSP and was further refined by Sprecher (1994) and Sprecher and Drexl (1998) to solve the multi-mode case. Hartman and Drexl (1998) showed that the precedence tree approach by Sprecher and Drexl (1998) outperforms the other available branch and bound algorithms with respect to computation times.

The precedence tree approach is based on the enumeration of all feasible sequences that correspond with different early-start schedules and the selection of the best amongst the feasible sequences.

In this section, we propose a modified structure of the precedence tree algorithm for the MIRCPSP. The procedure is based on the observation that any early-start schedule can be obtained by listing all activities in a sequence such that no successor of an activity is sequenced before its predecessor. Every such sequence corresponds with one early-start schedule by scheduling the different activities as soon as possible in the order of the sequence, but without violating the precedence, resource and mode identity constraints.

3.1. Branching Strategy

The procedure begins with starting the dummy start activity 1 with mode 1 at time 0. At each level *g* of the branch and bound tree, we determine the set SJ_g of the already scheduled activities and the set EJ_g of the eligible activities, that is, those activities whose predecessors are already scheduled. Then we select an eligible activity j_g. If activity j_g has at least one same subset activity scheduled at previous levels, its execution mode is fixed before, so activity j_g has only one allowable mode AM_{j_g}. Otherwise, we select a mode $m_{j_g} \in \{1,...,AM_{j_g}\}$ of this activity. Now we compute the earliest precedence feasible start time EST_{j_g} and the earliest resource feasible start time S_{j_g}, so that $s_{j_g} \geq EST_{j_g}$. Then we branch to the next level. If the dummy end activity *n* is eligible, we have found a complete schedule, and the finish time of activity *n*, FT_n, is the project duration. In this case, backtracking to the previous level occurs. Here we select the next untested mode. If there is no untested mode left, we select the next untested eligible activity. Note that if the selected activity j_g, has at least one same subset activity scheduled at previous levels, it has no untested mode. If we have tested all eligible activities in all allowable modes, we track another step back.

Having discussed all the necessary concepts of the algorithm, we present it with the pseudo-code given in Table 3.

Table 3
Branching Algorithm

```
Step 1: Initialization
  Initialization step sets the level of the precedence tree to 1, $g = 1$.
  Schedule the first (dummy) activity with the start time of zero, $j_1$=1, $m_{j_1} = 1$, $s_{j_1} = 0$
  Initialize the set of the already scheduled activities, $SJ_1 = \Phi$, then go to step 2.
Step 2: Compute the set of eligible activities
  Increase the level of the precedence tree and update the set of already scheduled activities, $g = g+1$; $SJ_g = SJ_{g-1} \cup \{j_{g-1}\}$
  Compute the set of eligible activities (i.e., activities not currently scheduled whose predecessors are
  already scheduled), $EJ_g = \{j \in \{1,...,n\} \setminus SJ_g \mid P_j \subseteq SJ_g\}$
  If the last (dummy) activity is eligible $n \in EJ_g$, then store the current solution and go to Step 5.
  Otherwise go to Step 3.
Step 3: Select the next activity to be scheduled
  If there is no untested activity left in $EJ_g$, then go to Step 5.
  Otherwise select an untested activity, $j_g \in EJ_g$, then go to step 4.
Step 4: Select a mode and compute the activity start time
  If there is no activity same subset $j_g$ in $SJ_g$ then
    If there is no untested mode left in $\{1,...,M_{j_g}\}$, then go to Step 3.
    Otherwise select an untested mode $m_{j_g} \in \{1,...,M_{j_g}\}$.
      Set the mode $m_{j_g}$ allowable for other activities in the same subset $j_g$:
        $AM_j = m_{j_g}; (\forall j \in \{1,...,n\} \setminus j_g, j \cup j_g = H_u, \exists u \in \{1,...,U\})$
      Compute the earliest precedence feasible start time, $EST_{j_g} = \max\{FT_i \mid i \in P_j\}$.
      Compute the earliest resource feasible start time $s_{j_g} \geq EST_{j_g}$, then go to Step 2.
  Otherwise (i.e. there is the activity same subset $j_g$ in $SJ_g$)
    If activity $j_g$ already scheduled at this level g, then go to Step 3.
    Otherwise set allowable mode $AM_j$ for $j_g$: $m_{j_g} = AM_j$.
      Compute the earliest precedence feasible start time, $EST_{j_g} = \max\{FT_i \mid i \in P_j\}$.
      Compute the earliest resource feasible start time $s_{j_g} \geq EST_{j_g}$, then go to Step 2.
Step 5: Backtracking
  Decrease the level of the precedence tree, $g = g - 1$.
  If the precedence level is equal to 1, then STOP.
  Otherwise go to Step 4.
```

