Journal of the Operational Research Society

Taylor & Francis
Taylor & Francis Group

Approximation algorithms for makespan minimization on identical parallel machines under resource constraints

Journal:	*Journal of the Operational Research Society*
Manuscript ID	TJOR-2019-OP-0366.R2
Manuscript Type:	Original Paper
Date Submitted by the Author:	n/a
Complete List of Authors:	Strusevich, Vitaly; University of Greenwich, School of Computing and Mathematical Sciences
Keywords:	parallel identical machines, resource constraints, group technology, approximation
Abstract:	The problem of minimizing the makespan on parallel identical machines is considered in the presence of additional resources, provided that some jobs at any time of their processing require one unit of a particular resource. We establish a lower bound on the worst-case performance of any group technology algorithm, which schedules the composite jobs formed of the original jobs that require the same resource. A simple group technology algorithm is given such that in the worst case no group technology algorithm performs better. An algorithm for the two-machine case is presented which guarantees a tight worst-case performance ratio of 6/5.

Approximation algorithms for makespan minimization on identical parallel machines under resource constraints

Vitaly A. Strusevich[a]

[a] School of Computing and Mathematical Sciences, University of Greenwich, Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, UK
and Vitaly A. Strusevich
School of Computing and Mathematical Sciences,
University of Greenwich, Old Royal Naval College,
Park Row, Greenwich, London SE10 9LS, UK;
E-mail: V.Strusevich@gre.ac.uk.

ARTICLE HISTORY
Compiled January 30, 2020

ABSTRACT
The problem of minimizing the makespan on parallel identical machines is considered in the presence of additional resources, provided that some jobs at any time of their processing require one unit of a particular resource. We establish a lower bound on the worst-case performance of any group technology algorithm, which schedules the composite jobs formed of the original jobs that require the same resource. A simple group technology algorithm is given such that in the worst case no group technology algorithm performs better. An algorithm for the two-machine case is presented which guarantees a tight worst-case performance ratio of 6/5.

KEYWORDS
parallel identical machines; resource constraints; group technology; approximation

1. Introduction

The problems of the classical Machine Scheduling are formulated in terms of jobs to be processed on machines. However, in practical applications to process a job not only a machine is required but also additional resources, which are different from the processing machines. Thus, within Machine Scheduling there is an established research area known as *Scheduling under Resource Constraints*. There are multiple publications in this area which have been reviewed in a number of surveys, from the earliest by Błażewicz et al. (1983) till the most recent by Błażewicz et al. (2019).

In this paper, we consider one of the problems on parallel machines under resource constraints. The jobs of set $N = \{J_1, J_2, \ldots, J_n\}$ have to be processed non-preemptively on m identical parallel machines $M_1, \ldots, M_m$. The processing time of job $J_j \in N$ is equal to p_j. There are also $q \geq 1$ additional renewable resources, so that at any time moment exactly one unit of each resource is available. Set N is partitioned into $q+1$ disjoint sets N_0, N_1, $\ldots N_q$, where each job of set N_0 does not require any resource, while each job of set N_k at any time of its processing requires one unit of resource k, $1 \leq k \leq q$. A job of set N_0 is called a *non-resource* job; all other jobs are called the *resource* jobs.

For any k, $1 \leq k \leq q$, at most one job of set N_k can be processed at a time. The objective is to minimize the makespan, i.e., the maximum completion time. This problem is clearly no easier than the corresponding problem without resource constraint. The latter problem is well-known to be NP-hard for any fixed number of machines $m \geq 2$. Therefore, a search for an approximation algorithm for the problem under consideration is of interest. In this paper, we focus on a class of so-called group technology algorithms, which treat the jobs that require the same resource as a single composite job.

The remainder of this paper is organized as follows. In Section 2, we discuss the issues of the notation for scheduling problems with resource constraints, briefly review relevant results and address the motivation for studying the model under consideration. In Section 3, we establish a lower bound on the worst-case ratio of any group technology algorithm and present a simple algorithm such that in the worst case no group technology algorithm may perform better. For the two-machine case, an algorithm

CONTACT Vitaly A. Strusevich. Email: V.Strusevich@gre.ac.uk

with an improved performance is given in Section 4. Concluding remarks are contained in Section 5.

2. Notation, Motivation and Review

For a scheduling problem, the completion time of a job J_j in a feasible schedule S is denoted by $C_j(S)$; often, if no confusion arises, the reference to the schedule is omitted and we simply write C_j. The most popular objective functions for the scheduling problems are the makespan $C_{\max}(S) = \max\{C_j | J_j \in N\}$ and the total completion time $\sum_{J_j \in N} C_j(S)$.

In order to denote scheduling problems in a clear and compact way, the three-field classification scheme of the form $\alpha|\beta|\gamma$ is widely accepted, where α describes a machine environment, β is responsible for presenting the processing conditions and γ is the objective function. For example, the classical problem of minimizing the makespan $C_{\max}$ on m parallel machines is denoted either by $P||C_{\max}$ if the number of machines is variable (part of the input) or by $Pm||C_{\max}$ if the number of machines is fixed and equal to m. The middle field is empty, and that means that in a feasible schedule no preemption is allowed. Similarly, the problem of minimizing the total completion time for the same machine environment is denoted either by $P||\sum C_j$ or by $Pm||\sum C_j$ depending on whether the number of machines is variable or fixed, respectively.

For the scheduling problems under the resource constraints, the notation that has become standard since its first appearance in Błażewicz et al. (1983) places into the middle field β a string that specifies the rules of resource usage by addressing three parameters.

Despite its popularity, that notation does not provide enough details in order to distinguish between various versions of the resource-constrained problems. Below, we propose to extend the number of the parameters to four and add to the middle field β a string of the form "res $\rho_1\rho_2\rho_3\rho_4$", where

- ρ_1 is the number of available renewable resources;
- ρ_2 is an upper bound on the number of resources a job may need;
- ρ_3 is an upper bound on the number of units of any resource available at a time;
- ρ_4 is an upper bound on the number of units of any resource that can be consumed by a job at a time.

The value of each of this parameters is either a known constant or the symbol "$\cdot$" if the value of the parameter is variable (part of the input). In accordance with this updated scheme, the problem of our primary concern can be denoted by $P|\text{res }\cdot 111|C_{\max}$. The notation implies that

- $\rho_1 =$ "$\cdot$", i.e., that there several renewable resources (as above, we use the variable q to denote their number);
- $\rho_2 = 1$, i.e., each job needs either none or exactly one resource at any time of its processing;
- $\rho_3 = 1$, i.e., one unit of each resource is available at a time;
- $\rho_4 = 1$, i.e., a resource job consumes one unit of the relevant resource at any time of its processing.

In the traditional scheme the parameter ρ_2 was missing. Therefore, in most of the previously considered models with resource constraints it was assumed that a job might need any number of resources. The four-parameter scheme is free from that drawback.

In fact, there is a considerable interest, both from the theoretical and practical prospective, in scheduling models in which a job may only need a fixed number of resources, in particular in those models that are described by the string "res $\cdot 111$".

A natural meaningful interpretation of problem $P|\text{res }\cdot 111|C_{\max}$ is related to human resource management. The projects (jobs) can be performed by any of the available m teams (machines). However, to be able to perform certain projects a team must additionally include an extra employee with a special skill. For each skill k, $1 \le k \le q$, there exactly one employee who possesses this skill, i.e., such an employee can be seen as a renewable resource k. This, for instance, happens if a project is to be done for a foreign client and a team must additionally include a linguistic expert to deal with issues of communication and documentation. An individual that knows a particular foreign language (e.g., French, Japanese or Norwegian) is understood as an additional resource. Provided that only one person is available to be included into a team that requires an expert in a particular language, in order to complete all projects as early as possible, problem $P|\text{res }\cdot 111|C_{\max}$ has to be solved.

