European Journal of Operational Research 230 (2013) 449–463

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Invited Review

Parallel machine scheduling with additional resources: Notation, classification, models and solution methods

Emrah B. Edis [a,*], Ceyda Oguz [b], Irem Ozkarahan [c]

[a] Celal Bayar University, Department of Industrial Engineering, Muradiye, 45140 Manisa, Turkey
[b] Koç University, College of Engineering, Sariyer, 34450 Istanbul, Turkey
[c] Troy University, Computer Science, P.O. Drawer 4419, Montgomery, AL 36104, USA

ARTICLE INFO

Article history:
Received 31 December 2010
Accepted 22 February 2013
Available online 7 March 2013

Keywords:
Scheduling
Parallel machines
Additional resources
Integer programming

ABSTRACT

Majority of parallel machine scheduling studies consider machine as the only resource. However, in most real-life manufacturing environments, jobs may require additional resources, such as automated guided vehicles, machine operators, tools, pallets, dies, and industrial robots, for their handling and processing. This paper presents a review and discussion of studies on the parallel machine scheduling problems with additional resources. Papers are surveyed in five main categories: machine environment, additional resource, objective functions, complexity results and solution methods, and other important issues. The strengths and weaknesses of the literature together with open areas for future studies are also emphasized. Finally, extensions of integer programming models for two main classes of related problems are given and conclusions are drawn based on computational studies.

1. Introduction

Scheduling models and algorithms are most widely used in manufacturing applications for efficient production. Parallel machine scheduling (PMS) problems are among the most studied areas in scheduling literature. While Cheng and Sin (1990) give a comprehensive analysis of PMS research, Mokotoff (2001) presents a survey for the makespan minimization on identical parallel machines. More recently, Pfund et al. (2004) review the literature related to traditional unrelated parallel machine scheduling problems. In most of the PMS studies, the only resource considered is the machine. However, in most real-life manufacturing environments, jobs also require, beside machines, certain *additional resources*, such as automated guided vehicles, machine operators, tools, pallets, dies and industrial robots, for their handling and processing (Slowinski, 1980; Blazewicz et al., 1983; Ventura and Kim, 2000). Thus, the study of PMS with additional resources is a significant area of research.

Consistent with the definition given by Blazewicz et al. (2004), we call an additional resource *processing resource* if it is required together with a machine (processor) during the processing of a job. Otherwise, i.e., if the resource is required either before or after the processing of a job, then it is called *input–output resource*. The additional resources are further classified with respect to their renewability (*resource constraints*) and divisibility (*resource divisibility*) (Slowinski, 1980; Blazewicz et al., 2007).

From the viewpoint of resource constraints (Slowinski, 1980; Blazewicz et al., 2007):

- A resource is *renewable*, if its only total usage at every moment is constrained. Once it is used for a job, it may be used again for another job.
- A resource is *non-renewable*, if its total consumption is constrained. In other words, once it is used by some job, it cannot be available for any other job.
- A resource is *doubly constrained*, if it is both renewable and non-renewable.

From the viewpoint of *resource divisibility* (Slowinski, 1980; Blazewicz et al., 2007):

- *Discrete resources* can be allocated to jobs in discrete units from a given finite set of possible allocations.
- *Continuous resources* can be allocated to jobs in arbitrary amounts within an interval.

This paper gives a review and discussion of studies related to PMS problems with additional resources, where the additional resources are processing, discrete and renewable. While the problems are investigated with respect to their main characteristics, the strengths and the weaknesses of the literature, open areas and future needs of the studies are also given.

* Corresponding author. Address: Celal Bayar University, Department of Industrial Engineering (Celal Bayar Universitesi Endustri Muhendisligi Bolumu), Muradiye Kampusu, 45140 Manisa, Turkey. Tel.: +90 236 2012207; fax: +90 236 2412143.
E-mail addresses: emrah.edis@cbu.edu.tr (E.B. Edis), coguz@ku.edu.tr (C. Oguz), iozkarahan@troy.edu (I. Ozkarahan).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.02.042

Nomenclature

n	number of jobs
N	set of jobs
m	number of machines
M	set of machines
i, h	indices of jobs, i, $h = 1,\ldots,n$
K_i	the set of possible (resource) modes belonging to job i
k	index of resource modes ($k \in K_i$)
j	index of machines, $j = 1,\ldots,m$
N_j	the set of jobs already assigned to machine j
T	number of time periods in the scheduling horizon
t	index of time periods, $t = 1,\ldots,T$
p_{ij}	processing time of job i on machine j
p_i	processing time of job i (independent of machine j)
$\hat{p}_{ik}$	processing time of job i when processed in mode $k \in K_i$
$\hat{p}_{ijk}$	processing time of job i on machine j when processed in mode $k \in K_i$
b	the size of the single additional resource type
r_i	the earliest time at which job i can start its processing, i.e., release time
d_i	due date of job i
w_i	weight of job i, denoting the importance of job i relative to other jobs
C_i	completion time of job i
$C_{\max}$	maximum completion time of all jobs in the system, i.e., makespan
T_i	tardiness of job i
U_i	equals to 1 if job i is tardy, 0, otherwise
$L_{\max}$	maximum lateness

The topics excluded in this paper can be found in different studies as follows. PMS problems with input/output resources are dealt with in Blazewicz et al. (2004), Hall et al. (2000) and Glass et al. (2000). Comprehensive studies on continuous resources can be found in Jozefowska and Weglarz (2004) and Blazewicz et al. (2007), while an example study related to non-renewable resources is given by Shabtay and Kaspi (2006), and a doubly constrained project scheduling problem is presented by Ozdamar and Ulusoy (1994). PMS problems including precedence constraints and preemptive jobs are considered in Blazewicz et al. (1986b, 2007). On the other hand, surveys on the general area of resource constrained scheduling problems can be found in Blazewicz et al. (2004, 2007).

The rest of the paper is organized as follows. The definitions, the notation and the classification scheme together with the main assumptions are given in the following section. The review of the studies under five different topics is presented in Section 3. The strengths and the weaknesses of the related studies are summarized in Section 4. Some extensions of the mathematical programming models for two main classes of problems are presented and discussed in Section 5. Finally the paper is concluded with Section 6.

2. Definitions, notation, classification and assumptions

The nomenclature given above will be used throughout the paper and additional notation will be introduced when necessary.

To define a resource constrained parallel machine scheduling problem (RCPMSP), consider a set of n jobs to be processed on a set of m unrelated parallel machines with the processing time of job i on machine j, p_{ij}. The jobs, during their processing, may also require a set of resource types $\boldsymbol{R} = \{R_1, R_2, \ldots, R_u\}$, which are available in the amounts of $b_1, b_2, \ldots, b_u$ units, respectively. The string $\boldsymbol{R}(\boldsymbol{i})$ represents the amount of additional resources required by job i, and can be expressed as $\boldsymbol{R}(\boldsymbol{i}) = [R_1(i), R_2(i), \ldots, R_u(i)]$, where $R_v(i) \leqslant b_v,\ v = 1,\ldots,u$, denotes the number of units of resource R_v required by job i. Resource-constrained scheduling problems are difficult because, as well as the efficient allocation of jobs, it is necessary to consider the feasible grouping of simultaneously processed jobs that will use resources within their availability limits at each point in time (Pinto and Grossman, 1997).

In the field of PMS with additional resources, a new class of problems, named parallel machine flexible resource scheduling (PMFRS), was introduced by Daniels et al. (1996). A PMFRS problem is formally defined as follows. A set of n jobs is processed on a set of m machines where assignment of jobs to machines is specified a priori. In addition, processing of each job requires a single renewable scarce resource and the processing time of each job is a non-increasing function of the associated amount of the allocated resource. When compared to the RCPMSP, the distinguishing feature of the PMFRS problem is the dependence of the processing time on the additional resource allocated. Each job i can be processed in any number of modes with K_i representing the set of possible modes for job i. Let $\hat{p}_{ik}$ denote the processing time of job i when processed in mode $k \in K_i$ (Daniels et al., 1996). To achieve processing time $\hat{p}_{ik}$, $k \leqslant b$ units of resources must be allocated to job i for its duration of processing. The PMFRS problem in its original form assumes that the assignment of jobs to machines is pre-specified, which eliminates the job-machine assignment sub-problem (Daniels et al., 1996). If the assignment of jobs to machines is not specified, we face a more general problem, i.e., the unspecified PMFRS (UPMFRS) problem, where an additional job-machine assignment sub-problem is to be solved (Daniels et al., 1999). In the UPMFRS problem, to achieve the processing time $\hat{p}_{ijk}$, job i should be allocated to machine j with $k \leqslant b$ units of resources for its duration of processing.

In this paper, we use the three-field notation, $\alpha|\beta|\gamma$, which is introduced by Graham et al. (1979) and later expanded by Blazewicz et al. (1983) and Kellerer and Strusevisch (2008) for the PMS problems with additional resources. In this notation:

- the first field α specifies the machine environment; P represents identical, Q uniform, and R unrelated parallel machines,
- the second field β denotes the job characteristics, and
- the third field γ denotes the objective function ($\varnothing$ indicates no objective function is considered).

Blazewicz et al. (1983) expand this classification scheme by allowing the use of additional resources and state that $\beta \in \{\varnothing, res\lambda\sigma\delta\}$ characterizes these resources, where:

$\beta = \varnothing$: there are no additional resource constraints,
$\beta = res\lambda\sigma\delta$: there are specified resource constraints, such that:

- If λ is a positive integer, then the *number of resource types* is constant and equal to λ; if $\lambda = \cdot$, then it is part of the input and arbitrary.
- If σ is a positive integer, then all *resource sizes* are constant and equal to σ; if $\sigma = \cdot$, then all resource sizes are arbitrary.
- If δ is a positive integer, then all *resource requirements* have a constant upper bound equal to δ; if $\delta = \cdot$, then no such bounds are specified.

Later, to define the dynamic PMFRS problem, Kellerer and Strusevisch (2008) extend the classification scheme with *Bi*, *Int*, and *Lin* for the β field, all of which refer to the case where processing times are resource-dependent. More specifically:

- "*Bi*" refers to binary resource allocation, i.e., if job $i \in N$ does not receive the resource, its processing time remains equal to p_i, otherwise its processing time becomes p_i - π_i.
- "*Int*" stands for integer resource allocation, i.e., if job $i \in N$ receives τ units of the resource, then its actual processing time is equal to a given $p_{i\tau}$.
- "*Lin*" stresses that the actual processing times depend linearly on the number of units of speeding-up resource allocated to the job, i.e., if job $i \in N$ is given τ units of the resource, then its actual processing time is equal to $p_{i\tau} = p_i - \tau \times \pi_i$.

To identify the static versions of the PMFRS and the UPMFRS problems with this classification scheme, we extend the β field with the term *Stc* which denotes a static resource allocation.

In terms of the classification described above, Kellerer and Strusevisch (2008) define the dynamic PMFRS problem with the binary resource allocation and makespan objective as $PDm|res111, Bi|C_{\max}$. In this representation, *PDm* denotes m parallel dedicated machines, whereas '*res*111' indicates one additional resource type with one unit available and each job being allocated with no more than one unit of the additional resource. Finally, "*Bi*" stresses that the additional resource is speeding-up and binary scenario of its consumption is applied. On the other hand, a static UPMFRS problem with identical machines can be defined as $P|res1\cdot\cdot, Stc|C_{\max}$.

Unless explicitly indicated, we assume throughout this paper that:

i. A job cannot be processed on more than one machine simultaneously.
ii. A machine cannot process more than one job at a time.
iii. No precedence constraints are allowed.
iv. Preemption is not allowed.
v. Job cancellation is not allowed and machines are always available.
vi. Processing times are independent of the schedule.
vii. Jobs are all known in advance and the problem is deterministic.

3. Review of the literature

In this review article, the studies are evaluated with respect to five main topics:

a. Machine environment.
b. Additional resource.
c. Objective functions.
d. Complexity results and solution methods.
e. Other important issues.

The following subsections analyze the papers in the literature and present their weaknesses and strengths based on these five main topics. The explanation of the papers given in each subsection is only related to the corresponding topic, e.g., Section 3.1 only deals with machine characteristics handled in the surveyed papers. Hence, the same papers may be repeated in the subsequent subsections to give a coherent review of the related topics.

3.1. Machine environment

In this subsection, the papers are grouped and presented with respect to the number of machines as well as the characteristics of the machine environment.

