Journal of the Operational Research Society (2005) 56, 406–413

www.palgrave-journals.com/jors

A new heuristic for the multi-mode resource investment problem

C-C Hsu[1]* and DS Kim[2]

[1]*General Motors Engineering, Pontiac, MI, USA; and* [2]*Oregon State University, Corvallis, OR, USA*

This paper considers the problem of minimizing resource investment required to execute the tasks in a project network by a given project due date. A project consists of non-pre-emptive tasks executed in a known and required precedence order. Each task is completed in one of its feasible modes, which may differ not only in task duration but also in consumption of renewable resources. A priority rule heuristic with polynomial computational complexity is presented for this computationally intractable problem. This heuristic simultaneously considers due date constraints and resource usage to select and schedule tasks with one decision rule. This differs from prior multi-mode priority rule scheduling heuristics that apply two consecutive decision rules to schedule tasks. Extensive computational testing indicates promising results.

Journal of the Operational Research Society (2005) **56,** 406–413. doi:10.1057/palgrave.jors.2601827
Published online 29 September 2004

Keywords: scheduling/sequencing; project management; combinatorics

Introduction

There are many examples of real projects where the project due dates are agreed upon or are the result of other higher-level decisions. With such due dates established, the objective is to schedule tasks in the project to minimize the project completion costs. An example of such a project, and one that motivated this research, is vehicle engineering and design. In this application, all the vehicle components must be electronically drafted and the order in which such drawings must be completed represents a project precedence network. Such a huge project is organized at different levels, and the engineering managers who schedule design tasks for the lower level vehicle subsystems (eg front-bumper) take the subsystem due date as a fixed input from the higher level vehicle system (eg exterior system).

The problem of scheduling tasks to meet due dates at minimal cost is called the resource investment problem (RIP), and is one that has received relatively little attention compared to its related problem, the resource constrained project scheduling problem (RCPSP). The RIP, like the RCPSP, is a computationally intractable problem. In this paper, we consider its multi-mode extension, the multi-mode resource investment problem (MMRIP) and present an effective and computationally efficient heuristic.

The remainder of this paper is organized as follows. We first define the MMRIP and present a review of the literature. The logic and details of the new heuristic are presented, and followed by computational results and conclusions.

*Correspondence: C-C Hsu, Operations Research Department, General Motors Engineering, 585 South Boulevard, MC: 483-585-372, Pontiac, MI 48341, USA.
E-mail: chihcheng.hsu@gm.com

Problem definition

The RIP was first introduced when Mohring[1] classified the family of project-scheduling problems into two categories:

(a) *The problem of scarce resources:* given a constant limit K_r for the available amount of resource r, what is the shortest possible project duration?
(b) *The problem of scarce time:* given a limit T for the project duration, what is the minimum total resource cost, assuming unlimited resources and cost $f(K_r)$ for K_r units of resource r?

Category (A) problems are commonly referred to as RCPSPs,[2–5] and category (B) problems were first called resource availability cost problems[6] and later RIPs.[7] This paper uses this later naming convention.

We consider a generalization of RIP, referred to as the MMRIP, in which at least one of the tasks may be undertaken in any of its several modes. For the MMRIP, a project has J non-pre-emptive tasks and R types of renewable resources. A task j takes duration d_{jm} to complete, if it is executed in mode m. Mode m of task j requires k_{jmr} units of resource r in each period of its execution. Without loss of generality, the project is assumed to have a source task (task 1) and an end task (task J), which represent the beginning and the end of the project, respectively. The scheduling problem is to determine the starting time and the

mode for each task j, which in turn determines the demand level of each resource r (K_r), so that the project will be completed by due date T with minimal total resource investment. The mathematical model for this problem is:

$$\text{Minimize} \quad \sum_r c_r K_r$$

Subject to

$$\sum_m \sum_t x_{jmt} = 1 \quad \forall j \tag{1}$$

$$\sum_m \sum_t x_{imt}(t + d_{im}) \leqslant \sum_m \sum_t x_{jmt} t \quad \forall j,\ \forall i \in P_j \tag{2}$$

$$\sum_j \sum_m k_{jmr} \sum_{\tau=t}^{t-d_{jm}+1} x_{jm\tau} \leqslant K_r \quad \forall t,\ \forall r \tag{3}$$

$$\sum_m \sum_t x_{Jmt} t \leqslant T \tag{4}$$

$$x_{jmt} \in \{0, 1\} \quad \forall j,\ \forall m,\ \forall t \tag{5}$$

The indices in this model are:

j task index ($j = 1, 2, \ldots, J$)
m mode index of each task ($m = 1, 2, \ldots, M_j$)
r resource index ($r = 1, 2, \ldots, R$)
t time index ($t = 1, 2, \ldots, T$)