An interesting application is reported by Hebrard et al. (2016), who motivate their study of problem $P|\text{res }\cdot 111|C_{\max}$ by the problem that arise in satellite data download management. The observation satellite makes multi-frequency optical recordings which are kept in q memory banks on board the satellite, each bank containing recordings made at the same frequency k, $1 \le k \le q$. When the satellite

establishes a communication link with a ground station, the observations can be downloaded via m communication channels, at most one file from each of the q banks at a time. The purpose is to perform the download as fast as possible due to a rather short communication window. Thus, here a file in each memory bank can be seen as a job that requires the same resource and the m channels play roles of the machines.

The following problem that is found in the microelectronic industry motivates the study of problem $P\,|\text{res}\ \cdot 111|\sum C_j$ in Janssen et al. (2018). Pieces of photolithography equipment are used to transfer the geometric pattern of the chip design onto a wafer. This is done by putting light onto the wafer through a reticle that contains the required geometric pattern. In scheduling terms, the pieces of photolithography equipment are (parallel) machines, and reticles are additional resources, one per each chip design.

Now we briefly review the issues of the computational complexity and approximability of scheduling problems that are relevant to the main object of our study, the problem $P\,|\text{res}\ \cdot 111|\,C_{\max}$.

We start with the classical problem on identical parallel machines with no resource constraints. It is well-known that Problem $P2\,|\,|\,C_{\max}$ is NP-hard in the ordinary sense, while problem $P\,|\,|\,C_{\max}$ is strongly NP-hard. For many years, the main topic of research on problem $P\,|\,|\,C_{\max}$ has been that of developing approximation algorithms and schemes. It is beyond the scope of this paper to review all results on approximability of problem $P\,|\,|\,C_{\max}$; we refer the reader to the focused survey Chen (2004). Below we only mention the definitions and results which are closely related to the content of this paper.

For an NP-hard scheduling problem of minimizing the makespan, the quality of an approximation algorithm that delivers a feasible schedule S^H is measured by bounding the ratio $C_{\max}\left(S^H\right)/C_{\max}\left(S^*\right)$. A polynomial-time algorithm is called an R-approximation algorithm if the inequality

$$\frac{C_{\max}\left(S^H\right)}{C_{\max}\left(S^*\right)} \leq R \tag{1}$$

holds for all instances of the problem. The value of R is tight if the inequality (1) holds as equality for at least one instance of the problem. A tight value of R is called the worst-case ratio.

Historically, one of the first results on worst-case analysis of scheduling approximation algorithms belongs to R.L. Graham who described and analyzed a so-called List Scheduling algorithm for problem $P\,|\,|\,C_{\max}$; see Graham (1966). The List Scheduling algorithm scans the jobs in accordance with a list and assigns the next job to the first available machine. In the case of an arbitrary list, the algorithm by Graham (1966) is a $(2-1/m)$-approximation algorithm. It is shown in Graham (1969) that if the jobs are sorted in the LPT order, i.e., in non-increasing order of their processing times, then the List Scheduling algorithm behaves as a $(4/3-1/(3m))$-approximation algorithm. A clarification on the performance of the LPT List Scheduling algorithm is reported in Chen (1993).

We now pass to considering problems under resource constraints on identical parallel machines. Most of known results in the area are on the models with unit processing times. In particular, problem $P\,|\text{res}\cdot\cdot 11|\,C_{\max}$ is NP-hard in the strong sense even if all processing times are unit; see Błażewicz et al. (1983). If the processing times are arbitrary, then clearly problem $P2\,|\text{res}\cdot 111|\,C_{\max}$ with at most one resource per job is no simpler than the classical problem $P2\,|\,|\,C_{\max}$ without resource constraints and is therefore NP-hard.

All surveys, including Błażewicz et al. (2004), Edis et al. (2013) and Błażewicz et al. (2019), demonstrate that there is lack of approximation results on parallel machine scheduling with arbitrary processing times under resource constraints. The only result mentioned in these reviews is a R-approximation algorithm designed in for problem $P\,|\text{res}\cdot\cdot\cdot\cdot|\,C_{\max}$ with q resources, where $R=\min\left\{\frac{1}{2}m, q+2-\frac{1}{m}\left(2q+1\right)\right\}$ is not bounded by a constant and is therefore impractical. Besides, the paper Hebrard et al. (2016) delivers a number of approximation algorithms for problem $P\,|\text{res}\cdot 111|\,C_{\max}$, in both on-line and off-line settings. In particular, one of the algorithms described in Hebrard et al. (2016) behaves as an R-approximation algorithm with $R=(2m)/(m+1)$, although this bound is only proved tight for $m=2$. One of the results of our paper is a simpler algorithm with the same worst-case performance that cannot be improved in a certain class of algorithms.

We conclude our brief review with considering the model with parallel dedicated machines under resource constraints. For this machine environment, denoted by "PD", it is known in advance on which machine a particular job must be processed

As shown in Kellerer & Strusevich (2004), problem $PD2\,|\text{res}\cdot 111|\,C_{\max}$ admits a linear time solution algorithm. On the other hand, problem $PD3\,|\text{res}1111|\,C_{\max}$ is NP-hard in the ordinary sense, while problem $PD\,|\text{res}1111|\,C_{\max}$ with a variable number of machines $m\geq 3$ is NP-hard in the strong sense, as proved in Kellerer & Strusevich (2003). In the latter paper it is also shown that problem $PD\,|\text{res}1111|\,C_{\max}$ admits an R-approximation algorithm, where $R=3/2-1/(m+1)$ for an odd m and $R=3/2-1/m$ for an even m. For $m=3$, the worst-case performance ratio can be improved to

5/4.

Kellerer & Strusevich (2004) show that problem $PD2\,|\text{res}11\cdot\cdot|\,C_{\max}$ with a single resource of an arbitrary quantity and problem $PD2\,|\text{res}2211|\,C_{\max}$ with two resources of unit quantity is solvable in $O(n)$ time each. On the other hand, each of the problems $PD2\,|\text{res}2222|\,C_{\max}$ and $PD2\,|\text{res}3311|\,C_{\max}$ is NP-hard. Furthermore, a 2-approximation algorithm is developed for problem $PD\,|\text{res}\cdot 111|\,C_{\max}$ with more than two parallel dedicated machines.

3. Group Technology Approach

In this section, we present and analyze an approximation algorithm for problem $P\,|\text{res}\cdot 111|\,C_{\max}$ that is based on the group technology approach. A group technology approach is a scheduling technique which is based on substitution of a group of jobs sharing a certain property by a single composite job. Such a technique is often used in scheduling with precedence constraints (see, e.g., Lawler (1978)) and with batching (see, e.g., Strusevich (2000)). An approximation algorithm for problem $PD\,|\text{res}1111|\,C_{\max}$ presented in Kellerer & Strusevich (2003) is also based on the group technology approach.

Given an instance of the problem, for a non-empty set $Q \subseteq N$ of jobs denote

$$p(Q) = \sum_{j \in Q} p_j;$$

for completeness, define $p(\varnothing) = 0$.

For a schedule S that is feasible for problem $P\,|\text{res}\cdot 111|\,C_{\max}$, the job that completes last is called the *terminal* job, and the machine to which that job is assigned is called the *critical* machine. The makespan is equal to the sum of the processing times of the jobs assigned to a critical machine.

Similarly to the classical problem $P\,||\,C_{\max}$ with no resource constraints, the makespan of any schedule that is feasible for problem $P\,|\text{res}\cdot 111|\,C_{\max}$ cannot be smaller that the processing time of any job of set N_0 that does not require any resource. Additionally, the makespan cannot be smaller than the average machine load $p(N)/m$. Besides, for any k, $1 \le k \le q$, no two jobs of set N_k cannot be processed in parallel. Thus, the following lower bound

$$C_{\max}(S) \ge \max\left\{\frac{1}{m}p(N), \max\{p_j | J_j \in N_0\}, \max\{p(N_k) \,|\, 1 \le k \le q\}\right\} \tag{2}$$

holds. A schedule that is optimal for problem $P\,|\text{res}\cdot 111|\,C_{\max}$ is denoted by S^*.

