Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 175 (2006) 279–295

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting

Jirachai Buddhakulsomsiri [a,*], David S. Kim [b]

[a] *School of Manufacturing Systems and Mechanical Engineering, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, 12121 Thailand*
[b] *Department of Industrial and Manufacturing Engineering, Oregon State University, Corvallis, OR 97331, USA*

Received 17 October 2003; accepted 14 April 2005
Available online 27 June 2005

Abstract

This paper presents results from an extensive computational study of the multi-mode resource-constrained project scheduling problem when activities can be split during scheduling under situations where resources may be temporarily not available. All resources considered are renewable and each resource unit may not be available at all times due to resource vacations, which are known in advance, and assignment to other finite duration activities. A designed experiment is conducted that investigates project makespan improvement when activity splitting is permitted in various project scenarios, where different project scenarios are defined by parameters that have been used in the research literature. A branch-and-bound procedure is applied to solve a number of small project scheduling problems with and without activity splitting. The results show that, in the presence of resource vacations and temporary resource unavailability, activity splitting can significantly improve the optimal project makespan in many scenarios, and that the makespan improvement is primarily dependent on those parameters that impact resource utilization.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Multi-mode resource-constrained project scheduling; Activity splitting; Time-varying resource capacities; Branch-and-bound

1. Introduction

Many real-world scheduling problems can be modeled as multi-mode resource-constrained project scheduling problems (MRCPSP). The MRCPSP consists in scheduling project activities to complete a

[*] Corresponding author. Tel.: +1 313 583 6312; fax: +1 313 593 3692.
E-mail addresses: jirachai@siit.tu.ac.th (J. Buddhakulsomsiri), david.kim@orst.edu (D.S. Kim).

0377-2217/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2005.04.030

project in the minimum possible time under the presence of precedence and resource constraints. In this paper, activities require resources and all resources are assumed to be renewable (no non-renewable resources are included), which are available in limited amounts, and each activity must be performed in one of several possible modes (with each mode possibly having different activity durations, and different resource requirements).

In actual scheduling problems that can be modeled as MRCPSPs resources will not be available at all times. In some cases the periods of resource unavailability are known in advance. This is particularly true when the resources are human resources. Human resources can become unavailable due to vacations, special projects, training, or an unlimited number of other reasons, and these types of absences are usually known in advance. For non-human resources predictable absences often take the form of scheduled maintenance, overhauls, or the use of a machine for a special activity. The sources of resource unavailability, as described in the prior examples, and which are typically known in advance, are referred to as *resource vacations*. The result of incorporating resource vacations is that the initial availability of resources varies over time, which is the same result as using time-varying resource capacities as defined in Drexl and Grünewald (1993) (there is a difference in how the varying resource capacities are generated). Real examples of MRCPSPs with resource vacations are the scheduling of engineering design activities in product development; the scheduling of activities in financial audits; and the assignment of coding activities in large software development.

In addition to resource vacations, which cause initial resource availability to vary over time, temporary resource unavailability is also caused by scheduling activities (when activities are scheduled serially). Unlike temporary resource unavailability caused by resource vacations this type of resource unavailability is not known in advance.

The most current heuristics and exact algorithms for the MRCPSP assume that an activity, once started, cannot be interrupted. In other words, each activity in a project can be scheduled only when the precedence constraints are satisfied and the required resources are available for the duration of the activity. This assumption may lead to schedules that can be significantly improved if activities can be split (i.e., suspended and restarted) around unavailable resources when scheduling. Examples of such work can be listed and classified as follows (for a thorough classification of RCPSPs, see Brucker et al., 1999; Herroelen et al., 1999):

- Exact methods—Hartmann and Drexl (1998), Kolisch et al. (1995), Heilmann (2003), Sprecher and Drexl (1998), Sprecher et al. (1997), among others;
- Heuristics (priority rule-based)—Boctor (1993), Kolisch (1996a,b), among others;
- Meta-heuristic methods—Alcaraz et al. (2003), Bouleimen and Lecocq (2003), Hartmann (2001), Heilmann (2001), Jozefowska et al. (2001), Nonobe and Ibaraki (2001), among others.

The focus and contribution of this work is to provide evidence, through computational experiments, into what types of project scenarios may result in significant makespan improvements with activity splitting, and how such scenarios can be characterized. Different project scenarios are investigated using test problem sets that are generated using parameters that characterize different resource requirement levels, resource capacities, and the periods and lengths of resource unavailability. The problem instances generated are solved to optimality with and without activity splitting using the branch-and-bound algorithm of Hartmann and Drexl (1998). Modifications are made to the algorithm so that it is capable of solving pre-emptive problems (details of the branch-and-bound algorithms used are in Appendix B). Additionally, splitting activities in many actual projects is real (especially when human resources are involved) and thus further understanding of the benefit and/or necessity of this practice will be of value.

The remainder of the paper is organized as follows: In Section 2, the mathematical model of the MRCPSP with activity splitting is presented. Section 3 contains a relevant literature review of the MRCPSP that consider activity splitting. In Section 4 the empirical approach used in this research is described, and in Section 5 the computational experiments and results are presented. Finally, conclusions are presented in Section 6.

2. Problem description

This section provides the precise description of the problem. The MRCPSP considered adheres to the following assumptions:

- A project consists of J activities represented using an activity-on-node (AON) representation, where nodes represent the activities and arcs denote the precedence relationships. Two dummy activities are introduced. Dummy activity 1 represents the start activity of the project and dummy activity J represents the end activity of the project, where each node is reachable from node 1 and node J is reachable from each node.
- Precedence relationships are *finish–start* with a time lag of zero meaning an activity can be started if and only if all of its predecessors have been completed.
- Each activity j must be performed in one of M_j possible modes, where each activity–mode combination has a fixed duration and requires a constant amount of one or more of R types of renewable resources when activity is being executed.
- Every resource unit within each type of resource has a known vacation schedule.
- Activities can only be split at discrete points in time. When activities are split, they are resumed without additional duration.
- Mode switching is not allowed when activities are resumed after splitting.