The parameters are:

J total number of tasks in the project
R total number of resources in the project
T project due date
c_r cost for one unit of resource r for the entire project
d_{jm} duration of task j if it is executed in mode m
k_{jmr} units of resource r required by task j if it is executed in mode m
M_j number of modes for task j
P_j set of immediate predecessors for task j

and the variables are:

x_{jmt} 1: if task j starts at time t and executes with mode m; 0: otherwise
K_r units of resource r required

In this model, the objective function is to minimize the total resource investment. Constraints (1) and (5) ensure that each job starts at only one time period, with only one mode. The precedence relationships among tasks are defined in constraint (2). Constraint (3) prevents resource overuse and constraint (4) enforces the project due date.

Literature review

Unlike the RCPSP, little research has focused on the RIP, even though many real RIP applications exist. The first RIP research was conducted by Mohring,[1] and it was not until Demeulemeester[6] that additional RIP research appeared in the literature. Both Mohring[1] and Demeulemeester[6] developed optimal solution procedures for the RIP.

Conceptually, Mohring[1] and Demeulemeester[6] utilize what can be called a two-stage algorithm. In the first stage, a capacity level is assigned to each resource. With this predetermined resource capacity level, the RIP is transformed into a corresponding RCPSP with fixed project duration (RCPSP with due date constraint). The second stage is used to verify whether this transformed problem has a feasible solution.

Mohring's algorithm starts with a high resource capacity level in the first stage. If a feasible solution is confirmed in the second stage, the capacity level is then decreased. The algorithm iterates until feasibility can no longer be confirmed. In that case, the last capacity level becomes the optimal solution. In contrast, Demeulemeester's algorithm starts with a low capacity level for each resource in the first stage. Then capacity level increases until a feasible solution is confirmed.

The only RIP heuristic discovered in the literature was proposed by Neumann and Zimmermann.[7] A modified serial scheme priority rule heuristic was utilized. Unlike the serial schedule generation method for the RCPSP, which schedules selected tasks at the earliest resource-feasible time, tasks are scheduled to minimize resource investment without violating the due date and precedence orders. Their computational experiments indicated promising results. Their heuristic is applicable to the single-mode RIP. To the best of our knowledge, no heuristic has been proposed for the MMRIP.

A multi-mode resource investment problem heuristic

In this section, we introduce a new MMRIP priority rule dispatching heuristic. Priority rule dispatching has been a common heuristic approach to project scheduling problems (see Kurtulus and Davis,[8] Boctor,[9] Lawrence and Morton,[10] Kolisch,[4,11] Neumann and Zimmermann[7]). The new heuristic we describe is a dispatching method, which can be applied within a serial schedule generation scheme (for a detailed review of the serial schedule generation scheme and the parallel schedule generation scheme, refer to Kelley[12] and Kolisch[4]).

A serial schedule generation scheme iteratively builds a complete schedule task by task. A priority rule is used to select and schedule the next task (among the unscheduled tasks). The scheme stops when all tasks have been scheduled.

From single mode to multi-mode: one possible straightforward extension

For the MMRIP, an additional decision not found in the RIP is determining a mode for each task. An intuitive approach is to again apply a priority rule to select a mode for each task. This approach has been used for the multi-mode RCPSP or MMRCPSP.[13,14]

An example of using this approach for the MMRIP is to select a task j^* using the minimum slack rule at each iteration (choose the task that has the minimum slack value, this will be explained later), and then use the minimum investment rule to determine the mode and starting time for task j^* (choose the mode m^* and starting time t^* for task j^* that requires the minimum extra investment). However, this two-priority-rule approach overlooks the potential inter-dependencies between tasks, their starting times, and modes. The development of a new approach is motivated by the following question: 'Can we use a single priority rule to select a good combination of task j^*, mode m^*, and starting time t^*?'