In terms of the number of machines, most of the studies deal with more than three machines. As expected, almost all papers concerning two or three machines either prove that the problems are NP-hard or propose polynomial-time exact algorithms (Blazewicz et al., 1983, 1986a, 1987; Garey and Johnson, 1975; Kellerer and Strusevisch, 2003, 2004).

Recall that the classical PMS theory classifies the machine environment into three main classes: identical, uniform and unrelated machines. However, literature related to the PMS problems with additional resources investigates a new category: *parallel dedicated machines* where the set of jobs that will be processed on each machine is pre-determined. This assumption simplifies the problem by eliminating the job-machine assignment sub-problem. Almost one third of the studies surveyed in this article assume that machines are dedicated.

Another widely studied machine environment is the case of identical parallel machines, which eases the design and the implementation of exact and/or approximation algorithms. For instance, Blazewicz (1978), Blazewicz and Ecker (1983), Blazewicz et al. (1987) and Ventura and Kim (2000) propose polynomial-time exact algorithms for the RCPMSPs with identical machines.

Uniform (Kovalyov and Shafransky, 1998; Ruiz-Torres et al., 2007) and unrelated machines (Grigoriev et al., 2005, 2006, 2007; Edis and Oguz, 2011,2012) are rarely studied. However, machine eligibility restrictions may be viewed as a special case of the unrelated parallel machine environment. In case of machine eligibility restrictions, job i is allowed to be processed only on a subset M_i of m machines. Edis et al. (2008) and Edis and Ozkarahan (2012) consider machine eligibility restrictions for the RCPMSPs with unit and arbitrary processing times, respectively. Machine eligibility restrictions are also considered in a related field, that is, PMS with auxiliary equipment of constraints (Tamaki et al., 1993; Chen, 2005; Chen and Wu, 2006). We note that these studies consider only the dies as additional resources which may not be treated as a common shared resource (like machine operators).

3.2. Additional resource

In this subsection, the papers are categorized with respect to the additional resource characteristics. In a significant number of studies surveyed, job processing times are not fixed but based on the amount of the additional resource allocated to it. As already stated, Daniels et al. (1996) refer to this problem as PMFRS. A common characteristic of the studies in this field is to deal with a single additional resource in limited supply. This type of problems should solve an additional *resource allocation* problem since the amount of the additional resource allocated to a job determines its processing time. In PMFRS problems, the resource allocation problem adds a significant complexity to the entire problem. Therefore, most of the studies in this field relax other characteristics of the problem. For example, PMFRS problems in its original form assume that the assignment of the jobs to the machines is pre-specified. This assumption simplifies the problem by eliminating the job-machine assignment sub-problem and the problem becomes equivalent to the case of *parallel dedicated machines* defined in Section 3.1.

Another simplifying assumption arises with the restricted use of the additional resources through the machines. A number of studies assume that the flexible resource may be allocated freely among the machines, but the resulting resource assignment must remain fixed for the whole scheduling horizon (Daniels et al., 1999; Ruiz-Torres and Centeno, 2007; Ruiz-Torres et al., 2007; Sue and Lien, 2009). Daniels et al. (1996) refer to this case as *static problem*. In this case, the resources are inevitably non-renewable and there is no competition for the additional resource across machines (Olafsson and Shi, 2000).

Recall that $\hat{p}_{ik}$ denotes the processing time of job i when processed in mode $k \in K_i$. Daniels et al. (1996) present the formulation of the static PMFRS problem with the makespan objective, i.e., $PD|res1\cdot\cdot, Stc|C_{\max}$ where decision variable $x_{ik} = 1$ if job i is pro-

cessed in mode k, and $x_{ik} = 0$ otherwise. They also propose an exact algorithm for this problem. First, a minimum amount of the resource is assigned to each machine to process each job with the slowest processing time. Next, the machine that determines the current makespan is identified, say machine j. If sufficient additional resource remains, then the amount of the resource assigned to machine j is increased by one unit, and the amount of the remaining resource is decreased accordingly. This process is repeated until the additional resource is exhausted.

Daniels et al. (1999) extend the formulation of the static PMFRS problem for the case where the job assignment to the machines is unspecified. They refer to this problem as static UPMFRS problem and state that this problem is NP-hard. For the static UPMFRS problem with identical machines and makespan objective, i.e., $P|res1\cdot\cdot, Stc|C_{\max}$, they additionally define a set of binary decision variables y_{ij} where $y_{ij} = 1$ if job i is assigned to machine j, and $y_{ij} = 0$ otherwise. The objective function is then defined as:

$$\text{Minimize } C_{\max} = \max_{j=1,\ldots,m} \sum_{i=1}^{n} \sum_{k\in K_i} \hat{p}_{ik} x_{ik} y_{ij}. \tag{1}$$

They also add a constraint set which guarantees that each job is assigned to only one machine. Notice that this formulation has a nonlinear objective function.

The study of the dynamic case, where the flexible resource can switch between the machines during the schedule, is more realistic. The dynamic PMFRS problem includes three sub-problems. First, a dynamic resource allocation must be determined. Next, a sequence of jobs on each machine must be determined, and then a starting time for each job has to be found (Daniels et al., 1996). To formulate the dynamic version of the PMFRS problem the following decision variables are defined (Daniels et al., 1996):

$$y_{ih} = \begin{cases} 1, & \text{if job } i \text{ precedes job } h, \text{ where } i,h \in N_j, \\ 0, & \text{otherwise}, \end{cases}$$

$$x_{ikt} = \begin{cases} 1, & \text{if job } i \text{ completes its processing with } k \\ & \quad \text{units of resource at time } t, \\ 0, & \text{otherwise}. \end{cases}$$

Let T be an upper bound on the makespan. Daniels et al. (1996) give the formulation of the dynamic PMFRS problem, i.e., $PD|res1\cdot\cdot, Int|C_{\max}$ as follows:

$$\text{Minimize} \quad C_{\max} = \max_{i=1,\ldots,n} C_i, \tag{2}$$

$$\text{subject to :} \quad \sum_{k\in K_i} \hat{p}_{ik} \sum_{t=1}^{T} x_{ikt} = p_i \quad i = 1,\ldots,n, \tag{3}$$

$$\sum_{t=1}^{T} t \sum_{k\in K_i} x_{ikt} = C_i \quad i = 1,\ldots,n, \tag{4}$$

$$\sum_{k\in K_i} \sum_{t=1}^{T} x_{ikt} = 1 \quad i = 1,\ldots,n, \tag{5}$$

$$C_h - C_i + T(1 - y_{ih}) \geqslant p_h \quad i \in N_j,\ h \in N_j \setminus \{i\},\ j = 1,\ldots,m, \tag{6}$$

$$y_{ih} + y_{hi} = 1 \quad i \in N_j,\ h \in N_j \setminus \{1,2,\ldots,i\},\ j \in M, \tag{7}$$

$$\sum_{i=1}^{n} \sum_{k\in K_i} \sum_{l=t}^{t+\hat{p}_{ik}-1} x_{ikl} k \leqslant b \quad t = 1,\ldots,T, \tag{8}$$

$$x_{ikt} \in \{0,1\} \quad i = 1,\ldots,n,\ k \in K_i,\ t = 1,\ldots,T, \tag{9}$$

$$y_{ih} \in \{0,1\} \quad i \in N_j, h \in N_j \setminus \{i\},\ j \in M, \tag{10}$$

$$C_i \geqslant 0 \quad i = 1,\ldots,n. \tag{11}$$

The objective function (2) minimizes the makespan over the jobs. Constraint sets (3) and (4) determine the actual processing time and the completion time of each job, respectively. Constraint set (5) guarantees that each job receives a fixed amount of the additional resource and completes at a unique time. Constraint sets (6) and (7) ensure that, on each machine j, either job $i \in N_j$ precedes job $h \in N_j$, or vice versa. Constraint set (8) states that the total amount of the additional resource consumed at each time should not exceed the available amount, b. Finally, constraint sets (9)–(11) define the domain of the decision variables.

Daniels et al. (1996, 1997) compare the optimal solutions to the static and the dynamic PMFRS problems and report significant improvements in the makespan performance when a flexible resource is dynamically allocated across the machines.

Recall that the dynamic PMFRS problem still assumes that the set of jobs to be processed on each machine is pre-specified. A more general version of this problem is the dynamic UPMFRS problem which also includes the job-machine assignment sub-problem.

Daniels et al. (1996) classify RCPMSP as a special case of the dynamic PMFRS problem in which the processing times and the resource requirements are given for each job. However, the PMFRS problem in its original form assumes that the machines are dedicated and there is only one additional resource type in limited supply. Therefore, the RCPMSP, in our opinion, may cover a more general field since it may also consider the job-machine assignment sub-problem and may allow more than one additional resource type. It only excludes the resource allocation problem. In the light of this discussion, the RCPMSP seems to be closer to the dynamic UPMFRS problem, which incorporates the job-machine assignment sub-problem.

A general RCPMSP with unrelated machines, u number of additional resource types and makespan objective function, i.e., $R|res\cdots|C_{\max}$ could then be formulated as follows:

$$x_{ijt} = \begin{cases} 1, & \text{if job } i \text{ completes its processing on machine} \\ & \quad j \text{ at time } t, \\ 0, & \text{otherwise}, \end{cases}$$

$$\text{Minimize} \quad C_{\max} = \max_{i=1,\ldots,n} \sum_{j=1}^{m} \sum_{t=p_{ij}}^{T} x_{ijt} t, \tag{12}$$

$$\text{subject to :} \quad \sum_{i=1}^{n} \sum_{s=t}^{t+p_{ij}-1} x_{ijs} \leqslant 1 \quad j = 1,\ldots,m,\ t = 1,\ldots,T, \tag{13}$$

$$\sum_{j=1}^{m} \sum_{t=p_{ij}}^{T} x_{ijt} = 1 \quad i = 1,\ldots,n, \tag{14}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{s=t}^{t+p_{ij}-1} R_v(i) x_{ijs} \leqslant b_v \quad v = 1,\ldots,u,\ t = 1,\ldots,T, \tag{15}$$

$$x_{ijt} \in \{0,1\} \quad i = 1,\ldots,n,\ j = 1,\ldots,m,\ t = 1,\ldots,T. \tag{16}$$

This integer programming (IP) model aims to minimize the makespan as defined in (12). Constraint set (13) makes sure that no more than one job can be assigned to any machine at any time period. Constraint set (14) ensures that each job should certainly be processed on a machine. Constraint set (15) states that, for each additional resource type, the total amount assigned to jobs at any time period is less than or equal to the available amount of each resource type b_v. Finally, constraint set (16) indicates that all x_{ijt} are binary variables.

The complexity of the RCPMSP grows with the number of the additional resource types. Therefore, most of the studies relevant to the RCPMSP consider single additional resource type. Moreover, the studies that deal with more than one additional resource type consider

- unit (equal) processing times (Garey and Johnson, 1975; Blazewicz et al., 1983, 1993; Blazewicz and Ecker, 1983; Srivastav and Stangier, 1997; Ventura and Kim, 2003) and/or
- two or three machines (Garey and Johnson, 1975; Blazewicz et al., 1993), or
- unit size of additional resources and 0/1 requirements (Kellerer and Strusevisch, 2004).

Another simplifying issue in case of more than one additional resource type is to restrict the RCPMSP with a fixed number R of task types (or classes). That is, the tuple $(p_i, R_1(i), R_2(i), \ldots, R_u(i))$ denotes the type of task i, and the set of tasks is divided into a fixed number of classes and all the tasks belonging to the same class have the same processing time and the same resource requirements. This assumption is specified as 'types = R' in the β field of the classification scheme (Brucker and Kramer, 1996).

To summarize the problem types studied in this survey paper, we first identify the following problem characteristics that determine each problem type, and then illustrate the six main problem types in Fig. 1:

- Number of additional resource types
  - One additional resource type (*one*).
  - One or more additional resource type(s) (*one*/*multi*).
- Effect of the additional resources on the processing times
  - Processing time reduces with the increasing amount of the additional resources allocated (*speeding-up*).
  - Resource requirements of the jobs are fixed and known a priori (*fix*).
- Allocation of resources
  - Each machine can only use a fixed amount of static additional resource, i.e., the additional resource cannot switch across other machines during the schedule (*static*).
  - The additional resource can switch among machines during the schedule (*dynamic*).
- Job-machine assignment
  - Assignment of the jobs to the machines is pre-specified (*specified-dedicated*).
  - Assignment of the jobs to the machines is unspecified (*unspecified*).