Table 1 lists the notation, parameter definitions, and variables used in the mathematical formulation. With the listed notation, the MRCPSP with activity splitting can be represented as follows:

$$\text{Minimize} \quad F_J \tag{1}$$

Subject to:

$$\sum_{m=1}^{M_j} y_{j,m} = 1 \quad \forall j \in \{1,\ldots,J\}, \tag{2}$$

$$\sum_{t=1}^{T} x_{j,m,t} = d_{j,m} \times y_{j,m} \quad \forall j \in \{1,\ldots,J\},\ \forall m \in \{1,\ldots,M_j\}, \tag{3}$$

$$x_{j,m,t} \times t \leqslant F_j \quad \forall j \in \{1,\ldots,J\},\ \forall m \in \{1,\ldots,M_j\},\ \forall t \in \{1,\ldots,T\}, \tag{4}$$

$$x_{j,m,t} \times t + M(1 - x_{j,m,t}) \geqslant S_j \quad \forall j \in \{1,\ldots,J\},\ \forall m \in \{1,\ldots,M_j\},\ \forall t \in \{1,\ldots,T\}, \tag{5}$$

$$F_i \leqslant S_j - 1 \quad \forall j \in \{1,\ldots,J\},\ \forall i \in P_j, \tag{6}$$

$$\sum_{j=1}^{J}\sum_{m=1}^{M_j} \left(k_{j,m,r} \times x_{j,m,t}\right) \leqslant K_{r,t}, \quad \forall r \in \{1,\ldots,R\},\ \forall t \in \{1,\ldots,T\}, \tag{7}$$

$$S_j \geqslant 0 \quad \forall j \in \{1,\ldots,J\}. \tag{8}$$

In the formulation, the objective function (1) minimizes the project makespan. Constraint set (2) ensures that each activity j is executed in only one mode. Constraint set (3) ensures that each activity j is executed only once, and that the total number of periods that activity j is executed is equal to the duration of that

Table 1
Notation, index sets, parameters, and decision variables

Index sets	
i, j	The activity indices, $i, j \in \{1, \dots, J\}$
m	The mode index, $m \in \{1, \dots, M_j\}$
t	The time period index, $t \in \{1, \dots, T\}$
r	The resource index, $r \in \{1, \dots, R\}$
Parameters	
J	The total number of activities in the project
$d_{j,m}$	Duration of activity j if it is executed in mode m
M_j	The number of modes for activity j
T	The upper bound of the project completion time, $T = \sum_{j=2}^{J-1} \max_{m=1}^{M_j}\{d_{j,m}\}$
R	The total number of renewable resources in the project
P_j	The set containing all activities i that precede activity j
$k_{j,m,r}$	Units of resource r required by activity j if it is executed in mode m
$K_{r,t}$	The capacity of renewable resource r available for period t
Decision variables	
$y_{j,m}$	Binary variable representing whether activity j is being executed in mode m, $y_{j,m} \in \{0,1\}$; $\forall j \in \{1, \dots, J\}$, $\forall m \in \{1, \dots, M_j\}$
$x_{j,m,t}$	Binary variable representing whether activity j, executed in mode m, is being executed at time t, $x_{j,m,t} \in \{0,1\}$; $\forall j \in \{1, \dots, J\}$, $\forall m \in \{1, \dots, M_j\}$, $\forall t \in \{1, \dots, T\}$
S_j	Represents the start time of activity j, $S_j \in \Re$
F_j	Represents the finish time of activity j, $F_j \in \Re$

activity when executed in mode m. Constraint sets (4) and (5) compute the finish time and start time for each activity j, respectively. Constraint set (6) represents the finish–start precedence relationships. Finally, constraint set (7) forces the total units of resource utilized to be no greater than the available resource capacity for every period.

3. Relevant literature review

RCPSP research with activity splitting (referred to as pre-emption) is found in Bianco et al. (1999), Demeulemeester and Herroelen (1996), and Valls et al. (1999), and has so far considered only the single mode problem.

Bianco et al. (1999) considered the problem of scheduling activities on dedicated resources, where each renewable resource can execute one activity at a time and each activity can be pre-empted at integer points of time and resumed later without additional duration. Their solution procedure is an exact algorithm using a graph-theoretical method based on new coloring techniques adapted from the exact algorithm for solving the weighted-vertex coloring problem. They established that the feasible/optimal solution of the weighted coloring problem is equivalent to one of the feasible/optimal solutions (since there may be several feasible/optimal solutions) of the resource-constrained project scheduling problem without precedence relations. The modifications of the mathematical model that represents the coloring problem are then made to incorporate the precedence relations of the problem so that the exact algorithm for the coloring technique can be used to solve the resource-constrained project scheduling problem. In order to establish the connection/analogy between the two problems, they have to assume that one resource can be used to execute only one activity at a time (and therefore do not consider multiple interchangeable resource units).

Demeulemeester and Herroelen (1996) used a branch-and-bound procedure for scheduling RCPSPs with activity splitting (called pre-emption), where each activity is split into subactivities with unit duration. Their procedure employs a depth-first solution strategy in which nodes in the search tree correspond to precedence and resource feasible partial schedules (Demeulemeester and Herroelen, 1992, 1995). In Demeulemeester and Herroelen (1996), computational experiments are performed on 110 problems from Patterson (1984) and 110 problems from Simpson (unpublished thesis (1991) referenced in Demeulemeester and Herroelen (1996)). Scheduling with activity splitting required an average of 33 times more computation time for the Patterson (1984) problems with an average makespan improvement of 0.7823%. A relative decrease in makespan of 2.8802% was reported on the Simpson problems with a 12-fold increase in average computational time. *Their conclusion was that activity splitting provides little benefit except when resource availability is variable*. (The Patterson (1984) problems had no variability in resource capacity, whereas the problems from Simpson had such variability.)