A new approach for the multi-mode case

To demonstrate this idea, we will integrate the two priority rules discussed above into a single priority rule and then introduce a new dispatching method, which will simultaneously select a task and determine its mode and starting time.

Selecting a task to schedule using a priority rule has two components. The first component evaluates each candidate task with a priority function. The second component is a decision rule that selects a task based on each task's priority function value. For example, in the minimum slack rule, the priority function calculates the slack time of each task and the decision rule is to choose the task that has the minimum value. Thus, integrating two priority rules will require integrating their underlying priority functions, as well as their corresponding decision rules.

We first introduce two new priority functions that are related to those discussed above (minimum slack rule and minimum investment rule) called the *transformed slack priority function* and the *transformed investment priority function*. We then describe the combined priority function, associated decision rule, and new heuristic.

Transformed slack priority function

Before discussing the development of the *transformed slack priority function,* some terminology will be introduced. The *slack time* of task j is defined as the difference between its latest starting time and earliest starting time ($LST_j - EST_j$). A task's LST_j, EST_j, and latest finish time (LFT_j) are calculated by the critical path method (CPM). For multi-mode project scheduling problems, the shortest duration mode of each task is used to derive these values. Additionally, define the threshold finish time (TFT_j) as the latest finish time calculated from CPM by using the longest duration mode of each task (see Figure 1).

The *transformed slack priority function* for task j in mode m starting at time t is defined as

$$v^1(j,m,t) = \begin{cases} \dfrac{\text{Max}(FT(j,\,m,\,t) - TFT_j,\,0)}{LFT_j - TFT_j} & \forall j,\, m,\, t \ni EST_j \leqslant t,\\ & FT(j,\,m,\,t) \leqslant LFT_j \\ \infty & \text{otherwise} \end{cases}$$

where $FT(j,\,m,\,t) = t + d_{jm}$.

$FT(j,\,m,\,t)$ is the finish time of task j, if it starts at period t and is executed in mode m. An explanation of the transformed slack priority function is presented next.

Given a partial schedule PS, for each unscheduled task j that is precedence feasible, the difference between its latest finish time and earliest starting time ($LFT_j - EST_j$) defines the size of the time-window, where task j must be scheduled into (started and finished) to meet the project due date. If the finish time goes beyond LFT_j (ie $LFT_j < FT(j,\,m,\,t)$), the project will not complete by the due date, even if all of task j's successors are executed in their shortest duration modes and resource constraints impose no delays. If this is the case, the *transformed slack priority function* returns a value of infinity. On the other hand, if task j is completed before the latest finish time, the function will return a finite positive value. To determine this function value, we recognize that LFT_j and TFT_j have different scheduling implications. If task j is completed before the threshold finish time (ie $FT(j,\,m,\,t) \leqslant TFT_j$), then its successors have an opportunity to use any of their available modes and still meet the project due date. Any starting time that results with $FT(j,\,m,\,t) \leqslant TFT_j$ is considered equivalent, and the function returns a priority value of zero. If task j is completed after the threshold finish time ($FT(j,\,m,\,t) \geqslant TFT_j$), then its successors are restricted in their available modes to meet the project due date. In this case, the function returns a positive value greater than zero (see Figure 2).

The transformed slack priority function evaluates the impact of a task's finish time on unscheduled tasks. A higher value reflects a 'tighter' subsequent scheduling situation. By dividing the numerator by $LFT_j - TFT_j$, the function is

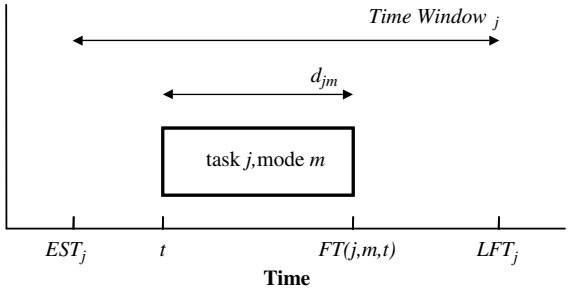

Figure 1 Slack priority function variables.