3.2. Bounding Rules

If it can be established that further branching from a node cannot lead to an optimal solution, then the node can be pruned away. While most of the rules are known from the literature, we present a new rule (Rule 5) and adapt some well-known ones for the MIRCPSP.

3.2.1. Bounding Rule 1 (Data Reduction)

This bounding rule has originally been proposed by Sprecher et al. (1997). An execution mode m_j is called *non-executable* if we have $r_{jm_jk} > R_k$ for any $k \in K$. Hence, non-executable modes may be excluded from the project data without losing optimality.

3.2.2. Bounding Rule 2

Due to the structure of the precedence tree, the algorithm may enumerate one schedule several times. To avoid duplicate consideration of a schedule, Hartman and Drexl (1998) proposed a bounding rule to exclude duplicate enumeration for the multi-mode RCPSP, which we adapt it for the MIRCPSP. Consider two activities *i* and *j* scheduled on the previous and on the current level of the branch and bound tree, respectively. If we have $s_i = s_j$ and $i > j$, then the current partial schedule does not need to be completed.

3.2.3. Bounding Rule 3

This bounding rule was proposed by Patterson et al. (1989) to avoid duplicate consideration of a schedule in the RCPSP. Here, we adapt it for the MIRCPSP. Consider two activities *i* and *j* scheduled on the previous and on the current level of the branch and bound tree, respectively. If we have $s_j < s_i$, then the current partial schedule does not need to be completed.

3.2.4. Bounding Rule 4

This bounding rule is based on critical path length. Patterson et al. (1989) employed the primal version of this rule in their algorithm to solve the RCPSP. We propose an alternative version for the MIRCPSP.

Consider an upper bound of the makespan of the project which is, for example, given by the sum of the maximal duration of the activities. If the algorithm finds the first or an improved schedule with a makespan *T*, the upper bound will be replaced by *T*. we add the remaining critical path length of the currently scheduled activity to its start time and if this value exceeds or equals the currently best solution, we can dominate the current node in the precedence tree. Note that computing the remaining critical path length is different. In computing the remaining critical path length of the currently scheduled activity *j*, if activity *j* has at least one same subset activity at the current partial schedule, its fixed mode duration needs to be considered. Otherwise, its minimal duration needs to be considered.

3.2.5. Bounding Rule 5

We drive another critical path-based bounding rule for the MIRCPSP. If an eligible activity cannot be feasibly scheduled in any mode in the current partial schedule without exceeding the currently best solution, then no other eligible activity needs to be examined on this level.

We add the remaining critical path length of an eligible activity to its start time in all modes and if these values, lower bounds of the project duration, exceed or equal the currently best solution, then no other eligible activity needs to be examined at the current level. The remaining critical path length is computed as the preceding bounding rule. Moreover, to strengthen this bounding rule, consider activity *i* scheduled on the previous level of the branch and bound tree, and eligible activity *j* on the current level of the branch and bound tree, respectively. If we have $s_i \geq s_j$ and $i > j$, then in computing the lower bound, the start time of activity *j* should be considered at the start time of activity *i* plus one, inspired by bounding rule 2. Another idea is also borrowed from bounding rule 3 in which if we have $s_i > s_j$, then in computing the lower bound, the start time of activity *j* should be considered at the start time of activity *i*.

3.2.6. Bounding Rule 6

This bounding rule was proposed by Hartman and Drexl (1998). The finish time and the start time of a scheduled activity *j* are denoted with f_j and s_j, respectively. We consider two activities *i* and *j* with $i > j$ that $f_i = s_j$. Now, an *order swap* is defined as the interchange of these two activities by assigning new start and finish times $s'_j := s_i$ and $f'_i := f_j$, respectively. Thus, the precedence and resource constraints may not be violated, and the modes and starts times of the other activities may not be changed. A schedule in which no order swap can be performed is called *order monotonous*. Clearly, it is sufficient to enumerate only order monotonous schedules. Assume that no currently unscheduled activity will be started before the finish time of a scheduled activity *j* when the current partial schedule is completed. If an order swap on activity *j* together with any of those activities that finish at its start time can be performed, then the current partial schedule does not need to be completed.