Valls et al. (1999) addressed a class of RCPSP with stochastic activity interruptions. A project consists of a set of activities, with fixed durations, that may not be interrupted, and a set of activities that may be interrupted by a random amount of time. Each activity that may be interrupted has an initial known fixed duration, however, the length of the interruption and the additional processing time resulting from the interruption are random. A heuristic was developed that can schedule other activities during the interruption of stochastic activities to take advantage of idle resources. However, the work did not focus on activity splitting in general to improve a schedule.

In summary, prior work has considered only single-mode RCPSP. Of this work only Demeulemeester and Herroelen (1996) explicitly considered activity splitting. Their conclusions are based on results obtained for problem sets constructed prior to the development of ProGen (Kolisch et al., 1992, 1995; Kolisch and Sprecher, 1996) and therefore may not effectively cover the "space" of RCPSP types. Our research has similarity to Demeulemeester and Herroelen (1996) except that multi-mode problems are considered; properties of resource vacations (frequency of occurrence and relative length) are included; and the sample of project problems examined uses a designed experiment in combination with ProGen.

4. Approach

The focus of this work is on developing evidence into what types of project scenarios may result in significant makespan improvements with activity splitting in the presence of known resource vacations. To this end the approach used was a designed computational experiment that includes the three factors found in ProGen (Kolisch et al., 1992, 1995; Kolisch and Sprecher, 1996) to characterize a RCPSP and two additional factors added to characterize resource vacations. Small problems (13 activities maximum) were generated with a "modified ProGen" (ProGen was fully implemented, based on Kolisch et al. (1992, 1995), and then extended for use in this research) that includes the two new factors characterizing resource vacations (see Appendix A for details). Each problem was optimally solved with and without activity splitting and the results (makespans) were compared. A brief description of each experimental factor is given next (factors 1–3 are ProGen factors).

1. Network complexity (NC) is a measure of project network complexity. It is the average number of nonredundant arcs per node.
2. Resource factor (RF) indicates the average number of resources required per activity. Normalized on a 0–1 scale, $RF = 0$ indicates all activities require no resources, and $RF = 1$ indicates that each activity requires some amount of each resource.
3. Resource strength (RS) is a measure of the "tightness" of resource constraints. RS is a normalized parameter on a 0–1 scale. Specifically, $RS = 0$ means that the resource availability is the minimum such that a feasible project schedule exists (i.e., for a specific resource, its capacity is equal to the maximum

Table 2
Data used to generate problem instances

Data item	A1	A2	B	C	D	E	F	G	H	I	J	K
	# Act. expr 1	# Act. expr. 2	Modes per act.	Dur. (time)	# of resr.	# of resr. req.	# of act. 1 successors	# of successors for act. *j*	# of predecessor of act. *j* (last act.)	# of predecessors of activity *j*	Short vac. range (time)	Long vac. range (time)
min	10	8	2	1	2	1	3	1	3	1	1	3
max	10	13	2	10	2	2	3	3	3	3	2	5

demand by any activities), and RS = 1 means that resource availability is not a constraint (i.e., for each resource the resource capacity is equal to the maximum demand imposed by the project when all activities are scheduled at the earliest start time).

4. The percentage of resource unavailability (V_P) is a percent of the total possible resource availability (e.g., total person days if all resources are people and no vacations occur) that resources are unavailable due to vacations.
5. The vacation length factor (V_L) characterizes the "spread" of resource vacations. V_L is between 0 and 1 (see Appendix A) where a smaller value indicates many shorter periods of unavailability spread out over time. The total number of vacation periods is randomly broken into several vacation periods, where "short" vacations and "long" vacations are generated from $U[d_{\min}^{\text{short}}, d_{\max}^{\text{short}}]$ and $U[d_{\min}^{\text{long}}, d_{\max}^{\text{long}}]$ distributions, respectively. V_L is the percent of resource vacations that are of "short" type (see the values of $d_{\min}^{\text{short}}, d_{\max}^{\text{short}}, d_{\min}^{\text{long}}, d_{\max}^{\text{long}}$ in Table 2).

After the problem instances are generated, they are solved to optimality with and without activity splitting using branch-and-bound algorithms. The branch-and-bound algorithms are adapted from the simplified formulation of the precedence tree algorithm by Hartmann and Drexl (1998). Details of the algorithms are provided in Appendix B. The computational experiments were conducted in two phases. The first phase was an initial factor screening that considered two levels for each factor and attempted to identify project scenarios where a makespan improvement due to activity splitting is likely. Both the existence/non-existence (binary response) of a makespan improvement, and the magnitude of makespan improvement were analyzed as responses. The second phase examined more factor levels and considered the magnitude of makespan improvement as the response. A statistical model was fit to describe makespan improvement with activity splitting, as a function of the experimental factors.

Table 2 shows ranges of data used to generate problem instances for both phases. The number of activities in a project (Columns A1 and A2) was limited by computation times. Columns B–I represent ProGen requirements and columns J and K are added to the modified ProGen to specify ranges for different vacation lengths. The maximum number of activities was limited to 13 because of the increased computation times necessary when activity splitting is permitted.