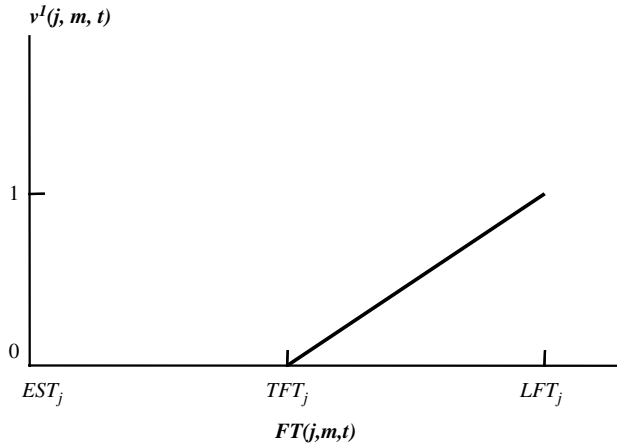

Figure 2 Relationship between FT and its v^1 score.

normalized and generates a dimensionless value between zero and one. The function increases linearly as the finish time increases beyond the threshold finish time, and reaches one when the task is completed at the latest finish time. A lower value reflects a better scheduling alternative.

Transformed investment priority function

Using similar reasoning, we next describe the *transformed investment priority function*. Given a partial schedule *PS*, define R_{rt} as the units of resource r that have been consumed at time period t. Then, define π_{rt} as the corresponding cost of R_{rt} units of resource r at time period t. The investment requirement for resource r (K_r) and the total required investment (Λ) can be expressed as

$$K_r = \operatorname{Max}_t \pi_{rt}$$

$$\Lambda = \sum_r K_r$$

With partial schedule *PS*, define $\alpha(j, m, t)$ as the *extra* investment required, if an unscheduled task j starts at time t with mode m.

$$\alpha(j, m, t) = \sum_r \operatorname{Max}(K_r, c_r \times (k_{jmr} + \operatorname{Max}_{\tau=t}^{t+d_{jm}} R_{r\tau}))$$

The possible extra investment required when scheduling a new task is illustrated in Figure 3. Assuming that a limit on the total resource investment (*IUB*) exists, the transformed investment priority function is defined as

$$v^2(j,m,t) = \begin{cases} \dfrac{\alpha(j, m, t)}{IUB - \Lambda} & \forall j, m, t \quad \ni \Lambda + \alpha(j, m, t) \leqslant IUB \\ \infty & \text{otherwise} \end{cases}$$

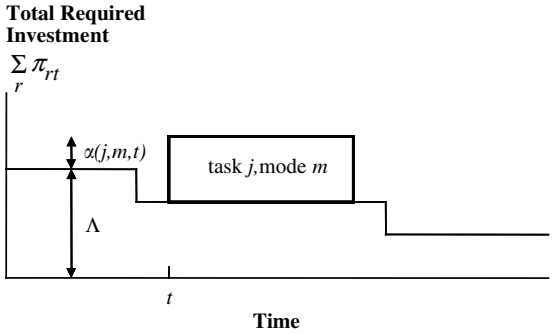

Figure 3 Investment priority function variables.

This function evaluates additional resource investment required when scheduling a task. The function also increases linearly as additional resource investment is required. This function is normalized between zero and one, and is dimensionless. For an unscheduled task, if every mode and starting time requires extra investment beyond the investment limit, no feasible schedule can be generated from the current partial schedule.

Decision rule: determining the best (task, mode, time) *combination*

v^1 and v^2 are combined into a single priority function, which can be used by a decision rule to evaluate all possible task, mode, and time alternatives. There are many ways to combine v^1 and v^2 functions. A simple method is to weight-sum v^1 and v^2:

$$v(j, m, t) = \omega \times v^1(j, m, t) + (1 - \omega) \times v^2(j, m, t) \qquad (6)$$

The decision procedure for selecting the best combination of j, m, and t is:

1. For each unscheduled precedence-feasible task j, select its mode and starting time that give the lowest v. This mode and starting time reflect the best scheduling option, if that task is scheduled.
2. For these tasks with mode and starting time determined, select the task that faces the 'worst' scheduling situation. This is the task j that has the highest v. Among different tasks, it makes intuitive sense to schedule the task that has the most constraints, before its situation becomes worse.