3.2.7. Bounding Rule 7

This bounding rule was developed by Demeulemeester and Herroelen (1992) for the RCPSP and generalized by Sprecher et al. (1997) to the multi-mode case. Here we use it for the MIRCPSP.

Consider an eligible activity j no mode of which is simultaneously performable with any currently unscheduled activity in any mode. If the earliest feasible start time of each other eligible activity in any mode is equal to the maximal finish time of the currently scheduled activities, then j is the only eligible activity that needs to be selected for being scheduled on the current level of the branch and bound tree.

Note that, if activity j has at least one same subset activity at the current partial schedule, its only execution mode is fixed before. Also, if activity j has no same subset activity at the current partial schedule, then activity j and its same subset activities should be executed in the identical mode. This notion is valid for any unscheduled activity too.

4. Computational Results

In this section we present the results of the computational studies concerning the proposed algorithm in the previous section.

4.1. Experimental design

In order to validate the proposed branch and bound method for the MIRCPSP, a problem set consisting of 90 problem instances was generated by the project generator ProGen/πx developed by Drexl et al. (2000), using the parameters given in Table 4.

Table 4
The parameters setting for the problem set

Control Parameter	Value
Number of activities (non-dummy)	20, 25, 30
Number of execution modes	3
Job subset strength (*JSS*)	0.5, 0.6, 0.7
Activity durations	[1,10]
Number of initial activities	3
Number of terminal activities	3
Maximal number of predecessors	3
Maximal number of successors	3
Coefficient of network complexity (*CNC*)	1.5
Resource factor (*RF*)	1
Resource strength (*RS*)	0.5
Number of resource types	2
Activity resource (per period) demand	[1,10]

The indication [x,y] means that the value is randomly generated in the interval [x,y]. Resource availability is assumed to be constant over time. For each combination of parameters (the number of activities and job subset strength), 10 problem instances were generated. The resource factor *RF* reflects the average portion of the resource required per activity. The resource strength *RS* reflects the scarceness of the resource. The job subset strength *JSS* introduced by Drexl et al. (2000) is an index which determines the number of disjoint subsets of activities, U, depends on the number of project activities, n, according to:

$$U = n\,(1 - JSS) \qquad \text{with} \quad JSS \in [0, 1] \qquad (8)$$

If $JSS = 0$, then n activity subsets with one activity per subset are created. If $JSS = 1$, then $U = 3$ activity subsets are created with $u_1 = \{1\}$ (dummy start activity), $u_2 = \{2,\ldots,n\text{-}1\}$, $u_3 = \{n\}$ (dummy finish activity).

4.2. Effects of the bounding rules

We have coded the branch and bound procedure in Borland C++ version 5.02. The problem set has been solved under Windows XP professional on a personal computer with an Intel Core2Dou, 2.5GHz processor and 3GB of memory.

Table 5 shows the average and the standard deviation of the CPU-time, in seconds, for a different number of activities and *JSS* using all the bounding rules. (The limit on the computational time value was set to 1000 seconds).

Table 5
The average and the standard deviation of the CPU-time, in seconds, using all the bounding rules

JSS	0.5		0.6		0.7	
# Activities	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
20	0.37	0.42	0.26	0.20	0.09	0.08
25	4.54	9.06	2.31	2.58	0.80	0.74
30	87.01	45.66	19.02	16.41	4.99	7.17

As Table 5 indicates, all 90 problems can be solved to optimality within the allowed time limit. It is apparent that the average computation time as well as the standard deviation of the computation time increase as the number of activities increases. Table 5 also reveals that an increase in the job subset strength *JSS* leads to a decrease in the problem complexity, measured by the average as well as the standard deviation of the CPU-time.

