

**University of Greenwich**  
**School of Computing and Mathematical Sciences**  
**Statistics and Operational Research Group**

**Report SORG-02/2012**

**Parallel Machine Scheduling:  
Impact of Adding an Extra Machine**

**Kabir Rustogi**

**Vitaly A. Strusevich**

# Parallel Machine Scheduling: Impact of Adding an Extra Machine

Kabir Rustogi and Vitaly A. Strusevich\*

School of Computing and Mathematical Sciences, University of Greenwich,  
Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, U.K.

e-mail: {K.Rustogi,V.Strusevich}@greenwich.ac.uk

## Abstract

We consider the classical scheduling problems of processing jobs on parallel identical machines to minimize (i) the makespan (the maximum completion time) or (ii) the total flow time (the sum of the completion times). The focus of this study is on the impact that an additional machine, if added to the system, may have. We measure such a machine impact by the ratio of the value of the function computed without an extra machine to the one with an additional machine. We give tight bounds on the machine impact for the problem of minimizing the makespan, for both the preemptive and non-preemptive versions, as well as for the problem of minimizing the total flow time. We also present a polynomial-time algorithm that determines an optimal number of machines, provided that each machine incurs a cost and the objective function captures the trade-off between the cost of the machines and a scheduling objective.

Subject classifications: Production/scheduling: approximations/heuristic. Production/scheduling: deterministic: sequencing: multiple machine

Area of review: Optimization

---

\*Corresponding author; Tel: +44-2083318662, Fax: +44-2083318665.

# 1 Introduction

In this paper, we address two scheduling problems on identical parallel machines, one to minimize the maximum completion time and the other to minimize the sum of the completion times. Both problems belong to the core of scheduling theory and since the 1950s have initiated most important research contributions.

In the problems under consideration, we are given a set of jobs, each of which can be processed by any of the available machines. The machines are identical, i.e., the processing time of a job does not depend on the machine assignment decisions. If no preemption is allowed, each job is assigned to exactly one machine and is processed on that machine without interruption. In a preemptive schedule, the processing of a job on a machine can be interrupted at any time and then resumed later on any other machine, provided that a job is not processed on two or more machines at a time and the total duration of its processing is equal to the given processing time.

In scheduling, an objective function to be minimized is normally a non-decreasing function that depends on the completion times of the jobs. The most popular functions are the *makespan*, i.e., the maximum completion time and *the total flow time*, i.e., the sum of the completion times.

The main aspect of this study is to investigate the influence that an additional machine may have on the objective function. We measure this influence by a *machine impact*, which is defined as a ratio of the objective function value computed without using an extra machine over the function value computed with an additional machine.

In this paper, we give tight bounds on the machine impact for the problem of minimizing the makespan, for both the preemptive and non-preemptive versions, as well as for the problem of minimizing the total flow time. For the latter problem only the non-preemptive version is considered, since, as shown by McNaughton (1959), for this objective function there is no advantage in allowing preemption.

We believe that the machine impact is an important characteristic of the processing system. In manufacturing, the decisions on adding a machine-tool to the existing park of similar equipment are often considered. In computing, parallel processors can be added to a computer system to boost its performance.

In reality however, adding a machine cannot be seen free. Introducing more machines reduces a scheduling objective function but may be unacceptable due to a high cost of machine usage. In this paper, we also study the problem of the cost-effective choice of the number of the machines. The total cost function captures the trade-off between the gain in reducing the value of a scheduling performance measure and a loss associated with increasing the number of machines. The study of online versions of the problems of the cost-optimal selection of the machines has been initiated by Imreh and Noga (1999); we give a brief review of this line of research in Section 4.

Recently, a link between the problem of minimizing the total flow time and organizing a safe pickup of the employees from the off-shore oil-mining installations has been established by Qian et al. (2011). Under this interpretation, the jobs are installations, the machines are helicopter flights, the processing time of job is the number of people to be picked up from an installation, and the objective function measures the total risk, understood as the total number of passengers exposed to landings and takeoffs (which are known to be the most dangerous parts of a helicopter's flight). For this application, the machine impact is related

to the risk reduction due to an additional flight. The problem of determining a cost-effective number of flights for a safe pickup is also of interest.