3.3. Objective function(s)

Except a few, all studies surveyed in this paper aim to minimize the makespan as the objective, which can be attributed to the following reasons:

- The makespan objective is widely used in the PMS studies since it balances the load between the machines and provides a high utilization of the machines (Pinedo, 2008).
- It is easier to handle the makespan objective when compared to other criteria such as due date based objectives.
- There exist a number of efficient solution algorithms, e.g., the longest processing time first (LPT) rule, to minimize makespan for the classical PMS problems and they are utilized in designing approximation approaches for the PMS problems with additional resources.

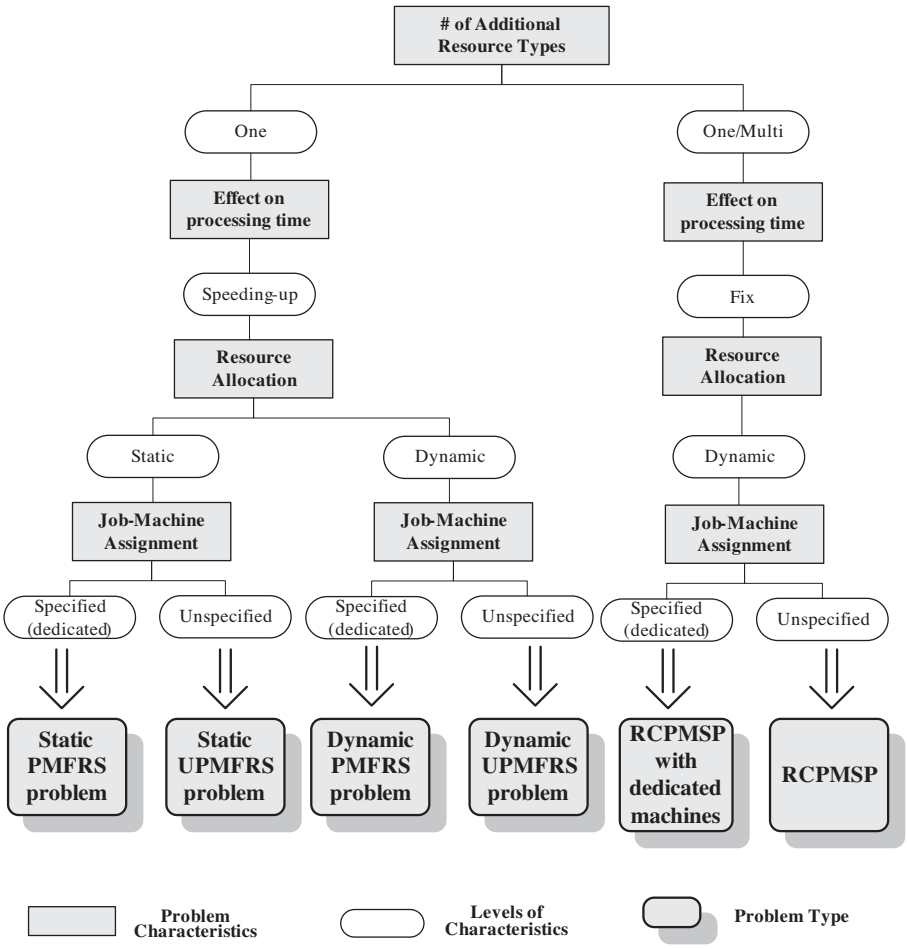

Fig. 1. Characteristics of the problem types.

– Developing lower bounds for the makespan objective is rather straightforward in comparison to other performance criteria.

Other objective functions considered are the lateness (Blazewicz et al., 1986a, 1993); the number of tardy jobs (Ruiz-Torres et al., 2007); the total absolute deviation either from common due dates, i.e., *TAD* (Ventura and Kim, 2000) or from arbitrary due dates, i.e., *TADD* (Ventura and Kim, 2003) and the total flow time (Blazewicz et al., 1987; Edis et al., 2008; Edis and Oguz, 2011).

We further note that all papers deal with a single objective function since the complex nature of the problems prevents the development of solution procedures for multiple criteria.

3.4. Complexity results and solution methods

Blazewicz et al. (1983, 2007) outline a range of initial results on the complexity of the resource constrained scheduling problems and classify the resource constraints into six different types. Fig. 2 illustrates these six types and the simple transformations between them in terms of $res\lambda\sigma\delta$ classification. An arc from type (a) to type (b) indicates that (a) is a special case of (b). Hence, the most general version of the resource constraints is '$res\cdots$'. Among the transformations given in Fig. 2, $(res1\cdot\cdot) \rightarrow (res\cdot 11)$ has been proved in Blazewicz et al. (1986b) and the others are trivial (Blazewicz et al., 2007).

Solution approaches related to the PMS problems with additional resources can be classified under three headings: polynomial-time algorithms, exact approaches and approximation/heuristic approaches. This classification is illustrated in Fig. 3 and detailed in subsequent sections based on the surveyed papers.

3.4.1. Polynomially solvable problems

Recall that most of the PMS problems with additional resources are NP-hard (Blazewicz et al., 1983). However, for a number of specialized cases, researchers have developed polynomial-time exact algorithms. Table 1 lists these problems in the chronological order with complexity functions of the proposed algorithms.

Garey and Johnson (1975) propose a polynomial-time algorithm for the $P2|res\cdots, p_i = 1|C_{max}$ problem with an idea of establishing the correspondence between the maximum matching in a graph displaying the resource constraints and the minimum length schedule.

An optimal schedule (even for arbitrary number of machines and arbitrary release times) exists for the RCPMSPs with identical machines, one additional resource type and 0/1 resource requirements, when all jobs requiring one unit of the additional resource are assigned to the first b (where $b \leqslant m$) machines, where the additional resource is available in b units (see Blazewicz, 1978; Blazewicz et al., 1987; Ventura and Kim, 2000, 2003). With this allocation, the sub-problem becomes a transportation problem and can be solved in polynomial time.

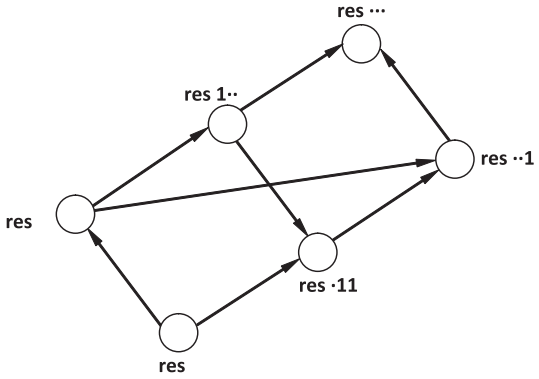

Fig. 2. Reductions between types of resource constraints (adapted from Blazewicz et al., 2007).

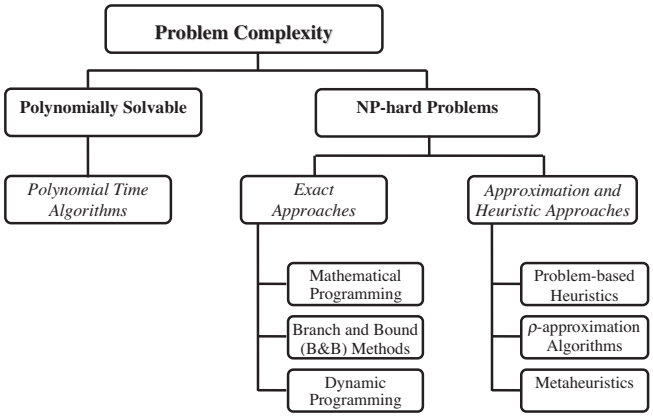

Fig. 3. Classification of solution approaches.

Blazewicz (1979) develops an $O(n^2)$ polynomial-time solution procedure for the $P|res1\cdot 1, p_i = 1, r_i, d_i|\varnothing$ problem, where the term $\varnothing$ indicates that no objective function is considered. The proposed procedure first determines the modified deadlines, and then, by a constructive algorithm, schedules the jobs so that each job meets its modified deadline. Blazewicz et al. (1983) give a polynomial-time algorithm for solving the $Q2|res1\cdot\cdot, p_i = 1|C_{max}$ problem optimally. Suppose that the speed of the first machine is greater than or equal to the speed of the second one ($v_1 \geqslant v_2$). The algorithm, at first, schedules all jobs on the first machine in non-increasing order of the resource requirements. Next, it consecutively removes the last job from the first machine and schedules it as early as possible on the second machine as long as the value of C_{max} is reduced. Blazewicz and Ecker (1983) state that a generalized version of this problem, i.e., $P|res\lambda\sigma\delta, p_i = 1|C_{max}$, may be identified as determining a partition of job set N into concurrently executable subsets where the number of subsets is to be minimized. Therefore, they state that this problem is equivalent to λ-dimensional, σ-restricted bin packing problem which can be solved in $O(\log n)$ time.

For two problems with fixed number of task types, i.e., $P|res\cdots$, types = R, $p_i \leqslant p|C_{max}$ and $Pm|res\cdots$, types = $R|C_{max}$, Blazewicz et al. (1989) develop dynamic programming based polynomial-time algorithms. Later, Brucker and Kramer (1996) present a polynomial-time algorithm for the $P|res\cdots$, types = R, $p_i \leqslant p|\sum w_i C_i$ and the $P|res\cdots$, types = R, $p_i \leqslant p|\sum T_i$ problems by reducing them to a shortest path problem in a suitable network. They also show that, even in the presence of release dates, the problem remains polynomially solvable for the objective functions of makespan, total completion time, and weighted number of tardy jobs.

Kovalyov and Shafransky (1998) present an $O(m\log m)$ algorithm for the $Q|res1\cdot 1, p_i = 1, nmit|C_{max}$ problem, where the case of no machine idle times ($nmit$) ensures that no machine may stand idle unless all jobs allocated to this machine have completed their processing. The idea behind the algorithm is that all resource jobs (i.e., jobs requiring the additional resource) are assigned to the fastest σ machines while the remaining non-resource jobs are allocated to $(m - \sigma)$ slower machines. Recall that σ is part of the input and denotes the size of the additional resource. If the resulting makespan of the schedule on the faster machines is greater than or equal to the resulting makespan on the slower machines, the combination of these two schedules gives the optimal schedule; otherwise, a different algorithm that considers all jobs together is applied to obtain the optimal schedule. Kovalyov and Shafransky

Table 1
Polynomially solvable problems.

Reference	Problem classification	Algorithm complexity
Garey and Johnson (1975)	$P2\|res\cdots, p_i = 1\|C_{max}$	$O(n^3)$
Blazewicz (1978)	$P\|res1\cdot1, r_i, p_i = 1\|C_{max}$	$O(n)$
Blazewicz (1979)	$P\|res1\cdot1, p_i = 1, r_i, d_i\|\varnothing$	$O(n^2)$
Blazewicz et al. (1983)	$Q2\|res1\cdot\cdot, p_i = 1\|C_{max}$	$O(n\log n)$
Blazewicz and Ecker (1983)	$P\|res\lambda\sigma\delta, p_i = 1\|C_{max}$	$O(\log n)$
Blazewicz et al. (1987)	$P\|res1\cdot1\|\sum C_i$	$O(n^3)$
Blazewicz et al. (1989)	$P\|res\cdots$, types = R, $p_i \leqslant p\|C_{max}$	$O(n^{R(p+1)})$
	$Pm\|res\cdots$, types = $R\|C_{max}$	$O(n^{R(m+1)})$
Brucker and Kramer (1996)	$P\|res\cdots$, types = R, $p_i \leqslant p\|\sum w_iC_i, \sum T_i$	$O(R(p+u)n^{Rp} + R^2pn^{R(p+2)})$
	$P\|res\cdots$, types = R, $p_i \leqslant p$, $r_i\|C_{max}, \sum C_i$	$O(R(p+u)n^{Rp} + R^2pn^{R(p+2)+1})$
	$P\|res\cdots$, types = R, $p_i \leqslant p\|\sum w_iU_i$	$O(R(p+u)n^{Rp} + R^2pn^{R(p+2)})$
	$Pm\|res\cdots$, types = $R\|\sum w_iC_i, \sum w_iU_i, \sum T_i$	$O(R^m\ mu + m(m+R)R^{m+1}\ n^{R(m+1)})$
	$Pm\|res\cdots$, types = R, $r_i\|C_{max}, \sum C_i$	$O(R^m\ mu + m(m+R)R^{m+1}\ n^{R(m+1)+m})$
Daniels et al. (1996)	$PD2\|res1\cdot\cdot\|C_{max}$	$O(n\log n)$
	$PD\|res1\cdot\cdot, Stc\|C_{max}$ (Static PMFRS)	$O(nb(n+m))$
Kovalyov and Shafransky (1998)	$P\|res1\cdot1, p_i = 1\|C_{max}$	$O(1)$
	$Q\|res1\cdot1, p_i = 1, nmit\|C_{max}$	$O(m\log m)$
Ventura and Kim (2000)	$P\|res1\cdot1, p_i, d_i = d\|TAD$	$O(n^4)$
Ventura and Kim (2003)	$P\|res1\cdot1, r_i, p_i = 1, d_i\|TADD$	$O(n^4)$
Kellerer and Strusevisch (2003)	$PD2\|res111\|C_{max}$	$O(n)$
Kellerer and Strusevisch (2004)	$PD2\|res1\cdot\cdot\|C_{max}$	$O(n\log n)$
	$PD2\|res211\|C_{max}$	$O(n)$

(1998) also state that the given algorithm can be extended to other objective functions. Finally, they show that a modification of this algorithm runs in O(1) time complexity for the identical machine case.