In order to estimate the effects of bounding rules on the performance of the procedure for the MIRCPSP, we add a counter to the algorithm to enumerate the fathomed nodes with each bounding rule. Table 6 displays the average percentage of the fathomed nodes with each bounding rule for a different number of activities, subsets and *JSS* when all the bounding rules are included in the algorithm. It’s clear that the new bounding rule 5 is the most efficient one with the percentage of 42.38% and the bounding rules 6 and 7 are the least efficient ones with the percentages of 0.73% and 0.12%, respectively.

Because of the lower efficiency of bounding rule 7, we run again the proposed algorithm without this bounding rule. Table 7 represents the average and the standard deviation of the CPU-time, in seconds, for a different number of activities and *JSS* without using bounding rule 7. Table 7 reveals that the elimination of bounding rule 7 leads to a decrease in the problem complexity, measured by the average as well as the standard deviation of the CPU-time.

The percentages of the fathomed nodes with the bounding rules in Table 8 demonstrate that with the elimination of bounding rule 7, contribution of the new bounding rule 5 is increased more than the others.

Table 6
The average percentage of the fathomed nodes when all the bounding rules are included in the algorithm

# Activities	JSS	# Subsets	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7
20	0.5	11	9.36%	26.27%	20.65%	42.64%	1.00%	0.08%
20	0.6	9	11.69%	38.86%	16.34%	32.10%	0.71%	0.30%
20	0.7	7	10.71%	30.99%	7.14%	49.21%	1.81%	0.14%
25	0.5	14	7.74%	33.68%	14.58%	43.68%	0.27%	0.04%
25	0.6	11	9.27%	33.25%	13.09%	43.10%	1.15%	0.14%
25	0.7	9	9.08%	39.31%	4.88%	45.83%	0.56%	0.34%
30	0.5	16	8.38%	26.67%	21.76%	42.76%	0.43%	0.00%
30	0.6	13	10.46%	29.04%	13.73%	46.34%	0.42%	0.01%
30	0.7	10	9.55%	46.10%	8.32%	35.78%	0.22%	0.02%
Total Average			9.58%	33.80%	13.39%	42.38%	0.73%	0.12%

Table 7
The average and the standard deviation of the CPU-time, in seconds, without using bounding rule 7

JSS	0.5		0.6		0.7	
# Activities	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
20	0.35	0.40	0.25	0.20	0.08	0.07
25	4.43	8.84	2.24	2.51	0.75	0.70
30	84.48	43.55	18.54	16.01	4.84	7.00

Table 8
The average percentage of the fathomed nodes without using bounding rule 7

# Activities	JSS	# Subsets	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
20	0.5	11	9.36%	26.29%	20.67%	42.68%	1.00%
20	0.6	9	11.72%	38.98%	16.39%	32.20%	0.71%
20	0.7	7	10.72%	31.04%	7.15%	49.28%	1.81%
25	0.5	14	7.74%	33.69%	14.59%	43.70%	0.27%
25	0.6	11	9.28%	33.30%	13.10%	43.16%	1.15%
25	0.7	9	9.11%	39.45%	4.90%	45.99%	0.56%
30	0.5	16	8.38%	26.67%	21.76%	42.76%	0.43%
30	0.6	13	10.46%	29.04%	13.73%	46.34%	0.42%
30	0.7	10	9.55%	46.11%	8.32%	35.79%	0.22%
Total Average			9.59%	33.84%	13.40%	42.43%	0.73%

A Paired t-test is used to test the mean difference between 90 paired observations in the algorithm outputs with and without bounding rule 7. The analysis of variance results in Table 9 show that at 95% confidence level, the null hypothesis is rejected. It means that there is a significant difference between the mean solutions of the results. So, the elimination of bounding rule 7 is justified.

Table 9
The analysis of variance results with and without bounding rule 7

	N	Mean	StDev	SE Mean
All Rules	90	13.26	31.17	3.29
Without 7	90	12.89	30.15	3.18
Difference	90	0.380	1.072	0.113

95% lower bound for mean difference: 0.192
T-Test of mean difference = 0 (vs > 0):
T-Value = 3.36 P-Value = 0.001