The remainder of this paper is organized as follows. In Section 2, we give tight estimates of the machine impact for the problem of minimizing the makespan. In particular, for the non-preemptive version of the problem we show that the impact factor is 2. This follows from a more general result that delivers a tight bound on the ratio of the makespan of a list schedule for a given number of machines to the optimal makespan on any number of machines. Section 3 handles the machine impact for the problem of minimizing the total flow time. Here we start with the problem with unit processing times, derive both upper and lower bounds on the machine impact, and show that the upper bound in fact holds for instances with arbitrary processing times. The cost-effective choice of the number of machines for the makespan is addressed in Section 4. We present a linear time algorithm for the preemptive version and an approximation algorithm accompanied by its worst-case analysis for the non-preemptive version. A similar problem with the total flow time as a scheduling measure is studied in Section 5. We establish a form of discrete convexity of the total cost function and this results into a fast exact algorithm. Concluding remarks are contained in Section 6. The proofs of some statements have been moved to appendices in order not to break the flow of logic of the paper. Some of these results may be of independent interest; in particular Appendix B contains improved results on the power of preemption on parallel machines to minimize the makespan.

## 2 Estimating Machine Impact: Makespan

Formally, in all problems in this paper, we are given the jobs of set  $N = \{1, 2, \dots, n\}$  and  $m$  parallel identical machines  $M_1, M_2, \dots, M_m$ , where  $m \geq 2$  and  $n \geq m$ . No matter to which machine a job  $j \in N$  is assigned, its processing time is equal to  $p_j$ . We denote the sum of all processing times by  $P$  and the largest processing time by  $p_{\max}$ .

Using standard scheduling notation (see, e.g., Lawler et al. (1993)), the scheduling problem of finding a non-preemptive schedule  $S_{np}^*(m)$  that minimizes an objective function  $F$  on  $m$  parallel identical machines is denoted by  $Pm || F$ . The problems of finding an optimal preemptive schedule  $S_p^*(m)$  is denoted by  $Pm |pmtn| F$ .

In this section, we focus on minimizing the *makespan*, i.e., the maximum completion time. For a schedule  $S$ , the makespan is denoted by  $C_{\max}(S)$ . Thus,  $F = C_{\max}$ , and the problems under consideration are  $Pm || C_{\max}$  and  $Pm |pmtn| C_{\max}$ .

Here, we do not quote individual results on these two problems, but refer to a recent focused survey on parallel machine scheduling with makespan as the objective function, by Chen (2004). It suffices to mention that the non-preemptive version of the problem is NP-hard for each  $m \geq 2$ , while its preemptive counterpart is polynomially solvable by a well-known ‘wrap-around’ algorithm by McNaughton (1959).

It is clear that for any schedule  $S$  on  $m$  machines, preemptive or not, the makespan cannot be less than the largest processing time, i.e.,

$$C_{\max}(S(m)) \geq p_{\max} \tag{1}$$

and less than the average machine load

$$C_{\max}(S(m)) \geq \frac{P}{m}. \tag{2}$$

For problem  $Pm |pmtn| C_{\max}$ , McNaughton's algorithm requires  $O(n)$  time and finds an optimal schedule  $S_p^*(m)$  with  $C_{\max}(S_p^*(m)) = \max\{p_{\max}, \frac{P}{m}\}$ . For the non-preemptive case, the above bounds need not be tight.

For problem  $Pm |pmtn| C_{\max}$ , it is very easy to compute the machine impact  $I(m)$ , i.e., the ratio

$$I(m) = \frac{C_{\max}(S_p^*(m))}{C_{\max}(S_p^*(m+1))}.$$

Indeed, if  $C_{\max}(S_p^*(m)) = p_{\max} \geq \frac{P}{m} > \frac{P}{m+1}$  then  $C_{\max}(S_p^*(m+1)) = p_{\max}$  and  $I(m) = 1$ . On the other hand, if  $C_{\max}(S_p^*(m)) = \frac{P}{m}$  then due to  $C_{\max}(S_p^*(m+1)) \geq \frac{P}{m+1}$ , we obtain that for problem  $Pm |pmtn| C_{\max}$

$$I(m) \leq \frac{m+1}{m}. \quad (3)$$

This bound is tight, which can be seen by considering an instance of the problem with  $m(m+1)$  jobs of unit duration each. In this case,  $C_{\max}(S_p^*(m)) = m+1$  and  $C_{\max}(S_p^*(m+1)) = m$ .