Mathematically, the best task, mode, and time combination (j^*, m^*, t^*) is determined by

$$(j^*, m^*, t^*) = \arg \operatorname{Max}_j (\arg \operatorname{Min}_{m \in M_j, t} v(j, m, t)) \qquad (7)$$

The algorithm

The above priority function (Equation (6)) and the decision rule (Equation (7)) will be applied within a serial schedule

generation scheme. During each iteration a new resource investment limit is determined, and the serial schedule generation scheme is applied. Note that there may be no feasible solution if a tight investment limit is used.

The algorithm starts with a tight (small) resource investment limit and iterates, increasing the investment limit (*IUB*), until a feasible schedule is obtained. This initial investment limit can be a lower bound or any small number. When the algorithm stops, the last complete (feasible) schedule is the solution.

There are J stages in each iteration, and one task is selected and added to the partial schedule at each stage. Three disjoint sets are associated with each stage: tasks that are in the partial schedule are in the *complete set*. The unscheduled precedence feasible tasks, which are the unscheduled tasks with all their predecessors in the complete set, are in the *decision set*. The remaining tasks are in the *remaining set*.

At each stage, among the tasks in the decision set, apply the new priority function and rule to select the best combination of task, mode, and starting time. After this task is scheduled, it is then moved from the decision set to the complete set. This will in turn cause some of the tasks to transfer from the remaining set to the decision set, if they become precedence feasible.

MMRIP Algorithm

Define:

TS	the set of all tasks
$CSet_i$	the complete set at stage i
$DSet_i$	the decision set at stage i
$RSet_i$	the remaining set at stage i
PS_i	a partial schedule at stage i

Step 0: (Initialization)

Initialize *IUB* with a small investment limit

Step 1: (Iteration initialization)

$$i = 1$$
$$RSet_1 = TS$$
$$CSet_1 = DSet_1 = PS_1 = \varnothing$$
$$R_{rt} = 0 \quad \forall r, t$$

Step 2: (Serial schedule generation scheme)

2.1. Move the precedence feasible tasks from $RSet_i$ to $DSet_i$, then

$$RSet_i = TS - CSet_i - DSet_i$$

2.2. Let

$$(j^*, m^*, t^*) = \arg\operatorname*{Max}_{j \in DSet_i} \left(\arg\operatorname*{Min}_{m \in M_j, t} v(j, m, t)\right)$$

(If there is a tie during *Min* or *Max* selection, break the tie with $v^2(j, m, t)$ value)

2.3. If infeasibility is encountered, $IUB = IUB + 1$, goto Step 1

2.4. Schedule task j^* with mode m^* and starting time t^* and update R_{rt}

2.5.

$$CSet_{i+1} = PS_{i+1} = CSet_i \cup \{j^*\}$$

2.6. $i = i + 1$

2.7. If $|PS_i| < J$, repeat Step 2

Step 3: (Final result)

The last (only) PS_{J+1} is the final schedule.

Experimental results

To test the effectiveness of the MMRIP algorithm, which we refer to as the *OneRule* heuristic, problems from the Kolisch problem set are modified to create a test problem set. The issue of what to compare the results against still remains. Many problems in the problem set are too large to solve optimally. Also, there are no other MMRIP heuristics in the literature. However, there are priority rule-based heuristics for the MRCPSP that first apply one rule for task selection, and a second rule for mode selection and starting time. We compare our algorithm against this type of priority rule heuristic, which will be referred to as *TwoRule*.

Design of the comparison heuristic

TwoRule also uses the serial schedule generation scheme to iteratively build a complete schedule. In each iteration a rule is applied to select a task and then another rule is used to determine the mode and starting time to schedule the selected task. The iteration stops when a complete schedule is obtained. The rules used for the experiments are shown in Table 1. For more detailed definition of these rules, refer to Kolisch[4] and Neumann and Zimmermann.[7]

These rules generate 15 possible variations for the *TwoRule* heuristic, as listed in Table 2. We will use the weight factor (w) of 0.5 for the *OneRule* heuristic as the start of the comparison. Our tests investigate the computational efficiency and solution quality of *OneRule* compared to *TwoRule-I* through *TwoRule-XV*.