Given an instance of problem $P\,|\text{res}\cdot 111|\,C_{\max}$, associate each set N_k of the resource jobs with a composite job V_k, $1 \le k \le q$. The processing times of these composite jobs are defined by

$$p(V_k) = p(N_k), \ 1 \le k \le q.$$

Here V_k is a new composite job, while N_k is the set of initially given jobs that require resource k, $1 \le k \le q$, and the processing time of the composite job V_k is equal to the sum of the processing times of the jobs in the corresponding set N_k.

For consistency, each non-resource job $J_j \in N_0$ is also treated as a composite job, and after an appropriate renumbering, these jobs are denoted by $V_{q+1}, \ldots, V_{q+n_0}$, where $n_0 = |N_0|$. The processing time $p(V_{q+i})$, $1 \le i \le n_0$, of a composite job associated with some non-resource job is equal to the processing time of that non-resource job.

Thus, with introduction of the composite jobs, we may associate the initial instance I of the problem that consists of n jobs, some of which require a resource, with a new instance I_G that consists of $q + n_0$ composite jobs. For illustration, consider the instance presented in Table 1. For such an instance I, the associated instance I_G will contain 10 composite jobs, each of duration 3.

The group technology approach is based on scheduling an instance I_G of composite jobs. Let S_G be a schedule for instance I_G on m parallel machines. In such a schedule, we need not worry about the resource constraints, since all jobs that require the same resource are processed as a block on one of the machines. We will call a schedule S_G of the composite jobs a *group technology* schedule. Clearly, a schedule S_G can be converted to a feasible schedule for the original instance by replacing a composite job by the block of the corresponding original jobs.

Let S_G^* be an optimal group technology schedule, i.e., the inequality

$$C_{\max}(S_G^*) \le C_{\max}(S_G)$$

Resource	Sets	Durations
1	$N_1 = \{J_1, J_4\}$	$p_1 = 1, p_4 = 2$
2	$N_2 = \{J_2, J_5\}$	$p_2 = 2, p_5 = 1$
3	$N_3 = \{J_3\}$	$p_3 = 3$
4	$N_4 = \{J_6, J_7, J_8\}$	$p_6 = p_7 = p_8 = 1$
5	$N_5 = \{J_9\}$	$p_9 = 3$
6	$N_6 = \{J_{10}\}$	$p_{10} = 3$
7	$N_7 = \{J_{11}\}$	$p_{11} = 3$
8	$N_8 = \{J_{12}\}$	$p_{12} = 3$
9	$N_9 = \{J_{13}\}$	$p_{13} = 3$
10	$N_{10} = \{J_{14}\}$	$p_{14} = 3$

Table 1. Illustration: Input with $m = 3$ machines, $q = 10$ resources and 14 original jobs

holds for any group technology schedule S_G. The following lemma shows how bad a group technology schedule could be with respect to the overall optimal schedule. Consider the ratio $C_{\max}(S_G^*)/C_{\max}(S^*)$ of the makespan for an optimal group technology schedule to the makespan of the overall optimal schedule. In order to show that in the worst case the above ratio cannot be smaller than a particular value, we need to exhibit at least one instance of the problem for which the ratio is equal to such a value. In the proof above we construct a series of instances that depend on an integer r. The required instance in built in such a way that in the optimal group technology schedule S_G^* the terminal job is scheduled in the rth position on the critical machine.

Lemma 3.1. *For any integer $r \geq 2$, there exists an instance of problem $P|\text{res} \cdot 111|C_{\max}$ such that*

$$\frac{C_{\max}(S_G^*)}{C_{\max}(S^*)} = \frac{rm}{(r-1)m+1}.$$

Proof. Given an integer $r \geq 2$, consider the following instance of problem $P|\text{res} \cdot 111|C_{\max}$ with $n = (r+1)m - 1$ jobs, $q = (r-1)m + 1$ resources, and the following structure of resource requirements

$$\begin{aligned} N_k &= \{J_k, J_{m+k}\};\ p_k = k,\ p_{m+k} = m-k,\ 1 \leq k \leq m-1; \\ N_m &= \{J_m\};\ p_m = m; \\ N_{m+1} &= \{J_{2m}, J_{2m+1}, \ldots, J_{3m-1}\}, p_{2m+i-1} = 1,\ 1 \leq i \leq m. \end{aligned}$$

Additionally, if $r \geq 3$ define

$$N_k = \{J_{2m-2+k}\}; p_{2m-2+k} = m;\ m+2 \leq k \leq (r-1)m+1.$$

In the described instance, there are no non-resource jobs, i.e., $N_0 = \varnothing$.

If we apply the group technology approach, we get $q = (r-1)m+1$ composite jobs V_k, $1 \leq k \leq q$, with $p(V_k) = m$. In an optimal group technology schedule S_G^*, there will be a machine that processes r composite jobs, so that $C_{\max}(S_G^*) = rm$.

For this instance, $p(N) = ((r-1)m+1)m$. Due to (2), for any feasible schedule $C_{\max}(S) \geq (r-1)m+1 = q$. An optimal schedule S^* for which $C_{\max}(S^*) = (r-1)m+1$ can be constructed as follows. For each k, $1 \leq k \leq m-1$, machine M_k processes job $J_k \in N_k$ in the time interval $[0, k]$ and

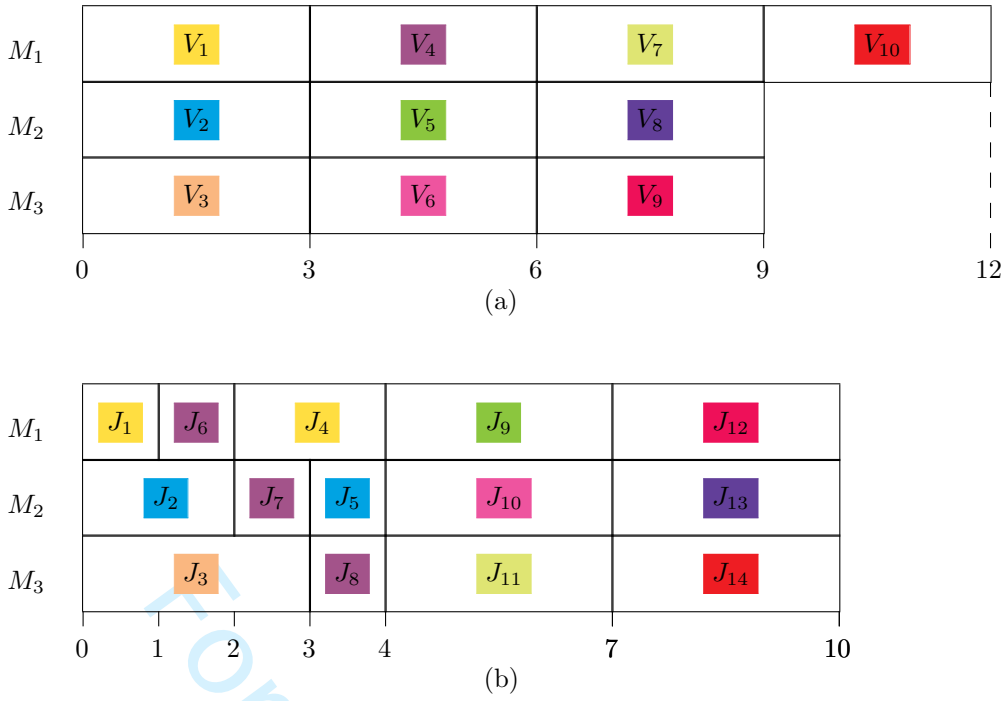


Figure 1. (a) an optimal group technology schedule S_G^*; (b) an overall optimal schedule S^*

the other job J_{m+k} of set N_k in the time interval $[k+1, m+1]$. Machine M_m processes job $J_m \in N_m$ in the time interval $[0, m]$. Additionally, each machine M_i, $1 \leq i \leq m$, processes job $J_{2m+i-1} \in N_{m+1}$ in the time interval $[i, i+1]$.