Notice that the bound (3) will appear again in Section 3, as an upper bound on the machine impact for the problem of minimizing the total flow time.

Now we pass to considering the machine impact for problem  $Pm || C_{\max}$ . Here the situation is different because the optimal values of makespan are non-available. Still, a tight upper bound on the machine impact

$$I(m) = \frac{C_{\max}(S_{np}^*(m))}{C_{\max}(S_{np}^*(m+1))}$$

can be derived.

Recall that for problem  $Pm || C_{\max}$ , Graham (1966) introduces the *List Scheduling* algorithm (later referred to in the paper as Algorithm LS), which delivers a heuristic schedule  $S_{LS}(m)$  reasonably close to the optimum. In the most general setting of Algorithm LS, the jobs are arranged in an arbitrary sequence (a list) and every time that a machine becomes available, the next job from the list is assigned to it. The classical worst-case analysis result by Graham (1966) states that

$$\frac{C_{\max}(S_{LS}(m))}{C_{\max}(S_{np}^*(m))} \leq 2 - \frac{1}{m},$$

and this bound is tight.

To estimate the machine impact for problem  $Pm || C_{\max}$  we will use the list scheduling approach.

Our goal is to prove

**Theorem 1** *For problem  $Pm || C_{\max}$ , the following bound*

$$I(m) = \frac{C_{\max}(S_{np}^*(m))}{C_{\max}(S_{np}^*(m+1))} \leq \frac{C_{\max}(S_{LS}(m))}{C_{\max}(S_{np}^*(m+1))} \leq 2 \quad (4)$$

*holds and this bound is tight.*

We start with the following general statement on the behaviour of the list scheduling algorithm.

**Theorem 2** For problem  $Pm \parallel C_{\max}$ , let  $S_{LS}(m)$  be a schedule found by Algorithm LS. Then for any number  $m' \geq 1$  of machines the bound

$$\frac{C_{\max}(S_{LS}(m))}{C_{\max}(S_{np}^*(m'))} \leq 1 + \frac{m' - 1}{m} \quad (5)$$

holds and this bound is tight.

**Proof:** For problem  $Pm \parallel C_{\max}$ , consider a schedule  $S_{LS}(m)$  found by Algorithm LS. Let  $k$  be the job that terminates that schedule, i.e.,  $C_{\max}(S_{LS}(m))$  is equal to the completion time of job  $k$ . Assume that job  $k$  starts at time  $\tau$ , so that

$$C_{\max}(S_{LS}(m)) = \tau + p_k.$$

If  $\tau \leq \frac{m'-1}{m} C_{\max}(S_{np}^*(m'))$  then due to (1), we derive that

$$C_{\max}(S_{LS}(m)) = \tau + p_k \leq \frac{m' - 1}{m} C_{\max}(S_{np}^*(m')) + C_{\max}(S_{np}^*(m')),$$

so that (5) holds.

If, on the other hand,  $\tau > \frac{m'-1}{m} C_{\max}(S_{np}^*(m'))$ , then recall that in schedule  $S_{LS}(m)$  one machine completes at time  $C_{\max}(S_{LS}(m)) = \tau + p_k$ , and each of the remaining  $m-1$  machines is permanently busy in the time interval  $[0, \tau]$ , so that  $P \geq C_{\max}(S_{LS}(m)) + (m-1)\tau$ . If the theorem were not true, we would have  $C_{\max}(S_{LS}(m)) > \left(1 + \frac{m'-1}{m}\right) C_{\max}(S_{np}^*(m'))$ , so that

$$\begin{aligned} P &\geq C_{\max}(S_{LS}(m)) + (m-1)\tau \\ &> \left(1 + \frac{m' - 1}{m} + \frac{(m' - 1)(m - 1)}{m}\right) C_{\max}(S_{np}^*(m')) \\ &= m' C_{\max}(S_{np}^*(m')) \geq \frac{m' P}{m'}. \end{aligned}$$

The last inequality, which is due to (2) applied to schedule  $S_{np}^*(m')$ , leads to a contradiction that  $P > P$ .