As stated in Section 3.2, Daniels et al. (1996) propose a polynomial-time algorithm which runs in $O(nb(n+m))$ time complexity for the static PMFRS problem with the makespan objective, i.e., $PD|res1\cdot\cdot, Stc|C_{max}$. They also propose a polynomial-time algorithm for the $PD2|res1\cdot\cdot|C_{max}$ problem. The algorithm determines the sequence of jobs on each machine such that the jobs in N_1 and N_2 are in non-increasing and non-decreasing order of resource requirements, respectively. A schedule on the first machine is constructed without idle time, and then the jobs on the second machine are scheduled at the earliest time allowed by the resource feasibility. The algorithm runs in $O(n\log n)$ time complexity. Kellerer and Strusevisch (2003, 2004) develop polynomial-time algorithms for various versions of the problems with two parallel dedicated machines. For the problem of $PD2|res111|C_{max}$, Kellerer and Strusevisch (2003) develop a group technology-based polynomial-time algorithm where each subset of jobs assigned to each machine j (i.e., N_j) is split into further two subsets: the subset of non-resource jobs Q_j, and the subset of resource jobs R_j. All jobs of these subsets are considered as a single batch. Notice that the batches composed of non-resource jobs may overlap with any other batch; while the batches composed of resource jobs cannot be processed simultaneously with each other. The algorithm assigns the batch composed of R_1 jobs and then the one with Q_1 jobs to the first machine consecutively without intermediate idle time. Then, it starts the batch of Q_2 jobs on the second machine at time zero, and finally it starts the batch of R_2 jobs on the second machine as early as possible. The resulting makespan is optimal. Kellerer and Strusevisch (2004) develop polynomial-time algorithms for two extensions of this problem. First, they develop a polynomial-time algorithm for the $PD2|res1\cdot\cdot|C_{max}$ problem by using machine and resource based lower bounds differing from Daniels et al. (1996). They first partition the subset of jobs for each machine (i.e., N_j) into further $(\sigma+1)$ subsets where each subset contains all jobs with the same resource requirements $(r = 0, 1, \ldots, \sigma)$. Then, subsets of the first machine are allocated successively, beginning from the ones with no-resource requirements $(r = 0)$ and ending with the ones with σ resource requirements. Accordingly, the subsets of the second machine are assigned in the reverse order. Finally, the start times of the subsets are modified to eliminate the overlaps by utilizing the derived lower bounds. In the same paper, the authors extend the $PD2|res111|C_{max}$ problem discussed in Kellerer and Strusevisch (2003) to include two additional resource types. Resource based lower bounds are calculated based on the subsets of resource and non-resource jobs with respect to additional resource types. Then the subsets are ordered in a way that will give the optimal schedule.

The above review shows that a considerable amount of research has been devoted to find efficient polynomial-time algorithms. However, many other problems are identified as NP-hard and the following section presents and discusses the related NP-hard problems.

3.4.2. NP-hard problems

Other than the problems listed in Table 1, almost all cases related to the PMS problems with additional resources are NP-hard (Blazewicz et al., 1983). Table 2 lists the NP-hard problems proved in the literature. According to the complexity hierarchy given in Pinedo (2008, p.26) and reductions between types of resource constraints given in Fig. 2, more complex versions of problems are also NP-hard. For instance, since the $Q2|res1\cdot\cdot, p_i = 1|C_{max}$ problem is NP-hard, $Q2|res\cdot11, p_i = 1|C_{max}$ and $Q2|res1\cdot\cdot, p_i = 1|L_{max}$ are also NP-hard.

Table 2
NP-hard problems proved in the literature.

Reference	Problem classification
Garey and Johnson (1975)	$P3\|res1\cdot\cdot, r_i, p_i = 1\|C_{max}$
Blazewicz (1981)	$P\|res\cdots, p_i = 1, r_i, d_i\|\varnothing$
Blazewicz et al. (1983)	$P3\|res\cdot11, p_i = 1\|C_{max}$
	$Q2\|res\cdot11, p_i = 1\|C_{max}$
Blazewicz et al. (1986a)	$P2\|res1\cdot\cdot, p_i = 1\|L_{max}$
	$P2\|res\cdot11, p_i = 1\|L_{max}$
Blazewicz et al. (1986b)	$P2\|res1\cdot\cdot, r_i, p_i = 1\|C_{max}$
Blazewicz et al. (1987)	$P2\|res1\cdot\cdot\|\sum C_i$
	$P2\|res\cdot\cdot1\|\sum C_i$
Kellerer and Strusevisch (2003)	$PD3\|res111\|C_{max}$
	$PD\|res111\|C_{max}$
Kellerer and Strusevisch (2004)	$PD2\|res222\|C_{max}$
	$PD2\|res311\|C_{max}$
Kellerer and Strusevisch (2008)	$PD2\|res111, Bi\|C_{max}$ (PMFRS)

Let us discuss the parameters that affect the complexity of the problem. Firstly, non-identical ready times cause strong NP-hardness of the problem (Blazewicz et al., 2007). Although $P2|res\cdot\cdot\cdot$, $p_i = 1|C_{max}$ can be solved in polynomial time (see Table 1), $P2|res1\cdot\cdot$, r_i, $p_i = 1|C_{max}$ (even with the presence of one additional resource type) is strongly NP-hard. Secondly, increasing the number of machines from two to three causes a strong NP-hardness of the problem (Blazewicz et al., 2007). For instance, while $P2|res1\cdot\cdot$, $p_i = 1|C_{max}$ can be solved in polynomial time, $P3|res1\cdot\cdot$, $p_i = 1|C_{max}$ is NP-hard. Finally, increasing the number of additional resource types makes the problems NP-hard, that is, while $PD2|res111|C_{max}$ can be solved in $O(n)$ time, $PD2|res311|C_{max}$ is NP-hard.

For solving related NP-hard problems, the research focuses on either optimally solving some special and more tractable cases with exact methods, or developing approximate algorithms to find near-optimal solutions. Table 3 presents a taxonomy of both exact and approximation solution procedures with respect to each study. The following two subsections review these studies based on specific solution methods by focusing on the above two aspects.

3.4.3. Exact methods

Exact algorithms, such as branch-and-bound (B&B) and dynamic programming (DP) in the literature are relatively few. Blazewicz et al. (1993) implement a depth-first B&B algorithm for the $P2|res\cdot\cdot\cdot$, $p_i = 1|L_{max}$ problem. At first, jobs are ordered according to the earliest due date (EDD) rule and then decision nodes are generated by assigning a job to one of the two machines step by step. A significant number of decision nodes are eliminated by embedding the investigated lower bounds and dominance relations into each node. The algorithm finds optimal results for the problems with up to 1000 jobs for one additional resource type and up to 100 jobs for more than one resource type in less than 10 seconds.

Daniels et al. (1996) also choose B&B to solve the dynamic PMFRS problem. Since the amount of the additional resource allocated to a job determines its processing time, they create a subproblem series of RCPMSP, in which, the processing times and the resource requirements are specified. For each sub-problem, lower bounds on the makespan are calculated and resource-allocation policies that cannot give better solutions are fathomed. For each candidate resource-allocation policy, the algorithm proceeds to a lower level search where job sequencing and start time decisions are considered simultaneously. Within a maximum computation time of 8 hours, this B&B algorithm is able to solve the problems with up to 15 jobs and four machines. Kellerer and Strusevisch (2008) propose a dynamic programming algorithm for the $PD2|res111$, $Bi|C_{max}$ problem, which solves two knapsack sub-problems formulated for each machine. For each sub-problem, they try to find which jobs should be resource jobs (i.e., the jobs with assigned resource), and which jobs should be non-resource jobs and solve all-capacities knapsack problem with all positive integer right-hand side values up to a given time bound. They finally obtain the overall optimal solution with a simple procedure by comparing the alternative solutions.

Table 3
A taxonomy of problems based on exact, approximation and heuristic solution procedures.

Reference	Problem classification	Solution method
Krause et al. (1973)	$P\|res1\cdot\cdot, p_i = 1\|C_{max}$	Polynomial-time (4/3)-approximation algorithm
Blazewicz et al. (1993)	$P2\|res1\cdot\cdot, p_i = 1\|L_{max}$	Problem-based heuristic algorithms
	$P2\|res\cdot\cdot\cdot, p_i = 1\|L_{max}$	A branch and bound algorithm
Daniels et al. (1996)	$PD\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic PMFRS)	A branch and bound algorithm
		A static-based heuristic (SBH)
Daniels et al. (1997)	$PD\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic PMFRS)	A tabu search heuristic
		A static-based tabu search heuristic
Srivastav and Stangier (1997)	$P\|res\cdot\cdot1, r_i, p_i = 1\|C_{max}$	A polynomial-time approximation algorithm
Daniels et al. (1999)	$P\|res1\cdot\cdot, Stc\|C_{max}$ (Static UPMFRS)	A decomposition heuristic (using SBH)
		A tabu search heuristic based on SBH
Olafsson and Shi (2000)	$PD\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic PMFRS)	A heuristic, named Nested Partitions (NP)
Kellerer and Strusevisch (2003)	$PDm\|res111\|C_{max}$	$O(mn)$ Group Technology approximation alg.
	$PD3\|res111\|C_{max}$	$O(n)$ heuristic approximation algorithm
	$PD4\|res111\|C_{max}$	$O(n)$ heuristic approximation algorithm
	$PDm\|res111\|C_{max}$	PTAS
Li et al. (2003)	$PD\|res1\cdot\cdot\|C_{max}$	A genetic algorithm (GA)
Ventura and Kim (2003)	$P\|res\cdot\cdot\cdot, r_i, p_i = 1, d_i\|TADD$	Lagrangian-based heuristic algorithm
Kellerer and Strusevisch (2004)	$PD\|res\lambda11\|C_{max}$	An $O(nm \min\{n, m\})$ greedy approximation alg.
	$PDm\|res\lambda11\|C_{max}$	PTAS
Grigoriev et al. (2005)	$R\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic UPMFRS)	$(4 + 2\sqrt{2})$-approximation algorithm
	$PD\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic PMFRS)	$(3 + 2\sqrt{2})$-approximation algorithm
Kumar et al. (2005)	$R\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic UPMFRS)	4-approximation algorithm
Grigoriev and Uetz (2006)	$PD\|res1\cdot\cdot, Lin\|C_{max}$ (Dynamic PMFRS)	A quadratic IP-based $(3 + \varepsilon)$-approximation algorithm
Grigoriev et al. (2006)	$R\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic UPMFRS)	3.75-approximation algorithm
Ruiz-Torres and Centeno (2007)	$P\|res1\cdot\cdot, Stc\|C_{max}$ (Static UPMFRS)	Problem-based heuristic algorithms
Ruiz-Torres et al. (2007)	$Q\|res1\cdot\cdot, Stc\|\sum U_i$ (Static UPMFRS)	Five different problem-based heuristic algorithms
Edis et al. (2008)	$P\|res1\cdot2, M_i, p_i = 1\|\sum C_i$	Lagrangian and problem-based heuristic alg.
Kellerer (2008)	$P\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic UPMFRS)	$(3.5 + \varepsilon)$-approximation algorithm
Kellerer and Strusevisch (2008)	$PD2\|res111, Bi\|C_{max}$ (Dynamic PMFRS)	A pseudo-polynomial-time dynamic prog. alg.
	$PD2\|res111, Bi\|C_{max}$ (Dynamic PMFRS)	FPTAS
	$PD\|res111, Bi\|C_{max}$ (Dynamic PMFRS)	Polynomial-time (3/2)-approximation algorithm
	$PD\|res1\sigma\sigma, Int\|C_{max}$ (Dynamic PMFRS)	Polynomial-time $(3 + \varepsilon)$-approximation alg.
	$PDm\|res111, Bi\|C_{max}$ (Dynamic PMFRS)	PTAS
Grigoriev and Uetz (2009)	$PD\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic PMFRS)	A (nonlinear) IP-based $(3 + \varepsilon)$-approximation alg.
Sue and Lien (2009)	$P\|res1\cdot\cdot, Stc\|C_{max}$ (Static UPMFRS)	Problem-based heuristic algorithms
Edis and Ozkarahan (2011)	$P\|res1\cdot2, M_i\|C_{max}$	IP, CP and IP/CP combined approach
Edis and Oguz (2011)	$R\|res1\cdot\cdot\|\sum C_i$	IP, CP, Lagrangian-based CP approach
Xu et al. (2011)	$P2\|res111, Bi\|C_{max}$ (Dynamic UPMFRS)	FPTAS
Edis and Ozkarahan (2012)	$P36\|res1\cdot2, M_i\|C_{max}$	IP/IP and IP/CP sequential approaches
Edis and Oguz (2012)	$PD\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic PMFRS)	IP/CP sequential approach
	$R\|res1\cdot\cdot, Int\|C_{max}$ (Dynamic UPMFRS)	IP/CP sequential approach