Besides, if $r \geq 3$, the remaining $(r-2)\,m$ jobs of duration m time units each are processed in the time intervals $[m+1, 2m+1]$, $[2m+1, 3m+1], \ldots,$ $[(r-2)\,m+1, (r-1)\,m+1]$, m jobs in each interval, each job on one machine. This schedule is feasible, since the jobs that require the same resource k, $1 \leq k \leq m$, are processed on machine M_k, while the jobs of set N_{m+1} are processed without overlapping. The makespan of the constructed schedule is $(r-1)\,m+1$, and it is optimal for problem $P\,|\mathrm{res} \cdot 111|\,C_{\max}$.

Comparing the expressions derived for $C_{\max}(S_G^*)$ and $C_{\max}(S^*)$, we deduce that the lemma holds. □

For illustration of the above proof, consider the following example of the input described above for the case of $m = 3$ and $r = 4$. In this case, there are $q = 10$ resources and $n = 14$ jobs as shown in Table 1, including the color scheme to denote each resource. In an optimal group technology schedule S_G^*, 4 of the 10 composite jobs are assigned to one of the machines, while each of the remaining two machines processes 3 composite jobs; see Figure 1(a). On the other hand, there is an overall optimal schedule S^*, built as described above and shown in Figure 1(b). Thus, for this instance $C_{\max}(S_G^*) = 12$ and $C_{\max}(S^*) = 10$, as required.

We now present a simple group technology algorithm that schedules the composite jobs in accordance with the famous LPT List scheduling algorithm by Graham (1969). Recall that the original Graham's algorithm is an approximation algorithm for problem $P\,|\,|\,C_{\max}$. It creates a list of jobs by sorting them in the LPT (Longest Processing Time) order, i.e., in non-increasing order of the processing times, and assigns the next job in the list to the first available machine. In the presence of the resource jobs, the algorithm can be stated as follows.

Algorithm GT

Step 1. Create composite jobs V_k, $1 \leq k \leq q$, and compute their durations $p(V_k) = p(N_k)$. If $N_0 \neq \varnothing$, replace the non-resource jobs by the composite jobs $V_{q+1}, \ldots, V_{q+n_0}$, where $n_0 = |N_0|$.

Step 2. If necessary, renumber the composite jobs in such a way that

$$p(V_1) \geq p(V_2) \geq \cdots \geq p(V_{q+n_0}).$$

Step 3. Considering the composite jobs in the order of their numbering, assign the next job to the first available machine, breaking ties arbitrary.

Step 4. Output the resulting schedule as schedule S_G^{LPT}.

Let us estimate the running time of Algorithm GT. Steps 1 and 2 require $O(n)$ and $O((q+n_0)\log(q+n_0))$ time, respectively. Step 3 can be implemented in $O((q+n_0)\log m)$ time by maintaining a non-decreasing sequence of the machine completion times in the current partial schedule. Thus, the overall time complexity of the algorithm is $O((q+n_0)(\log m+\log(q+n_0))+n)$.

For schedule S_G^{LPT}, let r be the position of the terminal job V_ℓ on the critical machine. Clearly, if $r=1$, then $C_{\max}\left(S_G^{LPT}\right)$ is equal to either the processing time of a non-resource job or to the total processing processing time of the set of jobs that require the same resource. Thus, in the case of $r=1$, schedule S_G^{LPT} corresponds to an optimal schedule S^*. The *a posteriori* behaviour of Algorithm GT is studied in the theorem below.

Theorem 3.2. *Suppose that schedule S_G^{LPT} found by Algorithm GT terminates by processing a composite job V_ℓ that is the rth on its machine, where $r\geq 2$. Then the bound*

$$\frac{C_{\max}\left(S_G^{LPT}\right)}{C_{\max}(S^*)}\leq\frac{rm}{(r-1)m+1} \tag{3}$$

holds.

Proof. Suppose that in schedule S_G^{LPT} the terminal job V_ℓ starts on its machine at time R_ℓ, so that

$$C_{\max}\left(S_G^{LPT}\right)=R_\ell+p(V_\ell).$$

Notice that due to the LPT List scheduling in Step 3 we have that

$$R_\ell\geq(r-1)p(V_\ell). \tag{4}$$

Suppose that the theorem does not hold, so that

$$R_\ell+p(V_\ell)>\frac{rm}{(r-1)m+1}C_{\max}(S^*),$$

which due to (2) implies that

$$R_\ell+p(V_\ell)>\frac{r}{(r-1)m+1}p(N). \tag{5}$$

Every machine is busy in the time interval $[0,R_\ell]$; otherwise, job V_ℓ would have started earlier than time R_ℓ. Thus, we deduce that

$$\begin{aligned}(m-1)R_\ell &< p(N)-(R_\ell+p(V_\ell))\\ &\leq \left(1-\frac{r}{(r-1)m+1}\right)p(N)=\frac{(r-1)(m-1)}{(r-1)m+1}p(N),\end{aligned}$$

i.e.,

$$R_\ell<\frac{r-1}{(r-1)m+1}p(N).$$

However, to guarantee (5) we must have $p(V_\ell)>\frac{1}{(r-1)m+1}p(N)$, which contradicts (4). □

Lemma 3.1 and Theorem 3.2 imply that Algorithm GT finds a group technology schedule which deviates from the overall optimum no more than an optimal group technology schedule does in the worst case.

Theorem 3.2 provides an *a posteriori* bound on the performance of Algorithm GT. To derive an *a priori* bound, without the knowledge of the position of the terminal job, define

$$R_{GT}=\frac{2m}{m+1}. \tag{6}$$

Corollary 3.3. *Given any instance of problem* $P\,|\mathrm{res}\cdot 111|\,C_{\max}$*, for schedule* S_G^{LPT} *found by Algorithm GT the bound*

$$\frac{C_{\max}\left(S_G^{LPT}\right)}{C_{\max}\left(S^*\right)} \leq R_{GT} \tag{7}$$

holds.

To see that Corollary 3.3 holds, notice that in schedule S_G^{LPT} the terminal job is either the only job on its machine and the schedule is optimal, or it is at least the second on its machine and (7) follows from (3) with $r = 2$.

Thus, there is a limit on the use the group technology approach: if we want an approximation algorithm which guarantees a worst-case ratio lower than R_{GT} defined by (6), we need to search for a heuristic schedule in which the jobs that require the same resource are not always kept as a single composite job.

Notice that Hebrard et al. (2016) present an algorithm, called Algorithm MaxLoad in their paper, that guarantees the same worst-case ratio of R_{GT} as Algorithm GT. However, Algorithm MaxLoad is more complicated than a group technology algorithm. Its running time is not explicitly estimated in Hebrard et al. (2016), but it is definitely larger than the running time of Algorithm GT. Besides, in Hebrard et al. (2016) the bound R_{GT} is only proved tight for $m = 2$. In our case, the tightness of R_{GT} follows from Lemma 3.1, the corresponding instance is obtained by setting $r = 2$.

4. Two-Machine Case

In this section, we focus on the two-machine case and present an algorithm that for problem $P2\,|\mathrm{res}\cdot 111|\,C_{\max}$ finds a heuristic schedule S_H such that the bound

$$\frac{C_{\max}\left(S_H\right)}{C_{\max}\left(S^*\right)} \leq \frac{6}{5} \tag{8}$$

holds.

Let the composite jobs V_k, $1 \leq k \leq q + n_0$, be as defined in Section 3. In what follows, we assume that $q + n_0 \geq 3$; otherwise, we have either $n_0 = 0$ and $q \leq 2$ or $n_0 = 1$ and $q = 1$, so that problem $P2\,|\mathrm{res}\cdot 111|\,C_{\max}$ is trivial.

The algorithm to be presented consists of two stages. The first stage is essentially a modified and simplified version of Algorithm GT. We present conditions which guarantee that a schedule found by that algorithm is of the required quality; otherwise, additional actions are taken in the second stage.