To see that the bound (5) is tight for each  $m'$  consider an instance with  $m(m' - 1)$  jobs of unit duration each plus one extra job of length  $m$ . In this case, in schedule  $S_{LS}(m)$  the short jobs may be processed on each machine in the time interval  $[0, m' - 1]$ , while the long job starts on some machine at time  $m' - 1$ , so that  $C_{\max}(S_{LS}(m)) = m' - 1 + m$ . On the other hand, in schedule  $S_{np}^*(m')$  each machine is busy in the time interval  $[0, m]$ , processing either the long job or a block of  $m$  short jobs, so that  $C_{\max}(S_{np}^*(m')) = m$ . ■

Notice that for  $m' = m$  Theorem 2 is equivalent to the classical list scheduling result by Graham (1966). For  $m' = m + 1$  the theorem delivers the bound of 2 on the machine impact, so that Theorem 1 follows. Notice that the bound (4) is tight, even for optimal non-preemptive schedules. To see this, consider an instance with  $m + 1$  jobs of unit duration each. In this case,  $C_{\max}(S_{np}^*(m)) = 2$  and  $C_{\max}(S_{np}^*(m + 1)) = 1$ . Thus, the bound of 2 on the machine impact remains the same even if all jobs are identical. Also, the bound cannot be improved even if we apply any algorithm for finding a schedule on  $m$  parallel machines, e.g., a non-polynomial exact algorithm or a popular LPT heuristic.

### 3 Estimating Machine Impact: Total Flow Time

In this section, we consider problem  $Pm \parallel F$  with  $F = \sum C_j$ , i.e., the problem of minimizing the total flow time on  $m$  identical parallel machines  $M_1, M_2, \dots, M_m$ . We derive bounds on the impact that adding an extra machine may have on this objective function.

There are two points of differences between problem  $Pm \parallel \sum C_j$  and problem  $Pm \parallel C_{\max}$  from Section 2:

- as proved by McNaughton (1959), for problem  $Pm \parallel \sum C_j$  preemption, if allowed, does not reduce the value of the function, so that we may consider only non-preemptive schedules;
- as proved by Conway, Maxwell and Miller (1967), problem  $Pm \parallel \sum C_j$  is solvable in polynomial time.

Suppose that in a schedule  $S^*(m)$  that is optimal for problem  $Pm \parallel \sum C_j$  there are  $h_i$  jobs assigned to machine  $M_i$  and these jobs are processed in accordance with a sequence  $\pi_i(1), \dots, \pi_i(h_i)$ , where  $1 \leq i \leq m$ . Thus, on machine  $M_i$ , job  $\pi_i(u)$  completes at time  $C_{\pi_i(u)} = \sum_{j=1}^u p_{\pi_i(j)}$ , so that the objective function can be written as

$$\sum_{j=1}^n C_j(S^*(m)) = \sum_{i=1}^m \sum_{u=1}^{h_i} \sum_{j=1}^u p_{\pi_i(j)} = \sum_{i=1}^m \sum_{j=1}^u (h_i - u + 1) p_{\pi_i(j)}.$$

Further in this section, for problem  $Pm \parallel \sum C_j$  with  $m$  machines, we denote

$$F(S^*(m)) = \sum_{j=1}^n C_j(S^*(m)). \quad (6)$$

Recall that the algorithm for problem  $Pm \parallel \sum C_j$  presented in the book by Conway, Maxwell and Miller (1967) is essentially a version of the List Scheduling algorithm outlined in Section 2, in which the jobs are assigned to the first available machine in the SPT order, i.e., in non-decreasing order of their processing times. It is easy to verify that if  $n = km + r$ , where  $0 \leq r \leq m - 1$  then in the resulting optimal schedule  $S^*(m)$ , there will be  $k + 1$  jobs assigned to  $r$  machines and  $k$  jobs assigned to the remaining machines. It is convenient to view this schedule in the following way. Assume that the jobs are numbered in accordance with the LPT rule

$$p_1 \geq p_2 \geq \dots \geq p_n, \quad (7)$$

i.e., in non-increasing order of their processing times. Then, in schedule  $S^*(m)$  each of the first  $m$  jobs takes the last position on one of the machines, the next  $m$  jobs takes the second from last position on one of the machines, etc. This implies that job  $j$  contributes its processing time into the objective function exactly  $\left\lceil \frac{j}{m} \right\rceil$  times, and we can rewrite

$$F(S^*(m)) = \sum_{j=1}^n p_j \left\lceil \frac{j}{m} \right\rceil. \quad (8)$$

To derive the bounds on the machine impact

$$I(m) = \frac{F(S^*(m))}{F(S^*(m+1))} = \frac{\sum_{j=1}^n p_j \left\lceil \frac{j}{m} \right\rceil}{\sum_{j=1}^n p_j \left\lceil \frac{j}{m+1} \right\rceil}$$

for this model, we first consider a simpler version of the problem with unit processing times, and then extend the obtained results to the general case.