5. Experimental results

5.1. Experiment 1: Initial factor screening

The objective of experiment 1 is to determine the project parameters where activity splitting has a high probability of improving schedules. Factors (defining a project scenario) having no impact on makespan improvement are removed in subsequent experiments. In experiment 1, two levels of the five project factors were considered resulting in a 2^5 full factorial design (see Table 3). We conducted 20 replications

Table 3
Experiment 1: Factor levels

	NC	RF	RS	V_P	V_L
Level 1	1.5	0.5	0.25	0.10	0.0[a]
Level 2	1.8	1.0	0.75	0.20	1.0[b]

[a] All resource vacations are short (generated from $U[1,2]$).
[b] All resource vacations are long (generated from $U[3,5]$).

(20 different problem instances) for each treatment. Factor levels were selected to represent various ranges of the factors.

The analysis for determining the effect of project parameters conducted in this section is comparable to the statistical analysis in Hartmann and Kolisch (2000). Hartmann and Kolisch (2000) used the average deviation from the optimal solution as a response variable, whereas in our research the response is a binary variable indicating whether or not a makespan improvement results from allowing activity splitting. The objective of the analysis is to determine what factors and factor interactions are significant relative to the probability of makespan improvement (decreased makespan) due to activity splitting.

A logistic regression analysis was performed. Logistic regression is appropriate since comparisons of schedules with activity splitting against schedules without activity splitting consider only the presence or absence of makespan improvement (a binary response). Multiple binary responses can be used to estimate a probability of makespan improvement. Logistic regression uses a logarithmic transformation of the odds ratio of the makespan improvement probabilities as a response variable. The overall proportion of schedules experiencing improvement is $\frac{315}{640} = 0.49$. Table 4 gives a summary of factor significance levels.

It can be seen from Table 4 that network complexity has no statistically significant effect on the *probability* of makespan improvement when activity splitting is permitted (a general value to indicate statistical significance is a p-value = 0.05). All other effects are significant with a significant interaction between resource strength and resource vacation length, ($\text{RS} \times V_\text{L}$). Activity splitting has a significant chance of improving the quality of schedule solutions, when (1) activities require more types of resources (high resource factor), (2) the resource constraints are tight (high resource strength), (3) the resource vacation percentage is high, and (4) resource vacations are relatively short. These results are intuitive except the existence of a single significant interaction effect. The results are also consistent with the observation made by Hartmann and Kolisch (2000) that the influence of network complexity on the quality of solution is low.

Experiment 1 results were also analyzed with the magnitude of makespan improvement as the response. This analysis was conducted to confirm that network complexity also has no effect on the magnitude of makespan improvement. Table 5 gives a summary of factor significance levels. The results are similar to the prior results. Based on these results the network complexity in experiment 2 is fixed at the lower level (NC = 1.5) and a full factorial experiment with the remaining factors could be conducted.

5.2. Experiment 2: Magnitude of solution improvement

Experiment 2 is a more detailed computational study of activity splitting to improve schedules. In experiment 2 the network complexity factor (NC) is fixed based on the prior results and more levels of the remaining factors are included. In experiment 2 we consider the magnitude of makespan improvement with activity splitting, *given that activity splitting improves the schedule*, as the response. The prior experiment provides insight into when a makespan improvement with activity splitting is likely. This experiment pro-

Table 4
Significant level of experimental factors with schedule improvement as the response (binary response)

Factor	NC	RF	RS	V_P	V_L	$\text{RS} \times V_\text{L}$
p-Value	0.2044	<0.0001	<0.0001	<0.0001	<0.0001	0.0467

Table 5
Significant level of experimental factors with makespan improvement percentage as the response

Factor	NC	RF	RS	V_P	V_L	$\text{RS} \times V_\text{L}$
p-Value	0.4277	<0.008	<0.0007	<0.0001	<0.0001	0.0017

vides further information on identifying project scenarios where large makespan improvements may be expected with activity splitting.

This experiment includes all four significant factors from experiment 1, namely, RF, RS, V_P, and V_L. Experiment 2 is a $3 \times 4 \times 3 \times 3$ full factorial design, with additional factors levels added as shown in Table 6.

Makespan improvement due to activity splitting is measured as the percent improvement in makespan and is calculated as follows:

$$\text{Makespan improvement} = \frac{\text{makespan w/o splitting} - \text{makespan with splitting}}{\text{makespan w/o splitting}}.$$

For each combination of problem sizes (8–13 activities) and problem parameters, five different project instances were generated. This yields 3240 individual problem instances that are solved to optimality with and without activity splitting.

From the 3240 observations, there were 1685 schedules with makespan improvement due to activity splitting (52%). Table 7 presents the distribution of makespan improvements and Table 8 is a summary of the overall results.

As can be seen from the tables the majority of the schedules with improvement fall in the category of 10% or less improvement. There are however, about 49% of the schedules with a greater than 10% improvement, so that a portion of the "space" of MRCPSPs can benefit significantly from activity splitting.

To identify the significant factors with respect to the magnitude of makespan improvement due to activity splitting, a multiple linear regression is performed with the log-transformed percent improvement as a response (with the four main factors as explanatory variables, and the number of activities as a continuous explanatory variable). The transformation of the response is necessary to satisfy the statistical model assumptions (normality and homogeneity of residual variance), and a continuous variable is included to reduce the experimental error variance. A summary of experimental factor significance is given in Table 9.

Table 6
Experiment 2: Factor levels

	RF	RS	V_P	V_L
Level 1	0.50	0.25	0.10	0.0
Level 2	0.75	0.50	0.20	0.50[a]
Level 3	1.0	0.75	0.30	1.0
Level 4		1.0		

[a] Resource vacations are a 50–50 mixed of short $U[1,2]$ and long $U[3,5]$ vacations.