For a schedule S, let $C^{(i)}\left(S\right)$ denote the completion time of the last job assigned to machine M_i, $i \in \{1, 2\}$.

A composite job V_k, $1 \leq k \leq q$, made up of the resource jobs is called *splittable,* if $|N_k| \geq 2$, i.e., job V_k can be split in at least two (composite) jobs each formed of the jobs of set N_k. All other jobs are called non-splittable; in particular all jobs V_{q+i} associated with the non-resource jobs are non-splittable.

In the algorithm to be used in the first stage there is no need to have a complete LPT list of the composite jobs. In fact, it suffices to preorder only three longest composite jobs.

Identify three composite jobs with the longest processing times. If necessary renumber the composite jobs in such a way that these longest composite jobs get the numbers 1, 2 and 3 so that

$$p\left(V_1\right) \geq p\left(V_2\right) \geq p\left(V_3\right).$$

If $q + n_0 \geq 4$, the remaining composite jobs are numbered arbitrary, starting from V_4. This renumbering requires $O\left(n\right)$ time.

Consider the following approximation algorithm, which is an adapted version of Algorithm GT.

Algorithm A0

Step 1. From time zero, assign composite job V_1 to be processed on machine M_1 and the sequence of composite jobs (V_2, V_3) on machine M_2.

Step 2. If $n_0 + q \geq 4$, scanning the remaining composite jobs in an arbitrary order, assign the next job to the first available machine; ties are broken in favour of machine M_2. Call the resulting schedule S^0.

We now present conditions under which Algorithm A0 delivers a schedule of the required quality.

Theorem 4.1. *For schedule S^0 found by Algorithm A0 the bound*

$$\frac{C_{\max}\left(S^0\right)}{C_{\max}\left(S^*\right)} \leq \frac{6}{5} \tag{9}$$

holds and this bound is tight, provided that for schedule S^0 at least one of the following conditions is satisfied:

(a) *the critical machine is M_1;*
(b) *the critical machine is M_2, which processes more than two composite jobs;*
(c) *the critical machine is M_2, which processes only jobs V_2 and V_3 such that $p\left(V_2\right)+p\left(V_3\right) \leq \frac{3}{5}p\left(N\right)$;*
(d) *the critical machine is M_2, which processes only jobs V_2 and V_3, and each job V_1, V_2 and V_3 is not splittable.*

Proof. Suppose that condition (a) holds. Exclude from consideration the case that only job V_1 is processed on that machine; otherwise, S^0 is an optimal schedule. Thus, the terminal job is job V_ℓ, where $\ell \geq 4$, and therefore

$$p\left(V_\ell\right) \leq p\left(V_3\right) \leq p\left(V_2\right). \tag{10}$$

Let R_ℓ denote the start time of job V_ℓ. Thus,

$$\begin{aligned} C_{\max}\left(S^0\right) &= C^{(1)}\left(S^0\right) = R_\ell + p\left(V_\ell\right); \\ C^{(2)}\left(S^0\right) &\geq R_\ell. \end{aligned}$$

Suppose that the theorem does not hold, so that

$$R_\ell + p\left(V_\ell\right) > \frac{6}{5}C_{\max}\left(S^*\right) \geq \frac{3}{5}p\left(N\right). \tag{11}$$

This implies that

$$R_\ell \leq C^{(2)}\left(S^0\right) = p\left(N\right) - C^{(1)}\left(S^0\right) \leq \frac{2}{5}p\left(N\right), \tag{12}$$

so that in order satisfy (11) we must have

$$p\left(V_\ell\right) > \frac{1}{5}p\left(N\right). \tag{13}$$

However,

$$C^{(2)}\left(S^0\right) \geq p\left(V_2\right) + p\left(V_3\right)$$

which contradicts (12) due to (10).

From now on, assume that in schedule S^0 the critical machine is M_2. No matter how many jobs are assigned to machine M_2 for the terminal job V_ℓ the inequalities (10) hold.

Under condition (b), the terminal job is job V_ℓ, where $\ell \geq 4$. Thus,

$$\begin{aligned} C_{\max}\left(S^0\right) &= C^{(2)}\left(S^0\right) = R_\ell + p\left(V_\ell\right); \\ C^{(1)}\left(S^0\right) &\geq R_\ell. \end{aligned}$$

Assume that (11) holds. This implies that both

$$R_\ell \leq C^{(1)}\left(S^0\right) = p\left(N\right) - C^{(2)}\left(S^0\right) \leq \frac{2}{5}p\left(N\right) \tag{14}$$

and (13) are true. However,

$$R_\ell \geq p\left(V_2\right) + p\left(V_3\right)$$

Resource	Sets	Durations
1	$N_1 = \{J_1\}$	$p_1 = 4$
2	$N_2 = \{J_2\}$	$p_2 = 2$
3	$N_3 = \{J_3\}$	$p_3 = 2$
4	$N_4 = \{J_4, J_5\}$	$p_4 = p_5 = 1$

Table 2. The input for $m = 2$

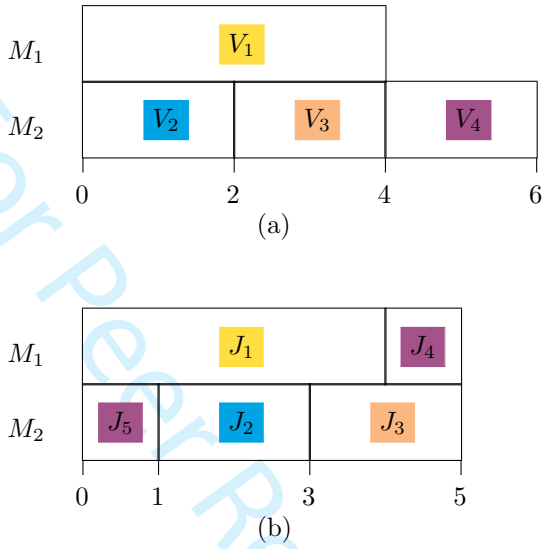


Figure 2. (a) schedule S^0; (b) optimal schedule S^*

which contradicts (14), due to (10).

Under condition (c) we have

$$C^{(1)}\left(S^0\right) \leq C^{(2)}\left(S^0\right) = C_{\max}\left(S^0\right) \leq \frac{3}{5}p\left(N\right) \leq \frac{6}{5}C_{\max}\left(S^*\right).$$

Finally, under condition (d), in any optimal schedule S^* two of the jobs V_1, V_2 and V_3 are assigned to the same machine, so that $C_{\max}\left(S^0\right) = C^{(2)}\left(S^0\right) = p\left(V_2\right) + p\left(V_3\right) \leq C_{\max}\left(S^*\right)$ and S^0 is an optimal schedule.

To see that the bound (9) is tight, consider an instance with five jobs and four resources presented in Table 2.

Algorithm A0 manipulates four composite jobs, such that

$$p\left(V_1\right) = 4, p\left(V_2\right) = p\left(V_3\right) = p\left(V_4\right) = 2.$$

In schedule S^0 the composite job V_1 is assigned to machine M_1, while the sequence (V_2, V_3, V_4) is assigned to machine M_2, so that $C_{\max}\left(S^0\right) = 6$; see Figure 2(a).

In order to construct an optimal schedule S^* of the original jobs, assign the sequence of jobs (J_1, J_4) to machine M_1 and the sequence of jobs (J_5, J_2, J_3) to machine M_2. Such a schedule is clearly feasible since the jobs J_4 and J_5 do not overlap. The makespan is equal to 5, which corresponds to the lower bound (2); see Figure 2(b). Thus, (9) holds as equality. □

We are left with the situation that for schedule S^0 none of the conditions of Theorem 4.1 holds. In such a schedule the critical machine is M_2 and the terminal job is V_3. Let $\tilde{V}$ denote the composite job (possibly, dummy) formed of jobs processed on M_1 after job V_1. The structure of schedule S^0 that does not satisfy the conditions of Theorem 4.1 is shown in Figure 3. In that figure and in the subsequent illustrative figures we assume that each composite job V_1, V_2 and V_3 is formed of the jobs that require

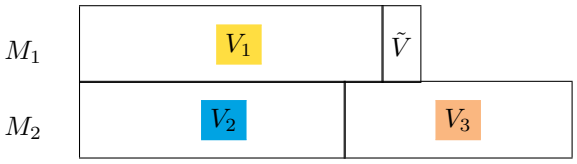


Figure 3. Schedule S^0 that does not satisfy the conditions of Theorem 4.1

a resource. In general, the composite job $\tilde{V}$ is a block of both the resource and the non-resource jobs, so that we do not highlight it by color.