We also test effects of the elimination of bounding rule 6 (Order Swap Rule) on the performance of the procedure for the MIRCPSP. Bounding rule 6 is the second least efficient according to Table 6. We run again the proposed algorithm without bounding rules 6 and 7. In Table 10, the impact of the elimination of bounding rule 6 on the complexity of the problem instances is examined. Clearly, the effect of the elimination of bounding rule 6 is not monotonously increasing or decreasing. That means there is a clear difference between the complexity of the problems with small, medium and large *JSS* values and the number of activities. With a small *JSS* value (*JSS*=0.5) the elimination of bounding rule 6 leads to an increase in the problem complexity measured by the average as well as the standard deviation of the CPU-time, in comparison with Table 7 where only bounding rule 7 is removed. With the medium and large *JSS* values (*JSS*=0.6 and 0.7) this conclusion is true only for the problems with small and medium number of activities (# of activities = 20 and 25). For the problems with large number of activities (# of activities = 30) with medium and large *JSS* values (*JSS*=0.6 and 0.7), the elimination of bounding rule 6 leads to a decrease in the problem complexity (indicated in italics and boldface in Table 10).

The percentages of the fathomed nodes with the bounding rules in Table 11 demonstrate that with the elimination of bounding rule 6, contribution of the new bounding rule 5 is increased more than the others. Of course, contribution of the bounding rule 3 is decreased.

A Paired t-test is used to test the mean difference between 90 paired observations in the algorithm outputs with and without bounding rule 6. The analysis of variance results in Table 12 show that at 95% confidence level, the null hypothesis is accepted. It means that there is no significant difference between the mean solutions of the results. So, bounding rule 6 should be kept in the algorithm.

Table 10
The average and the standard deviation of the CPU-time, in seconds, without using bounding rules 6 and 7

JSS	0.5		0.6		0.7	
# Activities	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
20	0.37	0.55	0.29	0.30	0.10	0.08
25	4.48	10.04	2.60	3.00	0.82	1.33
30	100.45	65.44	***18.48***	***15.80***	***3.63***	***4.06***

Table 11
The average percentage of fathomed nodes without using bounding rules 6 and 7

# Activities	JSS	# Subsets	Rule 2	Rule 3	Rule 4	Rule 5
20	0.5	11	9.15%	22.31%	22.05%	46.49%
20	0.6	9	12.96%	32.14%	18.85%	36.06%
20	0.7	7	10.70%	24.72%	6.41%	58.17%
25	0.5	14	7.93%	32.11%	15.00%	44.96%
25	0.6	11	9.06%	27.72%	12.86%	50.36%
25	0.7	9	8.71%	27.27%	5.85%	58.16%
30	0.5	16	8.52%	20.94%	22.74%	47.80%
30	0.6	13	10.60%	25.22%	14.00%	50.18%
30	0.7	10	10.41%	36.61%	10.05%	42.93%
Total Average			9.78%	27.67%	14.20%	48.35%

Table 12
The analysis of variance results with and without bounding rule 6

	N	Mean	StDev	SE Mean
Without 7	90	12.89	30.15	3.18
Without 6,7	90	14.58	37.85	3.99
Difference	90	-1.70	14.58	1.54

95% upper bound for mean difference: 0.86
T-Test of mean difference = 0 (vs < 0):
T-Value = -1.10 P-Value = 0.136

5. Summary and Conclusions

This paper presents an exact branch and bound procedure for the mode identity and resource constrained project scheduling problem, in which a set of activities is partitioned into disjoint subsets while all activities forming one subset have to be processed in the same mode. The objective is to schedule the activities in order to minimize the project duration. Depth first branching is based on the precedence tree approach. Seven rules including six bounding rules adapted from the literature and a new rule are used for node fathoming. Finally, the new branch and bound procedure is used for solving some test problems and the computational results demonstrate that the proposed branch and bound is in fact capable of solving problems in an acceptable time. The statistical analysis also revealed that the new proposed bounding rule is more efficient in comparison with other rules available in the literature.

6. References

[1] Błazewicz, J., Lenstra, J. K. and Rinnooy Kan, A. H. G., (1983). Scheduling subject to resource constraints, Discrete Applied Mathematics 5, 11–24.

[2] Kolisch, R., (1995). Project Scheduling under Resource Constraints Efficient Heuristics for Several Problem Classes, Physica, Heidelberg.

[3] Talbot, F. B., (1982). Resource-constrained project scheduling with time-resource trade-offs: the non-preemptive case. Management Science 28 (10), 1197–1210.