### 3.1 Unit Processing Times

In this subsection, we derive both upper and lower bounds on the machine impact  $I(m)$ , provided that  $p_j = 1$  for each job  $j \in N$ . The corresponding problem is traditionally denoted by  $Pm | p_j = 1 | \sum C_j$ . Notice that in Section 2 we have not looked at the problem  $Pm || C_{\max}$  with unit processing times, since, for that problem the bound of 2 on the machine impact remains tight even if the processing times are equal.

In our working, we use a close form formula for computing the total flow time  $F(S^*(m))$  for an optimal schedule on  $m$  machines.

**Lemma 1** *For problem  $Pm | p_j = 1 | \sum C_j$  the objective function can be computed by*

$$F(S^*(m)) = \sum_{j=1}^n \left\lceil \frac{j}{m} \right\rceil = \left( \left\lfloor \frac{n}{m} \right\rfloor + 1 \right) \left( n - \frac{m}{2} \left\lfloor \frac{n}{m} \right\rfloor \right). \quad (9)$$

The proof of the above statement can be found in Appendix A.

#### 3.1.1 Upper bound

Below we prove that for problem  $Pm | p_j = 1 | \sum C_j$  the bound on the machine impact

$$I(m) = \frac{\sum_{j=1}^n \left\lceil \frac{j}{m} \right\rceil}{\sum_{j=1}^n \left\lceil \frac{j}{m+1} \right\rceil} \leq \frac{m+1}{m}, \quad (10)$$

and this bound is tight. Incidentally, this is the same bound that is derived in Section 2 for problem  $Pm | pmtn | C_{\max}$ .

Define the sequence

$$A(j) = m \left\lceil \frac{j}{m} \right\rceil - (m+1) \left\lceil \frac{j}{m+1} \right\rceil, \quad j \in N. \quad (11)$$

To prove (10), we show that

$$\sum_{j=1}^n A(j) \leq 0. \quad (12)$$

We start with the statement that shows that the sequence  $A(j)$ ,  $j \in N$ , is periodic.

**Lemma 2** *The sequence  $A(j)$ ,  $1 \leq j \leq n$ , defined by (11) is periodic with a period of  $m(m+1)$ , so that for each  $j \leq n - m(m+1)$  the equality  $A(j + m(m+1)) = A(j)$  holds.*

**Proof:** Take an arbitrary  $j$ ,  $1 \leq j \leq n - m(m+1)$ . The following argument

$$\begin{aligned} A(j + m(m+1)) &= m \left\lfloor \frac{j + m(m+1)}{m} \right\rfloor - (m+1) \left\lfloor \frac{j + m(m+1)}{m+1} \right\rfloor \\ &= m \left\lfloor \frac{j}{m} \right\rfloor + m(m+1) - (m+1) \left\lfloor \frac{j}{m+1} \right\rfloor - m(m+1) \\ &= m \left\lfloor \frac{j}{m} \right\rfloor - (m+1) \left\lfloor \frac{j}{m+1} \right\rfloor = A(j) \end{aligned}$$

proves the lemma. ■

Consider the first  $m(m+1)$  elements of sequence  $A(j)$ ,  $1 \leq j \leq n$ , and call it a *block*. It follows from Lemma 2 that the sequence  $A(j)$ ,  $1 \leq j \leq n$ , is a collection of several blocks each containing  $m(m+1)$  elements, possibly followed by an incomplete block with less than  $m(m+1)$  elements. It appears that each full block can be further subdivided into  $m$  sequences of  $m+1$  elements each, which we call *patterns*. For a block  $A(j)$ ,  $1 \leq j \leq m(m+1)$ , its  $q$ -th pattern is given by the elements  $\{(q-1)(m+1)+1, (q-1)(m+1)+2, \dots, (q+1)(m+1)\}$ , where  $1 \leq q \leq m$ .