For schedule S^0 we have that

$$\begin{aligned} C_{\max}\left(S^0\right) &= C^{(2)}\left(S^0\right) = p\left(V_2\right) + p\left(V_3\right) \geq p\left(V_1\right); \\ C^{(1)}\left(S^0\right) &= p\left(V_1\right) + p\left(\tilde{V}\right). \end{aligned}$$

At least one of the composite jobs V_1, V_2 and V_3 is splittable and

$$C^{(2)}\left(S^0\right) = p\left(V_2\right) + p\left(V_3\right) > \frac{6}{5} C_{\max}\left(S^*\right) \geq \frac{3}{5} p\left(N\right). \tag{15}$$

Since $C^{(1)}\left(S^0\right) = p\left(N\right) - C^{(2)}\left(S^0\right)$, we have that

$$C^{(1)}\left(S^0\right) < \frac{2}{5} p\left(N\right),$$

which implies that

$$p\left(V_2\right) \leq p\left(V_1\right) \leq C^{(1)}\left(S^0\right) < \frac{2}{5} p\left(N\right).$$

Since $p\left(V_3\right) = p\left(N\right) - p\left(V_2\right) - C^{(1)}\left(S^0\right)$, we obtain

$$\frac{1}{5} p\left(N\right) < p\left(V_3\right).$$

Additionally, notice that $p\left(\tilde{V}\right) \leq \frac{1}{10} p\left(N\right)$; otherwise, $p\left(V_1\right) = p\left(N\right) - C^{(2)}\left(S^0\right) - p\left(\tilde{V}\right) < \frac{3}{10} p\left(N\right)$, which leads to a contradiction $p\left(V_2\right) + p\left(V_3\right) \leq 2p\left(V_1\right) \leq \frac{3}{5} p\left(N\right)$.

Thus, we need additional actions if at least one of the composite jobs V_1, V_2 and V_3 is splittable, (15) holds and additionally each of the inequalities

$$\frac{1}{5} p\left(N\right) < p\left(V_3\right) \leq p\left(V_2\right) \leq p\left(V_1\right) < \frac{2}{5} p\left(N\right); \tag{16}$$

$$p\left(V_1\right) + p\left(\tilde{V}\right) < \frac{2}{5} p\left(N\right); \tag{17}$$

$$p\left(\tilde{V}\right) \leq \frac{1}{10} p\left(N\right). \tag{18}$$

holds.

The algorithm below performs further processing by partitioning either one or two splittable composite jobs.

Algorithm Split

Step 1. If among V_1, V_2 and V_3 there exists a splittable composite job which contains no original jobs longer than $\frac{1}{5} p\left(N\right)$, then go to Step 2, otherwise go to Step 3.

Step 2. Renumber the composite jobs V_1, V_2 and V_3 using a bijection $\varphi : \{1, 2, 3\} \rightarrow \{\lambda, \mu, \nu\}$ in such a way that the splittable job with no original jobs longer than $\frac{1}{5} p\left(N\right)$ becomes V_λ. If necessary, renumber the actual jobs in such a way that the jobs of set N_λ are numbered by the integers starting from 1. Considering the jobs of N_λ in the order of their numbering, find the job J_u such

that

$$\begin{aligned} p(V_\mu) + p\left(\tilde{V}\right) + \sum_{j=1}^{u} p_j &\leq \frac{3}{5} p(N); \\ p(V_\mu) + p\left(\tilde{V}\right) + \sum_{j=1}^{u+1} p_j &> \frac{3}{5} p(N). \end{aligned} \tag{19}$$

Define the composite jobs W' and W'' comprised of the jobs of set $\{J_1, J_2, \ldots, J_u\}$ and of set $N_\lambda \backslash \{J_1, J_2, \ldots, J_u\}$, respectively. From time zero, assign the sequence $\left(V_\mu, \tilde{V}, W'\right)$ to be processed on machine M_1 and the sequence (W'', V_ν) on machine M_2, provided that the composite job W' starts on M_1 as early as possible, i.e., at time $\max\left\{p(V_\mu) + p\left(\tilde{V}\right), p(W'')\right\}$. Output the resulting schedule S^H.

Step 3. If exactly one of V_1, V_2 and V_3 is splittable, go to Step 4; if two of these composite jobs are splittable, go to Step 5; if all three composite jobs are splittable, go to Step 6.

Step 4. Renumber the composite jobs V_1, V_2 and V_3 using a bijection $\varphi : \{1, 2, 3\} \to \{\lambda, \mu, \nu\}$ in such a way that the splittable composite job is V_λ, while the non-splittable jobs are V_μ and V_ν, where $p(V_\mu) \geq p(V_\nu)$. Create two schedules, S_1^H and S_2^H. In schedule S_1^H, machine M_1 processes the sequence $\left(V_\lambda, \tilde{V}\right)$, while machine M_2 processes the sequence (V_μ, V_ν). For schedule S_2^H, identify job J_w, the longest job contained in V_λ and determine W_λ as the composite job formed of the jobs of set $N_\lambda \backslash \{J_w\}$. In schedule S_2^H, machines M_1 and M_2 process the sequences (J_w, V_ν) and $\left(V_\mu, \tilde{V}, W_\lambda\right)$, respectively, where the composite job W_λ starts as early as possible, i.e., at time $\max\left\{p_w, p(V_\mu) + p\left(\tilde{V}\right)\right\}$. Output the better of the two schedules S_1^H and S_2^H as schedule S^H.

Step 5. Renumber the composite jobs V_1, V_2 and V_3 using a bijection $\varphi : \{1, 2, 3\} \to \{\lambda, \mu, \nu\}$ in such a way that the non-splittable job is V_ν, while the splittable composite jobs are V_λ and V_μ; additionally if J_{w_1} is the longest job contained in V_λ and J_{w_2} is the longest job contained in V_μ under the chosen numbering the inequalities

$$\frac{1}{5} p(N) < p_{w_1} \leq p_{w_2} \tag{20}$$

hold. Let W_λ and W_μ be the composite jobs formed of the jobs of set $N_\lambda \backslash \{J_{w_1}\}$ and of set $N_\mu \backslash \{J_{w_2}\}$, respectively. Create two schedules, S_1^H and S_2^H. In schedule S_1^H machines M_1 and M_2 process the sequences (J_{w_1}, J_{w_2}) and $\left(W_\mu, V_\nu, \tilde{V}, W_\lambda\right)$, respectively, where the composite job W_λ starts as early as possible, i.e., at time $\max\left\{p_{w_1}, p\left(W_\mu \cup V_\nu \cup \tilde{V}\right)\right\}$. In schedule S_2^H machines M_1 and M_2 process the sequences $\left(J_{w_1}, V_\nu, \tilde{V}\right)$ and (V_μ, W_λ), where the composite job W_λ starts as early as possible, i.e., at time $\max\{p_{w_1}, p(V_\mu)\}$. Output the better of the two schedules S_1^H and S_2^H as schedule S^H.