3.4.4. Approximation and heuristic algorithms

Given that many of the PMS problems with additional resources are NP-hard, it becomes necessary to solve these problems approximately rather than optimally. In the scheduling literature, there are two approaches to evaluate such approximations. The first approach is to evaluate their performance by calculating the worst-case bound of the algorithms in a theoretical manner. The second approach is to make experimental studies on proposed algorithms through a series of test problems and to find the mean performances with respect to each other, or to the lower bounds developed, or to the optimal solutions (if any).

In the literature, there exist three well-defined simple heuristic rules (see Blazewicz et al., 2004, 2007; Eiselt and Sandblom, 2004):

First fit (FF). Each task is assigned to the earliest time slot in such a way that no resource or machine limits are violated.

First fit decreasing (FFD). A variant of the *FF* algorithm is applied to a list ordered in non-increasing order of $R_{\max}(i) = \max\{R_v(i)/b_v: 1 \leqslant v \leqslant u\}$.

Iterated lowest fit decreasing (ILFD) for only $u = 1$, *and* $p_i = 1$. Order the tasks as in the FFD algorithm. Assign some C as a lower bound to the optimal makespan. Place job i to the first time slot and process other tasks through the list by placing each job to a time slot for which the total resource requirement of tasks already assigned is minimum. In case that job i cannot be assigned to any of the C slots, terminate the iteration, increase C by 1, and start over.

Garey and Graham (1975) show that the performance bound of *FF* heuristic for the $P|res\cdots|C_{\max}$ problem is:

$$R_{FF}^{\infty} = \min\left\{\frac{m+1}{2}, u+2-\frac{2u+1}{m}\right\}.$$

Krause et al. (1975) establish performance ratios of the above three heuristic methods, based on the $P|res1\cdot\cdot, p_i = 1|C_{\max}$ problem as follows:

$$\frac{27}{10} - \left\lceil\frac{37}{10m}\right\rceil < R_{FF}^{\infty} < \frac{27}{10} - \frac{24}{10m}, \quad R_{FFD}^{\infty} = 2 - \frac{2}{m}, \quad \text{and} \quad R_{ILFD} \leqslant 2.$$

Note that, implementing *FFD* and *ILFD* algorithms improves the performance by about 30%. The following subsections give details of other solution approaches.

3.4.4.1. Problem-based heuristic algorithms. Blazewicz et al. (1993) propose a greedy approach for the $P2|res\cdots, p_i = 1|L_{\max}$ problem by specifying all job pairs that can be processed simultaneously subject to resource constraints. Ordering jobs by the EDD rule and breaking ties by $\max_{1\leqslant v\leqslant u}\{R_v(i)/b_v\}$, they try to assign jobs in pairs to two machines consecutively. They also add two improving heuristic algorithms to this greedy approach by introducing online and offline interchanges between jobs, respectively. We already know from Table 1 that Daniels et al. (1996) develop a polynomial-time algorithm for the static PMFRS problem. Daniels et al. (1996) also develop a heuristic algorithm, named static-based heuristic (SBH), for the dynamic PMFRS problem, i.e., $PD|res1\cdot\cdot, Int|C_{\max}$, that repeatedly solves a series of static PMFRS sub-problems. The results show that the SBH provides an average optimality gap of 1.1% and a maximum gap of 16.9% for the dynamic PMFRS problem over all 540 test problems with 10 and 15 jobs and two to four machines in less than one second of average computation time.

Daniels et al. (1999) propose a decomposition heuristic for a static version of the identical machine UPMFRS problem, i.e., $P|res1\cdot\cdot, Stc|C_{\max}$. In the decomposition algorithm, first, the LPT rule is used to assign n jobs to m machines; and given this initial assignment, the polynomial-time static PMFRS algorithm of Daniels et al. (1996) is applied to allocate the available resources to machines. Then, the new processing times of jobs are calculated with respect to the current resource allocation and again the LPT rule is applied to find the new job-machine assignments. The static PMFRS algorithm is then re-applied to generate the next resource allocation policy. This procedure continues until a job-to-machine assignment, which has been previously evaluated, is encountered. The decomposition heuristic provides an average optimality gap of 8.24% and a maximum gap of 40.59% over all 300 test problems with 10 and 15 jobs and three and four machines in less than one second of average computation time. For the same static UPMFRS problem, Ruiz-Torres and Centeno (2007) develop a lower bound by full enumeration of resource assignment alternatives and propose new heuristic algorithms that combine list scheduling and bin packing approaches with rules to iteratively modify the flexible resource allocation. They report that their heuristic outperforms those given in Daniels et al. (1999) for most cases, and provides results that are close to the lower bounds. Ruiz-Torres et al. (2007) also propose five heuristic algorithms for the same problem with the objective of minimizing the number of tardy jobs. They develop "component heuristics" to make the job-machine assignments which is a part of "complete heuristics" that include also assignment of the flexible resources to the machines. They state that loading one machine at a time outperforms the approach of simultaneously loading all machines. Finally, in a recent study, Sue and Lien (2009) develop two heuristic algorithms for the same static UPMFRS problem. The former assigns jobs to the machines first and then deploys the resources to jobs while the latter proceeds in the reverse manner. The computational results show that the latter heuristic dominates the former one.

Lagrangian relaxation is another viable technique for the PMS problems with additional resources. Typically, a problem-based heuristic is applied to convert the infeasible schedules of Lagrangian relaxation problem (LRP) to the feasible ones. Ventura and Kim (2003) propose a Lagrangian-based heuristic algorithm for the $P|res\cdots, r_i, p_i = 1, d_i|TADD$ problem and obtain efficient results with an average optimality gap of 0.63% in less than 25 seconds of computation time for problems with up to 300 jobs, five machines and three additional resource types. Edis et al. (2008) propose a Lagrangian-based and a problem-based heuristic algorithm for a RCPMSP with machine eligibility restrictions, i.e., $P|res1\cdot 2, M_i, p_i = 1|\sum C_i$. They obtain efficient results with 0.77% and 1.08% average optimality gaps for the Lagrangian heuristic and the problem-based heuristic, respectively, for the test problems with up to 100 jobs and eight machines in less than 1 minute of computation time.

3.4.4.2. Approximation algorithms with worst-case bounds. Since most of the PMS problems with additional resources are NP-hard, a great deal of the literature is dedicated to the development of approximation algorithms with increasingly better worst-case ratios. A polynomial-time algorithm that creates a solution with the objective function value that is at most $\rho \geqslant 1$ times the optimal value is called a ρ-approximation algorithm; the value of ρ is called a *worst-case ratio bound*. On the other hand, a family of algorithms is called a *polynomial-time approximation scheme* (*PTAS*) if, for a given $\varepsilon > 0$, it contains an algorithm that runs in polynomial time in the length of the program input and delivers a worst-case ratio bound of $1 + \varepsilon$. While the family of algorithms in a *PTAS* is desirable since it can approximate arbitrarily close to the optimal solution, it may be undesirable due to its running time which is not polynomial in $1/\varepsilon$. A *PTAS* is called a fully polynomial-time approximation scheme (*FPTAS*) if its running time is polynomial in $1/\varepsilon$.

Krause et al. (1973) propose a 4/3-approximation algorithm for a multi-programmed computer system with identical task processors and restricted amount of memory which can be denoted as $P|res1\cdot\cdot, p_i = 1|C_{\max}$. The algorithm assigns tasks to the successive

unit-time periods with the non-increasing order of memory requirements. Srivastav and Stangier (1997) give a polynomial-time approximation algorithm for the $P|res\cdot\cdot 1, r_i, p_i = 1|C_{\max}$ problem by utilizing the results on generalized version of the multidimensional bin packing problem. Kellerer and Strusevisch (2003) develop a polynomial-time so-called "group technology approximation algorithm" for the $PDm|res111|C_{\max}$ problem with a worst-case bound of 3/2 for larger values of m. The algorithm schedules batches belonging to the resource jobs in the order of machine index so that each batch starts exactly when the previous one is completed. For each machine, the non-resource jobs are scheduled as early as possible, and if required, moved to the time periods after the resource jobs are processed. Moreover, for three and four machines, they propose polynomial-time approximation algorithms with worst case bounds of 5/4. They further develop a *PTAS* for the $PDm|res111|C_{\max}$ problem. Kellerer and Strusevisch (2004) also present a polynomial-time greedy approximation algorithm for a problem with more than one additional resource type and arbitrary number of dedicated machines, i.e.; $PD|res\lambda 11|C_{\max}$. In the algorithm, whenever a machine becomes available, a job that can be processed on that machine at the earliest possible time is chosen and scheduled. This procedure is repeated until all jobs are scheduled. The worst-case performance ratio of this algorithm is 2. They extend this study by also presenting a *PTAS* for the $PDm|res\lambda 11|C_{\max}$ problem, in which the number of machines, m, is fixed. Kellerer and Strusevisch (2008) develop a *PTAS* for a simple dynamic PMFRS problem with binary resource allocation; i.e., $PDm|res111$, $Bi|C_{\max}$. They first try to guess a resource allocation that is very close to that in an optimal schedule, and then follow the lines of the *PTAS* obtained for the $PDm|res111|C_{\max}$ problem given in Kellerer and Strusevisch (2003). Kellerer and Strusevisch (2008) further present an *FPTAS* for the two-machine case of this problem, i.e., $PD2|res111$, $Bi|C_{\max}$. They formulate knapsack subproblems for each of two machines and use a rounding technique to solve these problems in polynomial time.

Recent studies in the literature show that a relaxed mathematical formulation of the original problem may be helpful in designing approximation algorithms. Grigoriev et al. (2005) propose an approximation algorithm that gives a worst case bound of $(4+2\sqrt{2})$ for the dynamic UPMFRS problem with unrelated machines. The approximation method is based on an IP model that defines a relaxation of the original problem. The main idea is the utilization of an aggregate version of the resource constraints, yielding a formulation that does not require time-indexed variables. They then consider the linear programming (LP) relaxation of this relaxed formulation and apply a two-phase rounding procedure to assign resources to jobs and jobs to machines. They finally apply a greedy list scheduling algorithm to generate a feasible schedule. They adapt this method to the dynamic PMFRS problem to obtain a $(3+2\sqrt{2})$-approximation. They also show that the LP-based analysis cannot yield better than a 2-approximation and the problem cannot be approximated within a factor smaller than 3/2 unless P = NP. Subsequently, Kumar et al. (2005) propose a new randomized rounding algorithm for the dynamic UPMFRS problem with unrelated machines. This rounding algorithm together with the greedy scheduling algorithm given in Grigoriev et al. (2005) yields a deterministic 4-approximation algorithm. Inspired by this paper, Grigoriev et al. (2006) develop a 3.75-approximation algorithm for the same problem by using a derandomized version of the rounding procedure of Kumar et al. (2005). They apply a more efficient LP relaxation combined with a new scheduling algorithm, which was inspired by the harmonic algorithm for the bin packing problem, to generate the final schedule. The new scheduling algorithm first partitions the set of jobs into three groups according to the amount of resource assigned and then schedules the jobs group by group. Grigoriev et al. (2007) review the proposed solution approaches given in their previous two studies described above. As already stated, the actual processing times may linearly depend on the number of units of the speeding-up resource. Grigoriev and Uetz (2006) consider such a dynamic PMFRS problem, i.e., $PD|res1\cdot\cdot, Lin|C_{\max}$, and develop a $(3+\varepsilon)$-approximation algorithm. They first determine the resource allocation of the jobs by utilizing a quadratic programming relaxation of the original problem and then obtain the final schedules by applying a greedy list scheduling algorithm.