Table 7
Distribution of makespan improvement

Percent improvement	$\leqslant 10$	10–20	20–30	30–40	40–50	50–60	60–70	>70	Total
Count	858	482	138	80	55	43	27	2	1685

Table 8
Summary of descriptive statistics for makespan improvement

Mean	Median	Standard deviation	Minimum	Maximum
14.84%	10.00%	13.64%	1.92%	75.00%

Table 9
Significant level of experiment 2 factors

	# Activities	RF	RS	V_P	V_L	RS × V_P
p-Value	<0.0001	0.0814	<0.0001	<0.0001	<0.0001	<0.0001

The results from Table 9 indicate the significance of three main effects, an interaction term, and the number of activities in a project (the effect of each factor on the percent improvement estimated by the regression analysis is a linear effect on a log-scale so the factor effects are multiplicative when transformed to the original scale). The results show that the average number of requested resources for each activity (resource factor—RF) has no statistically significant effect on the magnitude of solution improvement due to activity splitting.

Activity splitting has a greater chance of large makespan improvements, when (1) the resource constraints are tight (high resource strength), (2) percent resource vacation is high, and (3) resource vacations are short (relatively). The final regression model is

$$\begin{aligned}\log(\text{makespan improvement}) = {} & 2.32 - 0.05J - 0.16\text{RS}_{0.50} - 0.14\text{RS}_{0.75} - 0.29\text{RS}_{1.0} + 0.55V_{\text{P},0.2} \\ & + 1.07V_{\text{P},0.3} + 0.26V_{\text{L},0.5} + 0.40V_{\text{L},1.0} - 0.24\text{RS}_{0.5} \times V_{\text{P},0.2} \\ & - 0.36\text{RS}_{0.5} \times V_{\text{P},0.3} - 0.38\text{RS}_{0.75} \times V_{\text{P},0.2} - 0.56\text{RS}_{0.75} \times V_{\text{P},0.3} \\ & - 0.42\text{RS}_{1.0} \times V_{\text{P},0.2} - 0.57\text{RS}_{1.0} \times V_{\text{P},0.3},\end{aligned}$$

where RS_i, $V_{\text{P},i}$, and $V_{\text{L},i}$ are all binary indicator variables, which take a value of 1 when the factor is at level i (see Table 4), and 0 otherwise, and J ranges from 8 to 13.

Fig. 1 shows the relationships between resource strength and percent resource vacation on the median percent improvement in schedule makespan. Makespan improvements tend to decrease as resource constraints become looser, which is intuitive. The results also indicate that the makespan improvements are greater for higher amounts of resource vacations, and that the difference in improvements increases as resources become more limited. It can be seen that in some cases the improvement due to activity splitting is very large.

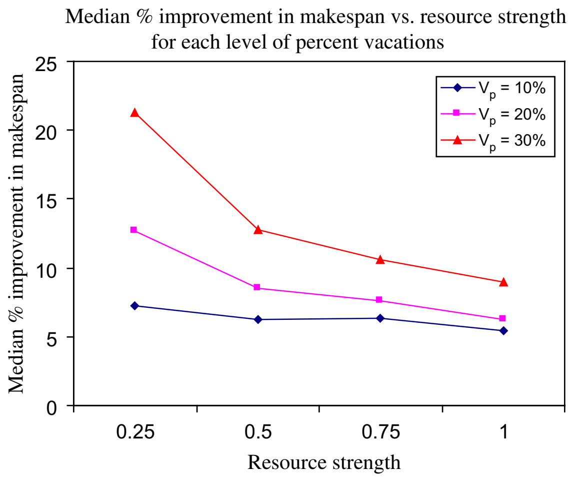

Fig. 1. Effect of resource strength on makespan improvement due to activity splitting.

6. Conclusions

This study investigated potential makespan improvements obtainable from allowing activity splitting in the scheduling of the multi-mode resource-constrained project scheduling problem where renewable resources may be temporarily unavailable. Full factorial experiments are conducted where experimental factors are project parameters that characterize different resource scenarios. A branch-and-bound procedure is applied to solve small problem instances with and without activity splitting to optimality. Statistical analyses to compare the two optimal solutions are performed to address whether or not activity splitting can benefit the scheduling solutions, and under what resource scenarios that significant improvement can be obtained.

Results from the computational experiments lead to insights about the pre-emptive MRCPSP that the project parameters having a significant effect on whether activity splitting improves a schedule are resource factor, resource strength, percent of resource vacations, and length of resource vacations. Further analysis of these parameters indicates that activity splitting has a better chance of improving schedules when a project is more difficult to schedule (i.e., high resource factor, high percent of resource vacation, and high variability of resource availability due to shorter length of vacation). With respect to the magnitude of makespan improvement, similar conclusions apply except that (surprisingly) resource factor does not have a significant effect. All significant factors are directly involved with the availability of resources (reflected by resource strength and percent resource vacation) as well as its variation (parameterized as vacation length).

It must be noted that in order to make a statistically valid statements, the scope of inference from the results of this research is limited to MRCPSPs within the range of project parameters used in these experiments. However we believe that the results presented provide evidence that activity splitting will improve schedules for larger MRCPSP problems. It should also be mentioned that the number of activity splits was not controlled. In reality uncontrolled splitting, even though beneficial to makespan, could result in unrealistic or unpractical schedules, so that the makespan improvements presented can be considered ‘‘best case’’ improvements.

The conclusions of this research suggest that in difficult project situations with limited resource availability, activity splitting can significantly improve a schedule compared to one where activity splitting is not allowed. Because this is often the case in real scheduling problems there is a need for an efficient heuristic procedure that can perform scheduling with activity splitting (and control activity splitting such that it is allowed only when it will likely provide improvement) on large-scaled MRCPSPs. In a follow-on paper (Buddhakulsomsiri and Kim, 2004) such a heuristic has been developed and tested.