[4] Speranza, M. G. and Vercellis, C., (1993). Hierarchical models for multi-project planning and scheduling. European Journal of Operational Research 64 (2), 312–325.

[5] Zhu, G. , Bard, J. F. and Yu, G., (2006). A branch-and-cut procedure for the multi-mode resource-constrained project scheduling problem. INFORMS Journal on Computing 18 (3), 377–390.

[6] De Reyck, B. and 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), 152–174.

[7] Dornorf, U., (2002). Project Scheduling with Time Windows: From Theory to Applications. Physica, Heidelberg

[8] Heilmann, R., (2003). A branch-and-bound procedure for the multi-mode resource constrained project scheduling problem with minimum and maximum time lags. European Journal of Operational Research 144 (2), 348–365.

[9] De Reyck, B. and Herroelen, W. S., (1999). The multi-mode resource-constrained project scheduling problem with generalized precedence relations. European Journal of Operational Research 119 (2), 538–556.

[10] Heilmann, R., (2001). Resource-constrained project scheduling: a heuristic for the multi-mode case. OR Spectrum 23 (3), 335–357.

[11] Calhoun, K. M., Deckro, R. F., Moore, J. T., Chrissis, J. W. and Van Hove, J. C., (2002). Planning and re-planning in project and production scheduling. Omega 30 (3), 155–170.

[12] Van Hove, J. C. and Deckro, R. F., (1998). Multi-modal project scheduling with generalized precedence constraints. In: Babarasogˇlu, G., Karabati, S., Ozdamar, L. , Ulusoy, G. (Eds.), Proceedings of the Sixth International Workshop on Project Management and Scheduling, Istanbul, pp. 137–140

[13] Barrios, A., Ballestin, F. and Valls, V., (2009). A double genetic algorithm for the MRCPSP/ max. Computers and Operations Research, doi: 10.1016/j.cor.2009.09.019.

[14] Salewski, F., Schirmer, A. and Drexl, A., (1997). Project scheduling under resource and mode identity constraints. European Journal of Operational Research, 102, 88–110.

[15] Drexl, A., Juretzka, J., Salewski, F. and Schirmer, A., (1998). New Modeling Concepts and Their Impact on Resource-Constrained Project Scheduling. In: Weglarz J, Editor. Project Scheduling- Recent Models, Algorithms and Applications, Boston: Kluwer Academic Publishers, 413–32.

[16] Coelho, J. and Vanhoucke, M., (2011). multi-mode resource-constrained project scheduling using RCPSP and SAT solvers, European Journal of Operational Research, 213, 73-82.

[17] Kyriakidis, T. S., Kopanos, G. M. and Georgiadis, M. C., (2012). MILP formulations for single- and multi-mode resource-constrained project scheduling problems Computers & Chemical Engineering, Volume 36, 369-385.

[18] Demeulemeester, E. L, (2002). Herroelen W. Project scheduling: a research handbook. Boston: Kluwer Academic Publishers.

[19] Patterson, J. H., Slowinski, R., Talbot, F. B. and Weglarz, J., (1989). An algorithm for a general class of precedence and resource constrained scheduling problems. In: Slowinski R, Weglarz J, Editors. Advances in Project Scheduling, Amsterdam: Elsevier Science Publishers, 3–28.

[20] Sprecher, A., (1994). Resource-Constrained Project Scheduling: Exact Methods for the Multi-Mode Case. Berlin: Springer.

[21] Sprecher, A. and Drexl, A., (1998). Solving multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. European Journal of Operational Research, 107, 431–50.

[22] Hartmann, S. and Drexl, A., (1998). Project scheduling with multiple modes: a comparison of exact algorithms. Networks, 32, 283–97.

[23] Sprecher, A., Hartmann, A. and Drexl, A., (1997). An exact algorithm for project scheduling with multiple modes. OR Spectrum, 19, 195–203.

[24] Demeulemeester, E. L. and Herroelen, W., (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management Science, 38, 1803–18.

[25] Drexl, A., Nissen, R., Patterson, J. H. and Salewski, F., (2000). ProGen/πx - An instance generator for resource constrained project scheduling problems with partially renewable resources and further extensions. European Journal of Operational Research, 125, 59–72.