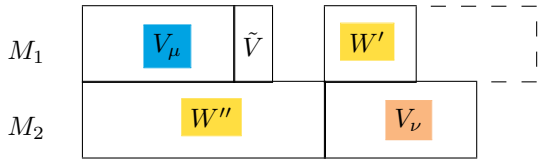
Step 6. Renumber the composite jobs V_1, V_2 and V_3 using a bijection $\varphi : \{1, 2, 3\} \to \{\lambda, \mu, \nu\}$ that the longest jobs in sets N_λ, N_μ and N_ν are the jobs J_{w_1}, J_{w_2} and J_{w_3}, respectively, where $J_{w_3} \geq \max\{J_{w_1}, J_{w_2}\}$ and the inequalities (7) hold. Let W_λ and W_μ be the composite jobs formed of the jobs of set $N_\lambda \backslash \{J_{w_1}\}$ and of set $N_\mu \backslash \{J_{w_2}\}$, respectively. Output schedule S^H, which is essentially schedule S_1^H created in Step 5, provided that the composite job W_λ starts at time $p\left(W_\mu \cup V_\nu \cup \tilde{V}\right)$.

It is clear that the running time of Algorithm Split is linear in n. We now prove that in any case for schedule S^H found by Algorithm Split the bound

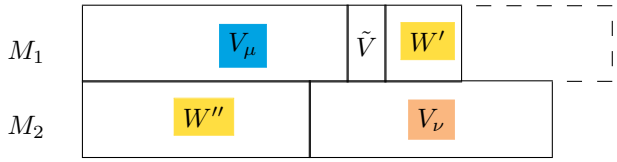
$$\frac{C_{\max}\left(S^H\right)}{C_{\max}\left(S^*\right)} \leq \frac{6}{5} \tag{21}$$

holds.

Lemma 4.2. *For schedule S^H found in Step 2 of Algorithm Split the bound (21) holds.*



(a)



(b)

Figure 4. Schedule S^H found in Step 2 of Algorithm Split; $\lambda = 1, \mu = 2, \nu = 3$

Proof. Consider machine M_1 first. Notice that if $p(W'') > p(V_\lambda) + p\left(\tilde{V}\right)$, then $C^{(1)}\left(S^H\right) = p(W'') + p(W') = p(V_\lambda) \leq C_{\max}(S^*)$; see Figure 4 (a). Otherwise, for machine M_1 by construction we have that

$$C^{(1)}\left(S^H\right) = p(V_\mu) + p\left(\tilde{V}\right) + p(W') \leq \frac{3}{5}p(N);$$

see Figure 4 (b).

On the other hand, in any case for machine M_2 we have that

$$\begin{aligned} C^{(2)}\left(S^H\right) &= p(W'') + p(V_\nu) = \left(p(N_\lambda) - \sum_{j=1}^{u} p_j\right) + p(V_\nu) \\ &= (p(N_\lambda) + p(V_\nu)) - \sum_{j=1}^{u+1} p_j + p_{u+1} = \\ &= p(N) - \left(p(V_\mu) + p\left(\tilde{V}\right)\right) - \sum_{j=1}^{u+1} p_j + p_{u+1}. \end{aligned}$$

Applying (19) and the inequality $p_{u+1} \leq \frac{1}{5}p(N)$, we deduce

$$\begin{aligned} C^{(2)}\left(S^H\right) &= p(N) - \left(p(V_\mu) + p\left(\tilde{V}\right) + \sum_{j=1}^{u+1} p_j\right) + p_{u+1} \\ &< \frac{2}{5}p(N) + \frac{1}{5}p(N) = \frac{3}{5}p(N), \end{aligned}$$

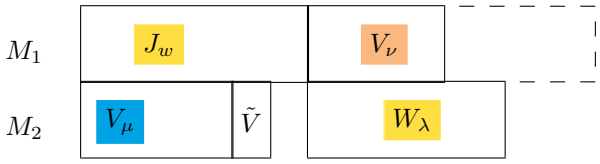
as required. ∎

Lemma 4.3. *For schedule S^H found in Step 4 of Algorithm Split the bound (21) holds.*

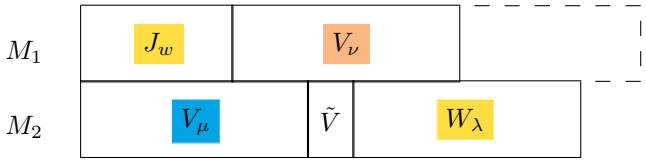
Proof: Notice that job J_w and the composite jobs V_μ and V_ν are non-splittable, i.e., there exists an optimal schedule S^* in which at least two of these jobs are assigned to the same machine.

If there exists an optimal schedule S^* in which jobs V_μ and V_ν are assigned to the same machine, then for schedule S_1^H

$$C^{(2)}\left(S_1^H\right) = p(V_\mu) + p(V_\nu) \leq C_{\max}(S^*).$$



(a)



(b)

Figure 5. Schedule S_1^H found in Step 4 of Algorithm Split; $\lambda = 1, \mu = 2, \nu = 3$

On the other hand, it follows from (17) that

$$C^{(1)}\left(S_1^H\right) = p\left(V_\lambda\right) + p\left(\tilde{V}\right) \leq p\left(V_1\right) + p\left(\tilde{V}\right) \leq \frac{2}{5}p\left(N\right).$$

For illustration, notice that for $\lambda = 1, \mu = 2$ and $\nu = 3$ schedule S_1^H coincides with schedule S^0, see Figure 3.

Assume now that in any optimal schedule jobs V_μ and V_ν are assigned to different machines. Then in any optimal schedule job J_w is on the same machine with one of the jobs V_μ and V_ν. Since $p\left(V_\nu\right) \leq p\left(V_\mu\right)$, it follows that $p_w + p\left(V_\nu\right) \leq C_{\max}\left(S^*\right)$. Then for S_2^H we have that $C^{(1)}\left(S_2^H\right) = p_w + p\left(V_\nu\right) \leq C_{\max}\left(S^*\right)$. On the other hand, either

$$C^{(2)}\left(S_2^H\right) = p_w + p\left(W_\lambda\right) = p\left(N_\lambda\right) \leq C_{\max}\left(S^*\right),$$

as in Figure 5(a) and S_2^H is an optimal schedule or

$$C^{(2)}\left(S_2^H\right) = p\left(V_\mu\right) + p\left(\tilde{V}\right) + p\left(W_\lambda\right),$$

as in Figure 5(b). In the latter case, applying (16), (17) and the inequality $p_w > \frac{1}{5}p\left(N\right)$ we deduce

$$\begin{aligned} C^{(2)}\left(S_2^H\right) &\leq \left(p\left(V_1\right) + p\left(\tilde{V}\right)\right) + \left(p\left(V_\lambda\right) - \frac{1}{5}p\left(N\right)\right) \\ &\leq \frac{2}{5}p\left(N\right) + \frac{1}{5}p\left(N\right) = \frac{3}{5}p\left(N\right), \end{aligned}$$

as required. □

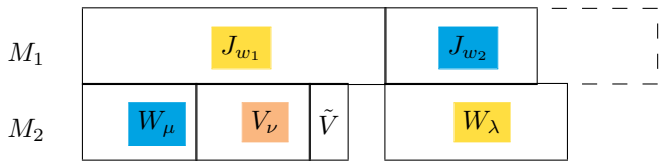
Lemma 4.4. *For schedule S^H found in Step 5 of Algorithm Split the bound (21) holds.*

Proof. Notice that jobs J_{w_1} and J_{w_2} and the composite job V_ν are non-splittable, i.e., there exists an optimal schedule S^* in which at least two of these jobs are assigned to the same machine.

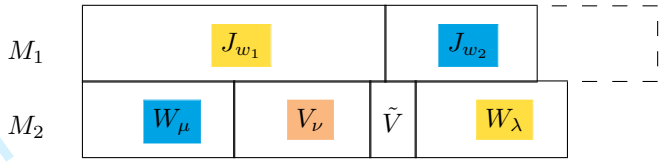
If for any optimal schedule S^* we have that $p_{w_1} + p\left(V_\nu\right) > C_{\max}\left(S^*\right)$ then in any optimal schedule job V_ν cannot be assigned to the same machine with any of the jobs $J_{w_1}$and J_{w_2}. Thus, in any optimal schedule the jobs J_{w_1} and J_{w_2} are assigned to the same machine, so that $p_{w_1} + p_{w_2} \leq C_{\max}\left(S^*\right)$.