Acknowledgments

The authors would like to thank the referees for their comments and suggestions which have considerably improved this paper.

Appendix A. Project generator

The problem instance generator constructed and used in this research is a modification of ProGen developed by Kolisch et al. (1992, 1995) and Kolisch and Sprecher (1996). All three ProGen parameters, namely network complexity, resource factor, and resource strength, are incorporated in our problem generator. Detailed descriptions of ProGen can be found in Kolisch et al. (1992, 1995). For our research we included two additional parameters to generate resource vacations: V_P is the percent of time resources are not available due to vacations, and V_L is the percent of vacations that are relatively "short" in length. Generation of resource vacations involves three steps as follows:

Generating resource vacations

Step 1: After a problem instance is generated using ProGen (with 3 parameters), a percentage of unavailability due to resource vacations as a fraction of the planning horizon is then specified. The total number of periods that every unit of resource is unavailable is set to be equal to $V_P T$, where T is a project makespan upper bound ($T = \sum_{j=2}^{J-1} \max_{m=1}^{M_j}\{d_{j,m}\}$).

Step 2: The percent of short vacations V_L is specified. The total number of vacation periods from Step 1 is randomly broken into several vacation periods, where short vacations are generated from $U[d_{\min}^{\text{short}}, d_{\max}^{\text{short}}]$ and long vacations are generated from $U[d_{\min}^{\text{long}}, d_{\max}^{\text{long}}]$ ($d_{\min}^{i}$and $d_{\max}^{i}$ are the minimum and maximum durations for vacation type i). The intervals defining short and long vacations are arbitrary but should be set relative to the minimum and maximum activity duration, and the scheduling horizon.

Step 3: Each individual vacation is randomly placed throughout the project planning horizon, ranging from period 1 to period T.

Appendix B. Branch-and-bound algorithm

This section provides a description of the branch-and-bound algorithms used in this paper. In order to investigate the advantage of allowing activity splitting on the MRCPSP with resource vacations, each problem instance generated is solved with and without activity splitting so that project makespan improvements due to activity splitting can be evaluated. The algorithm used in this study is a modification of the precedence tree algorithm, first proposed by Patterson et al. (1989), which is known to generate optimal schedules.

B.1. Algorithm 1: Precedence tree algorithm without activity splitting

The algorithm for solving the MRCPSP without activity splitting is adapted from the simplified formulation of the precedence tree algorithm by Hartmann and Drexl (1998). Table B.1 provides the notation used in the algorithm, and is followed by a description of the algorithm implemented.

Table B.1
Notation for Algorithm 1

Notation	Definition
g	Precedence tree level
j_g	Activity j selected at level g of the precedence tree
m_{j_g}	Mode selected for j_g
M_{j_g}	The number of available modes for j_g
s_{j_g}	Start time of j_g
SJ_g	The set of already scheduled activities at level g of the precedence tree
EJ_g	The set of eligible activities at level g of the precedence tree
P_j	The set of predecessors of activity j
EST_j	The earliest precedence feasible start time of activity j
FT_j	Finish time of activity j

Algorithm 1. Precedence tree algorithm without activity splitting

```
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, $SJ_1 = \emptyset$.
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), $EJ_g = \{j \in \{1, \dots, J\} \backslash SJ_g | P_j \subseteq SJ_g\}$.
    If the last (dummy) activity is eligible $J \in EJ_g$, then store the current solution and go to Step 5.
    Else 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.
    Else select an untested activity, $j_g \in EJ_g$.
Step 4: Select a mode and compute the activity start time
    If there is no untested mode left in $\{1, \dots, M_{j_g}\}$, then go to Step 3.
    Else select an untested mode $m_{j_g} \in \{1, \dots, M_{j_g}\}$.
    Compute the earliest precedence feasible start time, $\mathrm{EST}_{j_g} = \max\{\mathrm{FT}_i \mid i \in P_j\} + 1$.
    Compute the earliest resource feasible start time $s_{j_g} \geqslant \mathrm{EST}_{j_g}$, 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.
    Else go to Step 4.
```

Algorithm 1 starts with scheduling the first dummy activity at time 0. At each level g of the precedence tree, the set of already scheduled activity SJ_g is updated and the set of precedence eligible activities EJ_g is determined. Then, an untested activity j_g is selected and executed with an untested mode m_{j_g}. With an activity and a mode selected, the earliest precedence and resource feasible start time s_{j_g} is computed. The algorithm continues until the last dummy activity is eligible (a complete schedule is found). Then, backtracking begins until all untested modes of an activity, and all untested activities at each level of the search tree are examined.

A slight modification from the precedence tree algorithm of Hartmann and Drexl (1998) is made in Algorithm 1. The modification is such that the start time s_{j_g} is not restricted to be greater than or equal to the start time assigned on the previous level of the search tree. Instead it is calculated as the earliest precedence and resource feasible start time (similar to start times in the serial SGS by Kelley, 1963). This forces

Table B.2
Additional notation for Algorithm 2

Notation	Definition
CJ_g	The set of currently started but unfinished activities at level g of the precedence tree
$d(j_g, m_{j_g})$	Duration of activity j_g executed in mode m_{j_g}
$\tilde{d}(j_g, m_{j_g})$	Remaining duration of a current started but unfinished activity j_g executed in mode m_{j_g}
DJ_g	The set containing $\tilde{d}(j_g, m_{j_g})$, $DJ_g = \{\tilde{d}(j_g, m_{j_g}) \mid j_g \in CJ_g\}$
EM_{j_g}	The set of eligible mode of j_g

Algorithm 1 to only generate active schedules, as defined by Sprecher et al. (1995). This modification is made without loss of optimality. This algorithm is similar to the precedence tree algorithm of Hartmann and Drexl (1998) that it branches on all possible schedules (where different schedules may have a different order of scheduled activities), and examines each activity in all of its possible modes.