For a more restricted problem, i.e., the dynamic UPMFRS problem with identical parallel machines, Kellerer (2008) develops a $(3.5+\varepsilon)$-approximation algorithm. The solution procedure follows partially the approach of Grigoriev et al. (2005) by transforming their IP formulation to a multiple choice knapsack problem. The resource allocation is done by solving this problem via an *FPTAS*. The jobs are then allocated to the machines using a variation of greedy list scheduling algorithm. In a later study, Grigoriev and Uetz (2009) propose a nonlinear-programming based $(3+\varepsilon)$-approximation algorithm for the dynamic PMFRS problem. The relaxed nonlinear programming formulation is solved approximately with an *FPTAS* and results in the amount of resources allocated to every job. Then the jobs are scheduled according to an adaptation of the greedy list scheduling algorithm. The algorithm improves upon 3.75-approximation of Grigoriev et al. (2006) and $(3.5+\varepsilon)$-approximation of Kellerer (2008) for the case of the dynamic PMFRS problem. Moreover, the computation time of the new algorithm is polynomial in the input size and the precision $1/\varepsilon$.

Kellerer and Strusevisch (2008) develop a polynomial-time 3/2-approximation algorithm for a PMFRS problem with binary resource requirements and arbitrary number of machines, i.e., $PD|res111$, $Bi|C_{\max}$. They first apply a relaxed IP formulation by an *FPTAS*, which solves the resource allocation sub-problem and generates an instance of the $PD|res111|C_{\max}$ problem. They also find a lower bound on the makespan for the original problem from the solution of this sub-problem. In the second phase, they apply a 3/2-approximation algorithm given in Kellerer and Strusevisch (2003). In a similar manner, Kellerer and Strusevisch (2008) develop a polynomial-time $(3+\varepsilon)$-approximation algorithm for a more general version of the PMFRS problem, which they denote by $PD|res1\sigma\sigma, Int|C_{\max}$. The algorithm proposed for this problem is also a two-phase procedure. In the first phase, the resource allocations are found by solving a relaxed quadratic IP problem by an *FPTAS*. Accordingly, in the second phase, the jobs are allocated to the machines using a simple greedy algorithm. In a recent study, Xu et al. (2011) develop a FPTAS for a dynamic UPMFRS problem with two identical machines and binary resource allocation, i.e., $P2|res111, Bi|C_{\max}$. The authors propose a dynamic programming algorithm which tries to find a well-structured schedule by determining the batches of resource and non-resource jobs for each machine and sequencing them so that the maximum completion time is minimized.

3.4.4.3. Metaheuristics. Even though metaheuristics are widely applied to NP-hard scheduling problems, there are only a few studies using metaheuristics for solving the PMS problems with additional resources. For the dynamic PMFRS problem, Daniels et al. (1997) propose two tabu search heuristics, TSH and TSH–SBH. Both heuristics employ a hierarchical search strategy based on the decomposition of three sub-problems of the dynamic PMFRS problem. Within this hierarchy, tabu search is applied to evaluate the alternative resource allocation policies and job sequences. The first heuristic, TSH, is applied on three initial solutions, while the second heuristic, TSH–SBH, utilizes the final solution from the SBH as its only initial solution. Computational experiments state that TSH–SBH produces more efficient solutions than individual TSH and SBH procedures. Daniels et al. (1999) also propose a tabu search procedure for the static UPMFRS problem with identical machines. The procedure first determines the initial assignment of jobs to the

machines by using the LPT rule and applies SBH to determine the allocation of resources to the machines that minimizes the makespan. Alternative assignments are then generated by considering all pairwise exchange of jobs on different machines and evaluated by the static PMFRS algorithm. If an improvement in the makespan value is observed, then the heuristic solution is updated and the associated pairwise exchange is defined as a tabu move. The computational results show that the tabu search procedure outperforms the decomposition heuristic developed by the same authors.

Olafsson and Shi (2000) propose a solution methodology, named Nested Partitions, which combines global sampling of the feasible region and local search heuristics. They first give an alternative formulation of the dynamic PMFRS problem with the property of active schedules and prove that the optimal solution to the dynamic PMFRS problem is an active schedule. This property reduces the size of the feasible region. Then, they apply the Nested Partitions method to the reformulated PMFRS problem. The method partitions the feasible region in a similar way as the B&B method; however, it only needs to keep a limited number of branches at each iteration. The method also incorporates efficient heuristics to converge faster and to rapidly reach efficient solutions. They compare their method with the static PMFRS algorithm and state that considerable performance benefits may be obtained by using flexible resources.

Li et al. (2003) propose a genetic algorithm-based approach for the $PD|res1\cdot\cdot|C_{\max}$ problem. The proposed approach incorporates an encoding scheme, which gives priorities to jobs, and two decoding schemes. The first decoding scheme provides a general performance with lower time complexity while the second one provides a better performance with higher complexity. They use the first decoding scheme on all chromosomes but use the second scheme on those chromosomes with better fitness values to improve their quality further. They also propose three lower bounds to evaluate the performance of the proposed genetic algorithm approach. Their computational results show that the average percentage gap of the proposed algorithm is no more than 5%.

Constraint Programming (CP), from the field of artificial intelligence, is an alternative solution technique to classical IP methods particularly in scheduling, sequencing and strict feasibility problems. Edis and Ozkarahan (2011) develop CP and IP/CP combined models for the $P|res1\cdot 2, M_i|C_{\max}$ problem and achieve improvements in the computational performance particularly with IP/CP combined model. They also analyze the effect of machine flexibility on the proposed models and find out that the combined IP/CP model gives its best performance for the problems with high processing flexibility. With the combined IP/CP model, they reach optimal results for instances with up to 50 jobs and five machines in less than 1 hour of computation time. Later, for a similar problem, Edis and Ozkarahan (2012) develop IP/IP and IP/CP sequential heuristic approaches where the job-machine assignment is made with an IP loading model and scheduling of the jobs on dedicated machines is obtained by an IP (and alternatively CP) scheduling model. They show that the IP scheduling model performs better when the additional resource constraints are loose, while the CP scheduling model gives better results when the additional resource constraints are tight. For an RCPMSP with unrelated machines and one additional resource type, i.e., $R|res1\cdot\cdot|\sum C_i$, Edis and Oguz (2011) propose a Lagrangian-based CP approach. By relaxing the additional resource constraints, they first obtain an LRP. Given the job-machine assignment from LRP solutions, they then repeatedly find feasible (and efficient) schedules by the use of a proposed CP scheduling model. They finally compare the results with those of pure IP and pure CP model and find out that the proposed Lagrangian-based CP approach produces very efficient results when the resource constraints are tight. Recently, Edis and Oguz (2012) propose IP/CP sequential heuristic approaches for both the dynamic PMFRS and the dynamic UPMFRS problems. For the dynamic PMFRS problem, the authors handle the resource allocation problem with the relaxed IP model of Grigoriev et al. (2005), while the sequencing of jobs is tackled by the proposed CP model. Similarly, for the dynamic UPMFRS problem, they handle the job–machine assignments and resource allocations by the relaxed IP model of Grigoriev et al. (2005) while sequence the jobs by the same CP model. Through the test problems with 50 and 100 jobs, they obtain less than 0.67% and less than 3.31% average optimality gaps, for the PMFRS and UPMFRS problems, respectively, in less than a couple of minutes.

3.5. Other important issues

Other than the significant characteristics of the surveyed papers discussed above, one important point is that almost none of the studies deal with real life problems, although most of the related problems are encountered in real life environments and necessitate industrial-size data to be solved. Moreover, among the studies with computational experiments, the majority of papers, excluding a few ones (Blazewicz et al., 1993; Li et al., 2003; Ventura and Kim, 2003; Edis and Ozkarahan, 2012), deal with small (e.g., 10–15 jobs, 3–5 machines) or medium (e.g., 30–50 jobs, 5–10 machines) size problems.

Since most of the PMS problems with additional resources studied are NP-hard (Blazewicz et al., 1983), it is also not possible to compare the results of the proposed algorithms with the optimal ones. Therefore, a number of studies (e.g., Ventura and Kim, 2003; Li et al., 2003; Ruiz-Torres and Centeno, 2007; Edis et al., 2008; Edis and Oguz, 2011) provide methods to obtain lower bounds. Lagrangian relaxation with a subgradient optimization procedure is an efficient technique to get tight lower bounds (Ventura and Kim, 2003).

Another significant point is that one should choose either the continuous time or the discrete time IP formulations of the corresponding problems. In fact, both modeling approaches have their own strengths. Almost all continuous time formulations require the use of big-M constraints which results in weak LP-relaxation gap (see e.g., Pinto and Grossman, 1997). On the other hand, the discrete time formulations contain enormous number of variables (and also constraints). However, the discrete time formulations have two main advantages. Firstly, this type of formulations is easy to form in comparison to the continuous time ones. Secondly, the discrete time formulations generally perform better than the continuous ones due to its efficient LP relaxation (Van den Akker et al., 2000). All the mathematical programming models for the related problems given in the literature are based on the discrete-time formulations (see Daniels et al., 1996; Ventura and Kim, 2003; Edis et al., 2008; Edis and Oguz, 2011,2012).

There exist some other problem classes related to the RCPMSPs in the literature. As Blazewicz et al. (1983) state, the RCPMSP with identical machines and unit processing times is equivalent to a variant of the bin packing problem in which the number of items to be packed for each bin is restricted to m (number of machines). Therefore, the solution approaches for bin packing problems can be applied to this class of RCPMSPs. More information on the relationship between the bin packing and the resource constrained problems can be found in Garey et al. (1976), Srivastav and Stangier (1997) and Blazewicz and Ecker (1983). In addition, one may wonder if there is a relation between the RCPMSPs and the resource constrained project scheduling problem (RCPSP). In both problems, schedules are constructed subject to resource constraints. However, the RCPMSP differs from the RCPSP in several aspects. The RCPMSP requires an additional job-machine assignment sub-problem whereas the RCPSP does not incorporate machines. Furthermore, the RCPSPs include precedence constraints in nature, while most of the RCPMSPs do not contain precedence relations. A comprehensive review of RCPSPs can be found in Brucker et al. (1999).

4. Strengths and limitations of the existing literature and discussion

Strengths and weaknesses of the studies considered in this review together with the open areas related to PMS problems with additional resources can be summarized as follows:

- In terms of the machine environment, dedicated or identical machines, which are quite easy to handle and solve, are the main focus in most of the studies. On the other hand, uniform and unrelated machine environments are rarely studied.
- A significant number of papers consider the resource-dependent processing times (i.e., PMFRS and UPMFRS problems), which adds a resource allocation sub-problem into the scheme. We observe that a strong theoretical framework has been constructed for this class of problems. Some of the studies however, introduce a number of assumptions (e.g., machines are dedicated, allocation of resources to machines are static) to simplify the problem. More practical cases, the study of dynamic case, where the flexible resource can switch between machines during the schedule, as well as the problems with unspecified job-machine assignment are rarely studied.
- A single additional resource type is considered in most of the studies since it leads to more tractable cases. Moreover, the studies those take more than one additional resource type into consideration focus on the cases with unit processing times and/or two–three machines and/or unit size of additional resources. Thus, other versions with more than one additional resource type still remain as potential research areas.
- Excluding a few papers, all the studies aim to minimize makespan. Other performance criteria therefore remain as open areas for the future studies. Moreover, all studies deal with a single objective function due to the complex nature of this class of problems. The problems incorporating more than one criterion may further be considered in this area.