Let us analyze schedule S_1^H. It follows from (16) that

$$p\left(W_\mu\right) = p\left(N_\mu\right) - p_{w_2} < \frac{2}{5}p\left(N\right) - \frac{1}{5}p\left(N\right) < p_{w_1} \tag{22}$$



(a)



(b)

Figure 6. Schedule S_1^H found in Step 5 of Algorithm Split; $\lambda = 1, \mu = 2, \nu = 3$

and job J_{w_2} may start on machine M_1 at time p_{w_1}. Thus, for schedule S_1^H we have that

$$C^{(1)}\left(S_1^H\right) = p_{w_1} + p_{w_2} \leq C_{\max}\left(S^*\right).$$

On the other hand, it follows that either $C^{(2)}\left(S_1^H\right) = p_{w_1} + p\left(W_\lambda\right) = p\left(N_\lambda\right) \leq C_{\max}\left(S^*\right)$ as in Figure 6(a), so that S_1^H is an optimal schedule or

$$C^{(2)}\left(S_1^H\right) = p\left(W_\mu \cup V_\nu \cup \tilde{V}\right) \leq p\left(N\right) - \left(p_{w_1} + p_{w_2}\right) \leq \frac{3}{5}p\left(N\right),$$

as in Figure 6(b).

Assume now that $p_{w_1} + p\left(V_\nu\right) \leq C_{\max}\left(S^*\right)$ and analyze schedule S_2^H. Due to (18) we deduce that

$$C^{(1)}\left(S_2^H\right) = p_{w_1} + p\left(V_\nu\right) + p\left(\tilde{V}\right) \leq C_{\max}\left(S^*\right) + \frac{1}{10}p\left(N\right) \leq \frac{6}{5}C_{\max}\left(S^*\right).$$

On the other hand, either

$$C^{(2)}\left(S_2^H\right) = p_{w_1} + p\left(W_\lambda\right) = p\left(N_\lambda\right) \leq C_{\max}\left(S^*\right),$$

as in Figure 7(a), so that S_2^H is an optimal schedule or

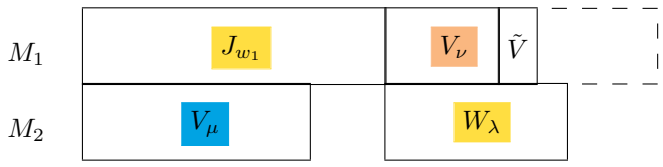
$$C^{(2)}\left(S_2^H\right) = p\left(V_\mu\right) + p\left(W_\lambda\right),$$

as in Figure 7(b), so that applying (16), (17) and (20) we deduce

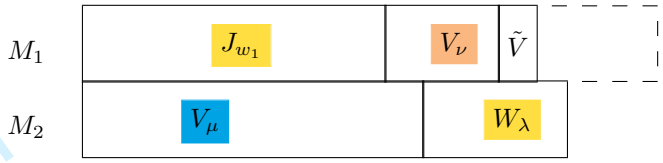
$$\begin{aligned} C^{(2)}\left(S_2^H\right) &\leq p\left(V_1\right) + \left(p\left(N_\lambda\right) - \frac{1}{5}p\left(N\right)\right) \\ &\leq \frac{2}{5}p\left(N\right) + \frac{1}{5}p\left(N\right) = \frac{3}{5}p\left(N\right), \end{aligned}$$

as required. □

Lemma 4.5. *For schedule S^H found in Step 6 of Algorithm Split the bound (21) holds.*



(a)



(a)

Figure 7. Schedule S_2^H found in Step 5 of Algorithm Split; $\lambda = 1, \mu = 2, \nu = 3$

Proof. In Step 6, the instance contains three jobs J_{w_1}, J_{w_2} and J_{w_3}, each longer than $\frac{1}{5}p(N)$. There exists an optimal schedule S^*, in which at least two of these jobs are assigned to the same machine. Since $p_{w_3} \geq \max\{p_{w_1}, p_{w_2}\}$, it follows that $p_{w_1} + p_{w_2} \leq C_{\max}(S^*)$.

As in the proof of Lemma 4.4, the inequalities (22) hold, so that job J_{w_2} starts on M_1 at time p_{w_1}, so that $C^{(1)}\left(S^H\right) = p_{w_1} + p_{w_2} \leq C_{\max}(S^*)$.

By construction, $p(V_\nu) \geq p_{w_3} \geq \max\{p_{w_1}, p_{w_2}\}$ and we deduce that $p\left(W_\mu \cup \tilde{V} \cup V_\nu\right) \geq p_{w_1}$, so that W_λ can start at time $p\left(W_\mu \cup \tilde{V} \cup V_\nu\right)$. See Figure 6(b). □

Putting together the statements proved for problem $P2\,|\text{res} \cdot 111|\,C_{\max}$, we obtain the following theorem.

Theorem 4.6. *For problem $P2\,|\text{res} \cdot 111|\,C_{\max}$, a schedule S^H for which the bound (21) holds and is tight can be found in $O(n)$ time by running Algorithm A0 and, if required, Algorithm Split.*

5. Conclusion

In this paper, we analyze the power of the group technology approach applied to the problem of minimizing the makespan on parallel identical machines subject to resource constraints. Both *a posteriori* and *a priori* bounds are derived on the performance of the LPT group technology algorithm. For two machines, a 6/5-approximation algorithm is designed.

It remains to be seen whether the two-machine problem admits a fully polynomial-time approximation scheme. In an attempt to answer this question we should start with trying to design a pseudopolynomial-time algorithm for the problem. Design of approximation algorithms for problem $P\,|\text{res} \cdot 111|\,C_{\max}$ and its variants with a fixed number of machines that would deliver a better performance guarantee that those established in this paper is also of interest. The search for approximation algorithms and schemes for other scheduling problems with arbitrary processing times under resource constraints is a promising direction of research.

References

Błażewicz, J., Brauner, N., Finke, G. (2004). Scheduling with discrete resource constraints. In: J. Y.-T. Leung (ed.) *Handbook of Scheduling: Algorithms, Models and Performance Analysis* (pp. 23-1–23-18). Boca Raton: Chapman & Hall/CRC.

Błażewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Sterna, M., Weglarz, J. (2019). Scheduling under resource constraints. In: *Handbook of Scheduling. From Theory to Practice* (Chapter 13, pp. 475-525), Cambridge: Spinger.

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

Chen, B. (1993). A note on LPT scheduling. *Operations Research Letters*, *14*, 139–142.

Chen, B. (2004). Parallel machine scheduling for early completion. In J.Y-T. Leung (ed.) *Handbook of Scheduling: Algorithms, Models and Performance Analysis* (pp. 9-175–9-184). Boca Raton: Chapman & Hall/CRC..

Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research, 230*, 449–463.

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

Graham, R. L. (1966). Bounds for certain multiprocessor anomalies. *Bell System Technical Journal, 45*, 1563–1581.

Graham, R. L. (1969). Bounds on multiprocessor timing anomalies. *SIAM Journal on Applied Mathematics*, *17*, 416–429.

Hebrard, E., Hugueta, M.-J., Jozefowiez, N., Maillard, A., Pralet, C., & Verfaillie, G. (2016). Approximation of the parallel machine scheduling problem with additional unit resources. *Discrete Applied Mathematics, 215*, 126–135.

Janssen, T., Swennenhuis, C., Bitar, A., Bosman, T., Gijswijt, D., van Iersel, L., Dausère-Pérèz, S., & Yugma, C. (2018) Parallel machine scheduling with a single resource per job. arXiv:1809.05009v3.

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

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

Lawler, E. L. (1978) Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics, 2,* 75–90.

Strusevich, V. A. (2000) Group technology approach to the open shop scheduling problem with batch setup times. *Operations Research Letters, 26,* 181–192.