The problem sets

We employed the standard problem sets C15, C21, J10, J12, J14, J16, J18, J20, and J30 from the Kolisch problem sets for the RCPSP (http://halfrunt.bwl.uni-kiel.de/bwlinstitute/Prod/psplib/data.htm). The sets contain 4950 different

scheduling problem instances with 12–30 tasks, and four resource groups. The non-renewable and doubly constrained resources in each problem are treated as renewable resources.

When using these RCPSP problem instances for the RIP experiments, three different due dates are explored for each problem. These due dates include a project's shortest duration, one-and-half times the shortest duration, and two times the shortest duration. This gives a total of 14 850 problem instances, which are solved using the MMRIP heuristics.

Experimental results

The experimental results are presented as the average percent of additional investment generated by the *TwoRule* heuristic compared to the results obtained using the *OneRule* heuristic. The results are summarized in Table 3.

On average, *OneRule* out-performs all of the *TwoRule* heuristics, with the *TwoRule* heuristics generating solutions with 25–50% more investment required. Among all the *TwoRule* heuristics, *TwoRule-I* (minimum slack rule and the minimum extra investment rule) produces the best results. The *TwoRule* heuristics that use the minimum resource demand rule produce the worst performance. This does not come as a surprise, as the minimum resource demand rule, being a static rule, does not use the information from the partial schedule. (For a detailed definition of static rules and dynamic rules, refer to Kolisch.[4])

Table 4 shows the percentage of problem instances, where *OneRule* out-performs all of the *TwoRule* heuristics (out of 14 850 problems).

The experimental results can be categorized based on the tightness of the project due date used. These results are shown in Tables 5–7. For each level of due date tightness, the average performance of *OneRule* is better than all the *TwoRule* combinations. However, *OneRule* performs relatively better as the due date becomes longer and more task starting time options are available.

Additional experimental results

The effectiveness of the *OneRule* heuristic when compared to the *TwoRule* heuristics is clearly indicated by the experi-

Table 1 Priority rules used for the *TwoRule* heuristic

Priority rule for task selection	*Priority rule for mode and starting time selection*
Minimum slack rule	Minimum extra investment rule
Minimum latest finish time rule	Minimum local investment rule
Minimum latest starting time rule	Minimum resource demand rule
Maximum number successor rule	
Maximum rank positional weight rule	

Table 3 Average percentage increase in total investment generated by the *TwoRule* heuristics (I–XV) when compared to the *OneRule* heuristic solutions

I	*II*	*III*	*IV*	*V*
25.56%	29.41%	50.07%	35.93%	40.82%
VI	*VII*	*VIII*	*IX*	*X*
50.07%	33.00%	36.66%	50.07%	35.49%
XI	*XII*	*XIII*	*XIV*	*XV*
39.51%	50.07%	33.36%	37.68%	50.07%

Table 2 The versions of *Two Rule* heuristic

	Task selection	*Mode and starting time selection*
TwoRule–I	Minimum slack rule	Minimum extra investment rule
TwoRule–II	Minimum slack rule	Minimum local investment rule
TwoRule–III	Minimum slack rule	Minimum resource demand rule
TwoRule–IV	Minimum latest finish time rule	Minimum extra investment rule
TwoRule–V	Minimum latest finish time rule	Minimum local investment rule
TwoRule–VI	Minimum latest finish time rule	Minimum resource demand rule
TwoRule–VII	Minimum latest starting time rule	Minimum extra investment rule
TwoRule–VIII	Minimum latest starting time rule	Minimum local investment rule
TwoRule–IX	Minimum latest starting time rule	Minimum resource demand rule
TwoRule–X	Maximum number successor rule	Minimum extra investment rule
TwoRule–XI	Maximum number successor rule	Minimum local investment rule
TwoRule–XII	Maximum number successor rule	Minimum resource demand rule
TwoRule–XIII	Maximum rank positional weight rule	Minimum extra investment rule
TwoRule–XIV	Maximum rank positional weight rule	Minimum local investment rule
TwoRule–XV	Maximum rank positional weight rule	Minimum resource demand rule

Table 4 The percentage of problems where *OneRule* out-performs *TwoRule*

I	*II*	*III*	*IV*	*V*
88%	91%	99%	93%	95%
VI	*VII*	*VIII*	*IX*	*X*
99%	91%	94%	99%	94%
XI	*XII*	*XIII*	*XIV*	*XV*
96%	99%	94%	95%	99%