In the light of the above discussion, we may list the most exciting open areas as follows:

- RCPMSPs with one additional resource type, uniform or unrelated machines, arbitrary processing times and due-date based objective functions, e.g., $Q|res1\cdot\cdot|\sum T_i$, $R|res1\cdot\cdot|\sum T_i$, $Q|res1\cdot\cdot|\sum U_i$, $R|res1\cdot\cdot|\sum U_i$, $Q|res1\cdot\cdot|\sum w_iT_i$, $R|res1\cdot\cdot|\sum w_iT_i$.
- RCPMSPs with more than one additional resource type and arbitrary processing times, e.g., $P|res\cdots|C_{\max}$, $P|res\cdot\cdot1|C_{\max}$, $P|res\cdot11|C_{\max}$.
- Dynamic PMFRS and UPMFRS problems with due date based objective functions, e.g., $PD|res1\cdot\cdot$, $Int|\sum T_i$, $P|res1\cdot\cdot$, $Int|\sum T_i$, $R|res1\cdot\cdot$, $Int|\sum U_i$, $Q|res1\cdot\cdot$, $Int|\sum w_iU_i$.

Regarding the solution approaches, a number of studies give polynomial-time algorithms for some special cases with two or three machines, 0/1 resource requirements, or dedicated machines, mostly by adapting the existing algorithms of some well-known problems, e.g., maximum matching, transportation, bin packing, and shortest path, for the special problem structures. Group technology algorithms are also used to construct polynomial-time algorithms. Our observations on this dimension of the research can be summarized as follows:

- Many problems are computationally difficult; therefore, a significant number of studies are devoted to prove the NP-hardness of the related problems.
- As a result of the above fact, researchers focus on exact and approximation/heuristic algorithms to solve these problems.
- However, exact algorithms are relatively few due to the combinatorial nature of the problem. There exist two B&B and a dynamic programming algorithm proposed for the related problems. In exact algorithms, lower bounds are of significant importance and should be carefully embedded into the proposed algorithms to obtain efficient results.
- Approximation/heuristic algorithms, on the other hand, can be classified into three groups: problem-based heuristic algorithms, ρ-approximation algorithms, and metaheuristics.
- Problem-based heuristic algorithms utilize the problem characteristics to construct the final schedules. A common solution approach in this field is hierarchically solving the sub-problems (e.g., machine assignment, resource allocation) in different orders. A number of problem-based heuristics, on the other hand, are constructed based on the solutions of LRPs. These Lagrangian-based solution approaches have two significant advantages. First, by repeatedly solving LRPs in a subgradient optimization framework, an efficient lower bound can be obtained. Second, by taking different (infeasible) solutions of LRPs at each subgradient step, new and probably better feasible solutions may be obtained.
- We observe that the literature is rich in developing approximation algorithms. A significant number of studies focus on ρ-approximation algorithms providing the worst-case performance guarantees. Some of these algorithms are designed for simple problems with 0/1 resource requirements and utilize the group technology by collecting jobs in resource and non-resource batches. For a group of problems with parallel dedicated machines, first, a near-optimal resource allocation is guessed and then polynomial-time exact algorithms are applied to construct the final schedules. For more complicated problems, on the other hand, researchers utilize relaxed mathematical programming formulations using the aggregated version of additional resources. By applying rounding techniques to these formulations, both the resource allocations and the job-machine assignments are obtained. Finally, the jobs are sequenced by employing simple greedy heuristics to obtain complete schedules.
- In terms of metaheuristics; tabu search and genetic algorithms are applied to the problems. Tabu search is generally used to develop the job-machine assignments, the resource allocations and the job sequences, while genetic algorithm is applied to obtain only the job sequence. An interesting method applied to the related problems is CP which is very powerful in sequencing/scheduling sub-problems where the resource constraints are tight. Hence, handling the job-machine assignment and the resource allocations by either IP techniques or metaheuristics, and tackling sequencing/scheduling decisions with CP may provide more efficient results for the real-life problems.

5. Model extensions and discussion

IP formulations for different problem types have been given in Section 3.2. Analyses of these IP models demonstrate that reformulation and/or revision of these models may result in some performance improvements. In Section 5.1, we present a revised IP model for the dynamic PMFRS problem. In Section 5.2, we extend this formulation to the dynamic UPMFRS problem whose IP model has not been given in the literature yet, and also convert the static UPMFRS model of Daniels et al. (1999) into a linear IP model. The performances of all IP models are briefly discussed through the computational studies.

5.1. Extensions for the dynamic PMFRS problem

For the dynamic PMFRS problem, Daniels et al. (1996) use two sets of binary decision variables in the related IP model ((2)–(11)

(we refer to it as PMFRS-I model). The decision variables x_{ikt} are used for determining the resource allocation and the completion time of jobs, while y_{ih} are used for sequencing the jobs. Our analysis showed that the use of y_{ih} may be redundant and big-M constraints related to these decision variables may result in loose LP relaxation of the IP model. Therefore, the reformulation of constraint sets (6) and (7) of the PMFRS-I model of Daniels et al. (1996) can be written in terms of x_{ikt} variables as follows:

$$\sum_{i\in N_j}\sum_{k\in K_i}\sum_{l=t}^{t+\hat{p}_{ik}-1} x_{ikl} \leqslant 1 \quad t=1,\ldots,T,\ j=1,\ldots,m. \tag{17}$$

Constraint set (17) states that, at most one job can be processed on each machine at any time period, in other words overlapping of jobs on each machine is not allowed.

We also add constraint set (18) considering that one of the machines will be the bottleneck and its total processing time will be a tight lower bound on the makespan.

$$\sum_{i\in N_j} p_i \leqslant C_{\max} \quad i=1,\ldots,n. \tag{18}$$

The reformulation of the dynamic PMFRS model with these two constraint sets is referred to as PMFRS–II model. Preliminary computational analyses show that adding constraint set (18) to the PMFRS-I model of Daniels et al. (1996) provides significant improvements on the solution performance by producing very tight lower bounds. Therefore, this constraint set is also added to PMFRS-I model in the computational study. For both IP models, the initial number of periods T is set as $T=\max_{j\in M}\sum_{i\in N_j}\hat{p}_{i1}$ as stated by Olafsson and Shi (2000).

While generating the data for the test instances, the method of Daniels et al. (1996) is followed. The effect of assigning the additional resource on the processing times is controlled by parameter α, where $0\leqslant\alpha\leqslant 1$. The processing time $\hat{p}_{ik}$ is given by (Daniels et al., 1996):

$$\hat{p}_{ik}=\left[1-\alpha\left(1-\frac{1}{k}\right)\right]\hat{p}_{i1}. \tag{19}$$

For larger values of α, the assignment of the additional resource has a greater impact on the reduction of the processing time. In (19), $\hat{p}_{i1}$ is the duration of job i when processed in its slowest mode and referred to as the normal processing time of job i. We generate $\hat{p}_{i1}$ from a uniform distribution of integers between [10,20] differing from Daniels et al. (1996) since the problem size grows significantly in terms of the number of decision variables x_{ikt} and the related number of constraints with their original data.

Our computational experiments show that the PMFRS-II model, compared to the PMFRS-I model, is more successful in solving instances up to 20 jobs and four, five and six machines to the optimality. We observe that the main disadvantage of the PMFRS-I model is the exponential increase in computation times due to the number of sequencing variables, y_{ih}. The results further indicate that the problem instances become more difficult with the increasing values of α and the decreasing values of R. We also note that since increasing the number of machines reduces the number of jobs per machine, the number of variables and constraints decreases in both models, which in turn reduces their computational difficulty. Consequently, the PMFRS-II model can be used as a benchmark for future studies. For further details on the extensions and the comprehensive computational results of the dynamic PMFRS problem, the interested readers are referred to Edis and Oguz (2012).

5.2. Integer programming models for the static and the dynamic UPMFRS problems

As given in Section 3.2, for the static UPMFRS problem, Daniels et al. (1999) present a nonlinear mathematical model with two sets of decision variables, x_{ik} and y_{ij}. This model can be converted to a linear IP model (for a more general case of unrelated parallel machines) by introducing a single set of decision variables x_{ijk}, where $x_{ijk}=1$ if job i is assigned to machine j with $k\in K_i$ number of additional resource, $x_{ijk}=0$, otherwise. Since the machines are unrelated, the processing time of job i on machine j with $k\in K_i$ number of additional resource is denoted as $\hat{p}_{ijk}$. Since these are trivial extensions, we omit the full formulation here.

On the other hand, although a number of studies deal with different versions of the dynamic UPMFRS problem and propose heuristic and approximation approaches (Grigoriev et al., 2005, 2006, 2007; Kellerer, 2008), a mathematical model of the dynamic UPMFRS problem is not given in the literature yet. We develop the following IP model for the dynamic UPMFRS problem with unrelated machines considering the below decision variables:

$$x_{ijkt}=\begin{cases}1, & \text{if job } i \text{ completes its processing on machine } j\\ & \quad \text{with } k\in K_i \text{ resource at time } t,\\ 0, & \text{otherwise,}\end{cases}$$

$$\text{Minimize} \quad C_{\max}=\max_{i=1,\ldots,n}\sum_{t=1}^{T} t\sum_{j=1}^{m}\sum_{k\in K_i} x_{ijkt}, \tag{20}$$

$$\text{subject to :} \quad \sum_{j=1}^{n}\sum_{k\in K_i}\sum_{t=\hat{p}_{ijk}}^{T} x_{ijkt}=1 \quad i=1,\ldots,n, \tag{21}$$

$$\sum_{i=1}^{n}\sum_{k\in K_i}\sum_{s=t}^{t+\hat{p}_{ijk}-1} x_{ijks}\leqslant 1 \quad j=1,\ldots,m,\ t=1,\ldots,T, \tag{22}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k\in K_i}\sum_{s=t}^{t+\hat{p}_{ijk}-1} x_{ijks}k\leqslant b \quad t=1,\ldots,T, \tag{23}$$

$$\sum_{i=1}^{n}\sum_{k\in K_i}\sum_{t=1}^{T}\hat{p}_{ijk}x_{ijkt}\leqslant C_{\max} \quad j=1,\ldots,m, \tag{24}$$

$$x_{ijkt}\in\{0,1\} \quad i=1,\ldots,n,\ j=1,\ldots,m,\ k\in K_i,\ t=1,\ldots,T. \tag{25}$$

The objective function (20) is to minimize the makespan over the jobs. Constraint set (21) ensures that each job is assigned to a single machine with a fixed amount of additional resource and completed at a unique time. Constraint set (22) states that, on each machine j, at most one job can be processed at any time. Constraint set (23) indicates that, at any time period, the total amount of additional resource assigned to jobs should be less than or equal to the available amount, b. Constraint set (24) provides a dynamic and efficient lower bound to the makespan. Finally, constraint set (25) states that all x_{ijkt} are binary decision variables.

The preliminary computational results show that the static UPMFRS model finds the optimal solutions for the instances with up to 50 jobs and five machines. Thus, for the above IP model of the dynamic UPMFRS problem, the initial number of periods, T, is set as the optimal makespan value of the static UPMFRS model.

Generation of the data for the test instances of the UPMFRS problem follows the procedure given in Section 5.1. The processing time $\hat{p}_{ijk}$ is calculated by:

$$\hat{p}_{ijk}=\left[1-\alpha\left(1-\frac{1}{k}\right)\right]\hat{p}_{ij1}, \tag{26}$$

where $\hat{p}_{ij1}$ is the duration of job i on machine j when processed in its slowest mode.

Our computational experiments indicate that the static UPMFRS model is more effective in obtaining the optimal solutions in comparison to the dynamic UPMFRS model, as expected. The static UPMFRS model can reach optimal solutions for the test instances up to 50 jobs and five machines, while the dynamic UPMFRS model is able to reach optimal solutions for the test instances with up to 15 jobs and three and five machines in 1 hour of computation time. We remark that the difficulty of the problem instances increases for the static UPMFRS model as the number of machines increases as well as the value of α increases. We further observe that the value of R affects both the static and the dynamic UPMFRS models similarly: As the value of R increases, the efficiency of the models decreases up to a point. Thereafter, the efficiency of the model increases. This middle point indicates the additional resource size corresponding to the maximum 'makespan-additional resource trade-off'. Consequently, researchers may use these two IP models to obtain benchmark solutions for their future studies. For further details on the extensions and the extensive computational study of the static and the dynamic UPMFRS problems, the interested readers are referred to Edis and Oguz (2012).

6. Conclusion

The study of PMS problems with additional resources is a significant area of research. In this paper, after giving main definitions, notation, classifications and assumptions, we surveyed the related studies and classified them in an efficient framework in terms of machine environments, objective functions, additional resource characteristics, complexity results and solution methods and other important issues. Then, we presented and discussed some model extensions for the dynamic PMFRS problem. Next we gave a linear IP model for the static UPMFRS problem and an IP model for the dynamic UPMFRS problem and discussed their performance based on a computational study.