B.2. Algorithm 2: Precedence tree algorithm with activity splitting

Additional steps are included to the precedence tree algorithm to allow activity splitting. Table B.2 contains additional notation used to describe Algorithm 2.

Algorithm 2. Precedence tree algorithm with activity splitting

```
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 $SJ_1 = \emptyset$; $\tilde{d}(j_1, m_{j_1}) = 0$; $CJ_1 = \{1\}$; $DJ_1 = \{1\}$.
Step 2: Compute the set of eligible activities
    Increase the level of the precedence tree, $g = g + 1$.
    Update $SJ_g$, $CJ_g$, $DJ_g$.
        If $\tilde{d}(j_{g-1}, m_{j_{g-1}}) = 0$,
        then $SJ_g = SJ_{g-1} \cup \{j_{g-1}\}$; $CJ_g = CJ_{g-1} \backslash \{j_{g-1}\}$; $DJ_g = DJ_{g-1} \setminus \{\tilde{d}(j_{g-1}, m_{j_{g-1}}) = 0\}$.
    Compute the set of eligible activities (i.e., activities whose predecessors are already scheduled),
    $EJ_g = \{j \in \{1, \dots, J\} \backslash SJ_g | P_j \subseteq SJ_g\}$.
    If the last (dummy) activity is eligible $J \in EJ_g$, then store the current solution and go to Step 5.
    Else 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.
    Else select an untested activity, $j_g \in EJ_g$.
    Compute the set of eligible modes.
        If $j_g \in CJ_{g-1}$, then $EM_{j_g} = \{m_{j_g}\}$.
        Else $EM_{j_g} = \{1, \dots, M_{j_g}\}$.
Step 4: Select a mode and compute the (sub)activity start time
    If there is no untested mode left in $EM_{j_g}$, then go to Step 3.
    Else select an untested mode $m_{j_g} \in EM_{j_g}$.
    Compute the earliest precedence feasible start time.
        If $j_g \in CJ_{g-1}$, then $\mathrm{EST}_{j_g} = s_{j_g} + 1$.
        Else$\mathrm{EST}_{j_g} = \max\{\mathrm{FT}_i \mid i \in P_j\} + 1$.

    Compute the earliest resource feasible start time $s_{j_g} \geqslant \mathrm{EST}_{j_g}$ for a subactivity with unit duration.
    Update $DJ_g$, $CJ_g$, $\mathrm{FT}_j$.
        If $j_g \in CJ_{g-1}$, then $CJ_g = CJ_{g-1}$; $\tilde{d}(j_g, m_{j_g}) = \tilde{d}(j_m, m_{j_g}) - 1$.
        Else $CJ_g = CJ_{g-1} \cup \{j_g\}$; $\tilde{d}(j_g, m_{j_g}) = d(j_g, m_{j_g}) - 1$; $DJ_g = DJ_{g-1} \cup \{\tilde{d}(j_g, m_{j_g})\}$.
        If $\tilde{d}(j_g, m_{j_g}) = 0$, then $\mathrm{FT}_j = s_{j_g}$.
    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.
    Else go to Step 4.
```

Additional steps included in Algorithm 2 to accommodate activity splitting are as follows:

1. At each level g of the search tree, an untested activity with an untested mode is scheduled for only 1 unit duration.
2. For an untested activity selected in Step 3, a list of untested modes EM_{j_g} must be computed. If the selected activity is already started but not yet finished ($j_g \in CJ_{g-1}$), then the only eligible mode is the current mode that activity is being executed in. Otherwise, the selected activity has not yet been started and all modes are eligible.
3. In Step 4, after the activity is scheduled and the start time is computed, DJ_g, CJ_g, FT_j must be updated. This is to determine whether or not the scheduled subactivity j_g finishes that activity (i.e., the scheduled subactivity is the last subactivity).

The additional steps are added without loss of optimality since these steps only affect the status of a scheduled activity at each level of the search tree. The branching and backtracking schemes remain the same. Within each level of the search tree the additional steps only affect whether an eligible activity is (a) not yet started, (b) started but unfinished (thus, still eligible but with only one available mode), or (c) finished.

B.3. Bounding rules

Existing bounding rules including time window-based rules using latest finish times (LFT), data reduction, and precedence tree specific rules are incorporated in both precedence tree algorithms in order to reduce computation times. Detailed descriptions of these bounding rules can be found in Hartmann and Drexl (1998). In addition, the following bounding rules are included in Algorithm 2 to further reduce computation times.

B.3.1. Bounding rule 1: Start backtracking level

Applied when a complete schedule is found and the backtracking process begins. Let g^* be the search tree level where the last unit duration subactivity (of the activity with the largest finish time, i.e., project makespan) is scheduled. Note that g^* is not necessarily the last level of the search tree because the start time $s_{j_g^*}$ is not restricted to be no less than the start time assigned on previous levels of the search tree. Backtracking at the levels of the search tree $g \geqslant g^*$ cannot improve the project makespan. This is because the project makespan is determined at g^* and thus, backtracking should begin at level g^* and all unchecked modes and activities at levels $g \geqslant g^*$ do not need to be considered.

B.3.2. Bounding rule 2: Extended time window rule

Let T denote an upper bound on the project's makespan and let LFT_j denote the latest finish time of activity j computed by the metra-potential method (MPM). At a level of the search tree g, let EFT_{j_g} denote the earliest finish time of an unfinished activity j at level g. Note that an unfinished activity j here can be an activity that has not previously been started (or even eligible) or an activity that has been started but not yet finished. EFT_{j_g} is the finish time of activity j computed as if all remaining subactivities of activity j are to be scheduled successively until activity j is finished. If there is an unfinished activity j (not necessarily started or eligible) which cannot be executed in any of its modes that results in EFT_{j_g} no more than LFT_j, then the current partial schedule cannot be completed with a makespan less than or equal to T.