Table 5 Average percentage increase in total investment generated by the *TwoRule* heuristics (I–XV) when compared to the *OneRule* heuristic solutions, and using a projects' shortest duration as the due date

I	*II*	*III*	*IV*	*V*
6.66%	8.51%	29.79%	12.61%	15.34%
VI	*VII*	*VIII*	*IX*	*X*
29.79%	10.46%	12.75%	29.79%	14.67%
XI	*XII*	*XIII*	*XIV*	*XV*
16.33%	29.79%	13.33%	15.26%	29.79%

Table 6 Average percentage increase in total investment generated by the *TwoRule* heuristics (I–XV) when compared to the *OneRule* heuristic solutions, and using one-and-half times the projects' shortest duration as the due date

I	*II*	*III*	*IV*	*V*
32.55%	36.40%	57.41%	44.01%	49.14%
VI	*VII*	*VIII*	*IX*	*X*
57.41%	40.63%	44.14%	57.41%	43.20%
XI	*XII*	*XIII*	*XIV*	*XV*
47.21%	57.41%	40.49%	45.09%	57.41%

mental results presented thus far. However, to possibly further improve the *OneRule* heuristic and explore the effect of the weight factor (w), some additional experimentation was performed. For this purpose, we ran the *OneRule* heuristic with several weight settings using the same 14 850 problem instances. We also investigated the impact of multiplying the v^1 and v^2 values to obtain v (instead of using the weight-sum of v^1 and v^2). That is, we used the following in place of Equation (6):

$$v(j, m, t) = v^1(j, m, t) \times v^2(j, m, t) \tag{8}$$

In all, 12 different priority functions (one using Equation (8) and the others differing by the weighting value w) for the *OneRule* heuristic were tested and are listed in Table 8.

Table 7 Average percentage increase in total investment generated by the *TwoRule* heuristics (I–XV) when compared to the *OneRule* heuristic solutions, and using two times the projects' shortest duration as the due date

I	*II*	*III*	*IV*	*V*
37.48%	43.34%	63.02%	51.18%	58.00%
VI	*VII*	*VIII*	*IX*	*X*
63.02%	47.90%	53.10%	63.02%	48.60%
XI	*XII*	*XIII*	*XIV*	*XV*
54.98%	63.02%	46.25%	52.70%	63.02%

Table 8 The versions of *OneRule* heuristic tested with different weighting factors for v^1 and v^2

	Priority function
OneRule–I	$v(j, m, t) = v^1(j, m, t) \times v^2(j, m, t)$
OneRule–II	$v(j, m, t) = 0 \times v^1(j, m, t) + 1 \times v^2(j, m, t)$
OneRule–III	$v(j, m, t) = 0.1 \times v^1(j, m, t) + 0.9 \times v^2(j, m, t)$
OneRule–IV	$v(j, m, t) = 0.2 \times v^1(j, m, t) + 0.8 \times v^2(j, m, t)$
OneRule–V	$v(j, m, t) = 0.3 \times v^1(j, m, t) + 0.7 \times v^2(j, m, t)$
OneRule–VI	$v(j, m, t) = 0.4 \times v^1(j, m, t) + 0.6 \times v^2(j, m, t)$
OneRule–VII	$v(j, m, t) = 0.5 \times v^1(j, m, t) + 0.5 \times v^2(j, m, t)$
OneRule–VIII	$v(j, m, t) = 0.6 \times v^1(j, m, t) + 0.4 \times v^2(j, m, t)$
OneRule–IX	$v(j, m, t) = 0.7 \times v^1(j, m, t) + 0.3 \times v^2(j, m, t)$
OneRule–X	$v(j, m, t) = 0.8 \times v^1(j, m, t) + 0.2 \times v^2(j, m, t)$
OneRule–XI	$v(j, m, t) = 0.9 \times v^1(j, m, t) + 0.1 \times v^2(j, m, t)$
OneRule–XII	$v(j, m, t) = 1 \times v^1(j, m, t) + 0 \times v^2(j, m, t)$

Table 9 Average percentage increase in total investment generated by versions of the *OneRule* heuristic when compared to *OneRule-VII* solutions

I	*II*	*III*	*IV*
23.35%	35.03%	4.05%	0.60%
V	*VI*	*VIII*	*IX*
−0.05%	−0.20%	0.82%	2.02%
X	*XI*	*XII*	
3.29%	4.46%	5.21%	

OneRule-VII (w factor of 0.5) was used as the standard for comparison. The results are summarized in Table 9 where the results are the average percent of additional investment generated by different versions of the *OneRule* heuristic, when compared to *OneRule-VII* (the comparison of *OneRule-VII* to itself is not included).