The contribution of this review paper to the related scheduling literature is threefold. First, a systematic representation of papers in the area of PMS with additional resources, including relationships and connections with each other, is provided so that researchers can identify different problem types and characteristics. Second, the strengths and the weaknesses of the literature together with open areas for future studies are emphasized in terms of five main categories. Finally, a number of extensions to the available IP models are given and their advantages are discussed.

Acknowledgements

Part of this study has been carried out when Emrah B. Edis was with Dokuz Eylül University and during his post-doctoral research at Koç University with the fellowship of TUBİTAK (Scientific and Technological Research Council of Turkey) under Grant No: 2218. We also thank the anonymous referees for their helpful comments and suggestions.

References

Blazewicz, J., 1978. Complexity of computer scheduling algorithms under resource constraints. Proceedings of the First Meeting AFCET-SMF on Applied Mathematics, 169–178.

Blazewicz, J., 1979. Deadline scheduling of tasks with ready times and resource constraints. Information Processing Letters 8 (2), 60–63.

Blazewicz, J., 1981. Solving the resource constrained deadline scheduling problem via reduction to the network flow problem. European Journal of Operational Research 6, 75–79.

Blazewicz, J., Ecker, K., 1983. A linear time algorithm for restricted bin packing and scheduling problems. Operations Research Letters 2 (2), 80–83.

Blazewicz, J., Lenstra, J.K., Rinnooy Kan, A.H.G., 1983. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5, 11–24.

Blazewicz, J., Barcelo, J., Kubiak, W., Röck, H., 1986a. Scheduling tasks on two processors with deadlines and additional resources. European Journal of Operational Research 26, 364–370.

Blazewicz, J., Cellary, W., Slowinski, R., Weglarz, J., 1986b. Scheduling under Resource Constraints – Deterministic Models. J.C. Baltzer AG, Scientific Publishing, Switzerland.

Blazewicz, J., Kubiak, W., Röck, H., Szwarcfiter, J., 1987. Minimizing mean flow time with parallel processors and resource constraints. Acta Informatica 24, 513–524.

Blazewicz, J., Kubiak, W., Szwarcfiter, J., 1989. Scheduling independent fixed-type tasks. In: Slowinski, R., Weglarz, J. (Eds.), Advances in Project Scheduling. Elsevier, Amsterdam, pp. 225–236.

Blazewicz, J., Kubiak, W., Martello, S., 1993. Algorithms for minimizing maximum lateness with unit length tasks and resource constraints. Discrete Applied Mathematics 42, 123–138.

Blazewicz, J., Brauner, N., Finke, G., 2004. Scheduling with discrete resource constraints. In: Leung, J.Y.-T. (Ed.), Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, USA (Chapter 23).

Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J., 2007. Handbook on Scheduling. Springer-Verlag, New York (Chapter 12).

Brucker, P., Kramer, A., 1996. Polynomial algorithms for resource-constrained and multiprocessor task scheduling problems. European Journal of Operational Research 90, 214–226.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E., 1999. Resource-constrained project scheduling: notation, models, and methods. European Journal of Operational Research 112 (1), 262–273.

Chen, J.-F., 2005. Unrelated parallel machine scheduling with secondary resource constraints. International Journal of Advanced Manufacturing Technology 26, 285–292.

Chen, J.-F., Wu, T.-H., 2006. Total tardiness minimization on unrelated parallel machine scheduling with auxiliary equipment constraints. OMEGA 34, 81–89.

Cheng, T.C.E., Sin, C.C.S., 1990. A state-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research 47, 271–292.

Daniels, R.L., Hoopes, B.J., Mazzola, J.B., 1996. Scheduling parallel manufacturing cells with resource flexibility. Management Science 42 (9), 1260–1276.

Daniels, R.L., Hoopes, B.J., Mazzola, J.B., 1997. An analysis of heuristics for the parallel-machine flexible-resource scheduling problem. Annals of Operations Research 70, 439–472.

Daniels, R.L., Hua, S.Y., Webster, S., 1999. Heuristics for parallel-machine flexible-resource scheduling problems with unspecified job assignment. Computers and Operations Research 26, 143–155.

Edis, E.B., Oguz, C., 2011. Parallel machine scheduling with additional resources: a Lagrangian-based constraint programming approach. In: Achterberg, T., Beck, J.C. (Eds.), LNCS, Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, vol. 6697. Springer-Verlag, Berlin, Germany, pp. 92–98.

Edis, E.B., Oguz, C., 2012. Parallel machine scheduling with flexible resources. Computers and Industrial Engineering 63, 433–447.

Edis, E.B., Ozkarahan, I., 2011. A combined integer/constraint-programming approach to a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. Engineering Optimization 43 (2), 135–157.

Edis, E.B., Ozkarahan, I., 2012. Solution approaches for a real-life resource-constrained parallel machine scheduling problem. The International Journal of Advanced Manufacturing Technology 58, 1141–1153.

Edis, E.B., Araz, C., Ozkarahan, I., 2008. Lagrangian-based solution approaches for a resource-constrained parallel machine scheduling problem with machine eligibility restrictions. In: Nguyen, N.T., Borzemski, L., Grzech, A., Ali, M. (Eds.), LNAI, New Frontiers in Applied Artificial Intelligence, vol. 5027. Springer-Verlag, Berlin, Germany, pp. 337–346.

Eiselt, H.A., Sandblom, C-L., 2004. In: Decision Analysis, Location Models, and Scheduling Problems. Springer-Verlag, New York, pp. 399–421.

Garey, M.R., Graham, R.L., 1975. Bounds for multiprocessor scheduling with resource constraints. SIAM Journal on Computing 4 (2), 187–200.

Garey, M.R., Johnson, D.S., 1975. Complexity results for multiprocessor scheduling under resource constraints. SIAM Journal on Computing 4 (4), 397–411.

Garey, R.L., Graham, D.S., Johnson, A., Yao, A.C.-C., 1976. Resource constrained scheduling as generalized bin packing. Journal of Combinatorial Theory, Series A 21 (3), 257–298.

Glass, C.A., Shafransky, Y.M., Strusevich, V.A., 2000. Scheduling for parallel dedicated machines with a single server. Naval Research Logistics 47 (4), 304–328.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5, 287–326.

Grigoriev, A., Uetz, M., 2006. Scheduling parallel jobs with linear speedup. In: Erlebach, T., Parsiano, G. (Eds.), LNCS, Approximation and Online Algorithms, vol. 3879. Springer-Verlag, Berlin, Germany, pp. 203–215.

Grigoriev, A., Uetz, M., 2009. Scheduling jobs with time-resource trade-off via non-linear programming. Discrete Optimization 6, 414–419.

Grigoriev, A., Sviridenko, M., Uetz, M., 2005. Unrelated parallel machine scheduling with resource dependent processing times. In: Jünger, M., Kaibel, V. (Eds.), LNCS, Integer Programming and Combinatorial Optimization, vol. 3509. Springer-Verlag, Berlin, Germany, pp. 182–195.

Grigoriev, A., Sviridenko, M., Uetz, M., 2006. LP rounding and an almost harmonic algorithm for scheduling with resource dependent processing times. In: Diaz, J., Jansen, K., Rolim, J., Zwick, U. (Eds.), LNCS, Approximation, Randomization, and Combinatorial Optimization, vol. 4110. Springer-Verlag, Berlin, Germany, pp. 140–151.

Grigoriev, A., Sviridenko, M., Uetz, M., 2007. Machine scheduling with resource dependent processing times. Mathematical Programming Series B 110, 209–228.

Hall, N.G., Potts, C.N., Sriskandarajah, C., 2000. Parallel machine scheduling with a common server. Discrete Applied Mathematics 102, 223–243.

Jozefowska, J., Weglarz, J., 2004. Scheduling with resource constraints –continuous resources. In: Leung, J.Y-T. (Ed.), Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, USA (Chapter 24).

Kellerer, H., 2008. An approximation algorithm for identical parallel machine scheduling with resource dependent processing times. Operations Research Letters 36, 157–159.

Kellerer, H., Strusevisch, V.A., 2003. Scheduling parallel dedicated machines under a single non-shared resource. European Journal of Operational Research 147, 345–364.

Kellerer, H., Strusevisch, V.A., 2004. Scheduling problems for parallel dedicated machines under multiple resource constraints. Discrete Applied Mathematics 133, 45–68.

Kellerer, H., Strusevisch, V.A., 2008. Scheduling parallel dedicated machines with the speeding-up resource. Naval Research Logistics 55 (5), 377–389.

Kovalyov, M.Y., Shafransky, Y.M., 1998. Uniform machine scheduling of unit-time jobs subject to resource constraints. Discrete Applied Mathematics 84, 253–257.

Krause, K.L., Shen, V.Y., Schwetman, H.D., 1973. A task scheduling algorithm for a multiprogramming computer system. ACM SIGOPS Operating Systems 7 (4), 112–118.

Krause, K.L., Shen, V.Y., Schwetman, H.D., 1975. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. Journal of the Association for Computing Machinery 22 (4), 522–550.

Kumar, V.S.A., Marathe, M.V., Parthasarathy, S., Srinivasan, A., 2005. Approximation algorithms for scheduling on multiple machines. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 254–263.

Li, Y., Wang, F., Lim, A., 2003. Resource constraints machine scheduling: a genetic algorithm approach. Congress on Evolutionary Computation 1–4, 1080–1085.

Mokotoff, E., 2001. Parallel machine scheduling problems: a survey. Asia Pacific Journal of Operational Research 18, 193–242.

Olafsson, S., Shi, L., 2000. A method for scheduling in parallel manufacturing systems with flexible resources. IIE Transactions 32, 135–146.

Ozdamar, L., Ulusoy, G., 1994. A local constraint based analysis approach to project scheduling under general resource constraints. European Journal of Operational Research 79, 287–298.

Pfund, M., Fowler, J.W., Gupta, J.N.D., 2004. Multi-objective unrelated parallel-machine deterministic scheduling problems. Journal of the Chinese Institute of Industrial Engineers 21 (3), 230–241.

Pinedo, M., 2008. Scheduling: Theory, Algorithms and Systems, third ed. Springer, New York.

Pinto, J.M., Grossman, I.E., 1997. A logic-based approach to scheduling problems with resource constraints. Computers and Chemical Engineering 21 (8), 801–818.

Ruiz-Torres, A.J., Centeno, G., 2007. Scheduling with flexible resources in parallel workcenters to minimize maximum completion time. Computers and Operations Research 34, 48–69.

Ruiz-Torres, A.J., Lopez, F.J., Ho, J.C., 2007. Scheduling uniform parallel machines subject to a secondary resource to minimize the number of tardy jobs. European Journal of Operational Research 179 (2), 302–315.

Shabtay, D., Kaspi, M., 2006. Parallel machine scheduling with a convex resource consumption function. European Journal of Operational Research 173, 92–107.

Slowinski, R., 1980. Two approaches to problems of resource allocation among project activities – a comparative study. Journal of the Operational Research Society 31 (8), 711–723.

Srivastav, A., Stangier, P., 1997. Tight approximations for resource constrained scheduling and bin packing. Discrete Applied Mathematics 79, 223–245.

Sue, L.-H., Lien, C.-Y., 2009. Scheduling parallel machines with resource-dependent processing times. International Journal of Production Economics 117, 256–266.

Tamaki, H., Hasegawa, Y., Kozasa, J., Araki, M., 1993. Application of search methods to scheduling problem in plastics forming plant: a binary representation approach. In: Proceedings of the 32nd IEEE Conference on Decision and, Control, 3845–3850.

Van den Akker, J.M., Hurkens, C.A.J., Savelsbergh, M.W.P., 2000. Time-indexed formulations for machine scheduling problems: column generation. Informs Journal on Computing 12 (2), 111–124.

Ventura, J.A., Kim, D., 2000. Parallel machine scheduling about an unrestricted due date and additional resource constraints. IIE Transactions 32, 147–153.

Ventura, J.A., Kim, D., 2003. Parallel machine scheduling with earliness–tardiness penalties and additional resource constraints. Computers and Operations Research 30, 1945–1958.

Xu, H., Chen, L., Ye, D., Zhang, G., 2011. Scheduling on two identical machines with a speed-up resource. Information Processing Letters 111, 831–835.