This bounding rule is included to trigger early pruning of the search tree. To illustrate this rule, suppose at level g' all eligible activities can be executed in one of their modes that generate finish times for all activities to be less than its LFT. Thus, at level g', the time window (LFT) rule will not prune the search tree.

However, if other ineligible activities are considered at this level and it turns out that if an ineligible activity j^* cannot be executed in any of its modes such that a finish time better than LFT_{j^*} can be achieved, then this branch of the search tree should not be considered further because eventually the algorithm will proceed to pruning this branch due to j^* at some $g > g'$.

B.3.3. Bounding rule 3: Good initial solution

This rule can be considered as an additional step of pre-processing. Before Algorithm 2 is executed, a "variant" of Algorithm 1 (which takes considerably less computation time) can be used to obtain an initial solution for Algorithm 2. The variant of Algorithm 1 allows activity splitting in Step 4 (i.e., resources do not need to be available continuously throughout the whole duration of the selected activity executed in the selected mode). Once an activity is selected in Step 3 and a mode is selected in Step 4, all its subactivities will be successively scheduled at this level of the search tree until the selected activity is finished. The underlying idea of this rule is that starting Algorithm 2 from an initial good solution will lead to good bounding criteria, which results in shorter computation times.

All instances were run on a variety of personal computers, ranging from PCs with Pentium II processors at 300 MHz to Pentium III processors at 800 MHz. Within this varied computing environment 97% of the instances were solved to optimality within 60 minutes. Several instances required over 40 hours to obtain an optimal solution.

References

Alcaraz, J., Maroto, C., Ruiz, R., 2003. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. Journal of the Operational Research Society 54, 614–626.

Bianco, L., Caramia, M., Dell'Olmo, P., 1999. Solving a preemptive project scheduling problem with coloring techniques. In: Weglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms, and Applications. Kluwer Academic Publishers, Massachusetts, pp. 135–145.

Boctor, F., 1993. Heuristics for scheduling projects with resource restrictions and several resource-duration modes. International Journal of Production Research 31 (11), 2547–2558.

Bouleimen, K., Lecocq, H., 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. European Journal of Operational Research 149, 268–281.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112, 3–41.

Buddhakulsomsiri, J., Kim, D.A., 2004. Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. European Journal of Operational Research, under review.

Demeulemeester, E.L., Herroelen, W.S., 1996. An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem. European Journal of Operational Research 90, 334–338.

Demeulemeester, E.L., Herroelen, W.S., 1992. A branch-and-bound procedure for the multiple resource-constrained project scheduling problems. Management Science 38 (12), 1803–1818.

Demeulemeester, E.L., Herroelen, W.S., 1995. New benchmark results for the resource-constrained project scheduling problem. Management Science 43 (11), 1485–1492.

Drexl, A., Grünewald, J., 1993. Nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions 25 (5), 74–81.

Hartmann, S., 2001. Project scheduling with multiple modes: A genetic algorithm. Annals of Operations Research 102, 111–135.

Hartmann, S., Drexl, A., 1998. Project scheduling with multiple modes: A comparison of exact algorithms. Networks 32 (4), 283–298.

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

Heilmann, R., 2001. Resource-constrained project scheduling: A heuristic of the multi-mode case. OR Spektrum 23, 335–357.

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, 348–365.

Herroelen, W.P., Demeulemeester, E.L., De Reyck, B., 1999. A classification scheme for project scheduling. In: Weglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms, and Applications. Kluwer Academic Publishers, Massachusetts, pp. 1–26.

Jozefowska, J., Mika, M., Rozycki, R., Waligora, G., Weglarz, J., 2001. Simulated annealing for multi-mode resource-constrained project scheduling. Annals of Operations Research 102, 137–155.

Kelley Jr., J.E., 1963. The critical path method: Resource planning and scheduling. In: Muth, J.F., Thompson, G.L. (Eds.), Industrial Scheduling. Prentice-Hall, New Jersey, pp. 347–365.

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

Kolisch, R., 1996b. Efficient priority rules for the resource-constrained project scheduling problem. Journal of Operational Management 14 (3), 179–192.

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

Kolisch, R., Sprecher, A., Drexl, A., 1992. Characterization and generation of a general class of resource-constrained project scheduling problems. Manuskripte aus den Instituten fur Betriebswirtschaftslehre, No. 301, Kiel.

Kolisch, R., Sprecher, A., Drexl, A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 41 (10), 1693–1703.

Nonobe, K., Ibaraki, T., 2001. Formulation and tabu search algorithm for the resource constrained project scheduling problem. In: Ribeiro, C.C., Hansen, P. (Eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, Dordrecht, pp. 557–588.

Patterson, J.H., 1984. A comparison of exact procedures for solving the multiple constrained resource project scheduling problem. Management Science 30 (7), 854–867.

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

Sprecher, A., Drexl, A., 1998. Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm. European Journal of Operational Research 107, 431–450.

Sprecher, A., Kolisch, R., Drexl, A., 1995. Semi-active, active and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research 80, 94–102.

Sprecher, A., Hartmann, A., Drexl, A., 1997. An exact algorithm for project scheduling with multiple modes. OR Spektrum 19, 195–203.

Valls, V., Laguna, M., Lino, P., Perez, A., Quintanilla, S., 1999. Project scheduling with stochastic activity interruptions. In: Weglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms, and Applications. Kluwer Academic Publishers, Massachusetts, pp. 333–353.