On average, there is little performance difference among *OneRule-IV* through *VIII*, which represent the weighting factors of 0.2–0.6. This means that the heuristic performs well when a middle value of the weighting factor is used, and that the heuristic is robust with respect to this factor. *OneRule-II*, which considers only resource utilization, produced the worst results. *OneRule-I*, which multiplies v^1 and v^2 to obtain v, also did not perform well. An explanation

Table 10 The percentage of problems where *OneRule-VII* out-performs other versions of the *OneRule* heuristic

I	*II*	*III*	*IV*
91%	95%	70%	67%
V	*VI*	*VIII*	*IX*
69%	77%	79%	76%
X	*XI*	*XII*	
75%	77%	78%	

for this performance is that a perfect score relative to one scheduling aspect (v^1 or v^2 is zero) will turn the entire v score into zero and hide a potentially bad score on the other aspect, which is an undesirable property.

Table 10 shows the percentage of problem instances that *OneRule-VII* (w factor of 0.5) out-performs the other *OneRule* versions (out of 14 850 problems). A weighting factor close to 0.5 gives close to optimal performance for this heuristic.

Conclusions

This paper presents a new heuristic for the MMRIP. The heuristic uses a single decision rule for selecting task, mode, and starting time in a serial schedule generation scheme. Extensive computational experimentation against heuristics using separate common priority rules for task and mode selection is carried out. The experimental results confirm the effectiveness of the new heuristic.

Possible further research efforts could apply the same approach with other priority rule concepts that arise from time-based, resource-based, and network-based perspectives.[4] Additionally, investigation on how these priority functions can be combined may provide further insight into the MMRIP.

References

1 Mohring R (1984). Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Opns Res* **32**: 89–120.
2 Icmeli O (1993). Project scheduling problems: a survey. *Int J Opns Prod Mngt* **13**: 80–91.
3 Sprecher A (1994). *Resource-Constrained Project Scheduling: Exact Methods for the Multi-Mode Case*. Springer-Verlag: Germany.
4 Kolisch R (1995). *Project Scheduling under Resource Constraints*. Physica-Verlag: Germany.
5 Herroelen W, De Reyck B and Demeulemeester E (1998). Resource-constrained project scheduling: a survey of recent developments. *Comput Opns Res* **25**: 279–302.
6 Demeulemeester E (1995). Minimizing resource availability costs in time-limited project networks. *Mngt Sci* **41**: 1590–1598.
7 Neumann K and Zimmermann J (1999). Resource leveling for projects with schedule-dependent time windows. *Eur J Opl Res* **117**: 591–605.
8 Kurtulus I and Davis E (1982). Multi-project scheduling: categorization of heuristic rules performance. *Mngt Sci* **28**: 161–172.
9 Boctor F (1990). Some efficient multi-heuristic procedures for resource-constrained project scheduling. *Eur J Opl Res* **49**: 3–13.
10 Lawrence S and Morton T (1993). Resource-constrained multi-project scheduling with tardy costs: comparing myopic, bottleneck, and resource pricing heuristics. *Eur J Opl Res* **64**: 168–187.
11 Kolisch R (1996). Efficient priority rules for the resource-constrained project scheduling problem. *J Ops Mngt* **14**: 179–192.
12 Kelley JE (1963). *The Critical-Path Method: Resource Planning and Scheduling, Industrial Scheduling*. Prentice-Hall: NJ, pp 347–365.
13 Slowinski R (1981). Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *Eur J Opl Res* **7**: 265–273.
14 Talbot B (1982). Resource-constrained project scheduling with time-resource trade-offs: the non-preemptive case. *Mngt Sci* **28**: 1197–1210.

Received August 2003;
accepted May 2004 after one revision