The following statement proves that the sum of the elements in each pattern is negative.

**Lemma 3** *For each pattern  $q$ ,  $1 \leq q \leq m$ , of the first block of elements  $A(j)$ ,  $1 \leq j \leq m(m+1)$ , of sequence (11), the relation*

$$\sum_{j=(q-1)(m+1)+1}^{q(m+1)} A(j) = -q < 0$$

*holds.*

**Proof:** It is easy to see that for each  $j \in \{(q-1)(m+1)+1, \dots, q(m+1)\}$  within the pattern, the value of  $\left\lfloor \frac{j}{m+1} \right\rfloor$  is equal to  $q$ . However, the value of the expression  $\left\lfloor \frac{j}{m} \right\rfloor$  does not remain constant for the entire range of  $j$  within the pattern and is given by

$$\left\lfloor \frac{j}{m} \right\rfloor = \begin{cases} q, & j \in \{(q-1)(m+1)+1, \dots, qm\} \\ q+1, & j \in \{qm+1, \dots, q(m+1)\} \end{cases}.$$

We rewrite

$$\begin{aligned} \sum_{j=(q-1)(m+1)+1}^{(q+1)m} A(j) &= \sum_{j=(q-1)(m+1)+1}^{qm} \left( m \left\lfloor \frac{j}{m} \right\rfloor - (m+1) \left\lfloor \frac{j}{m+1} \right\rfloor \right) \\ &\quad + \sum_{j=qm+1}^{q(m+1)} \left( m \left\lfloor \frac{j}{m} \right\rfloor - (m+1) \left\lfloor \frac{j}{m+1} \right\rfloor \right) \\ &= \sum_{j=(q-1)(m+1)+1}^{qm} (mq - (m+1)q) + \sum_{j=qm+1}^{q(m+1)} (m(q+1) - (m+1)q). \end{aligned}$$

In the right-hand side of the last expression, the first term reduces to  $(-q)$  taken  $m - q + 1$  times, and the second term reduces to  $m - q$  taken  $q$  times, so that

$$\sum_{j=(q-1)(m+1)+1}^{(q+1)m} A(j) = -(m - q + 1)q + q(m - q) = -q < 0,$$

which proves the lemma.  $\blacksquare$

In sequence (11), the sum of the elements of each complete block is equal to the sum of the elements of all its patterns and is, therefore, negative due to Lemma 3. Besides, in sequence (11) the last block may be incomplete, i.e., may contain less than  $m(m + 1)$  elements, or, equivalently, less than  $m$  complete patterns of  $m + 1$  elements each and one incomplete pattern, of less than  $m + 1$  elements. Due to Lemma 3, each complete pattern will make a negative contribution to the sum of the elements of sequence (11). As seen from the proof of Lemma 3, each incomplete pattern will have less positive elements than a complete pattern. Thus, the sum of the elements of an incomplete pattern is less than what it would have been, if that pattern were complete. We conclude that an incomplete pattern also makes a negative contribution to the overall sum of the elements of sequence (11). Thus, (12) holds, and the required upper bound (10) is proved.

To see that (10) is a tight bound, consider an instance of problem  $Pm | p_j = 1 | \sum C_j$  with  $n = Wm(m + 1)$  jobs of unit duration each, where  $W$  is a large positive number. For this instance, we use (9) to compute

$$\begin{aligned} F(S^*(m)) &= \frac{mW(m + 1)(Wm + W + 1)}{2}, \\ F(S^*(m + 1)) &= \frac{(m + 1)Wm(Wm + 1)}{2}. \end{aligned}$$

For the machine impact, we obtain that

$$I(m) = \frac{F(S^*(m))}{F(S^*(m + 1))} = \frac{mW(m + 1)(Wm + W + 1)}{(m + 1)Wm(Wm + 1)} = 1 + \frac{W}{Wm + 1} = 1 + \frac{1}{m + \frac{1}{W}},$$

and  $I(m)$  approaches  $\frac{m+1}{m}$  as  $W \rightarrow \infty$ .

### 3.1.2 Lower bound

We now derive a lower bound on the value of the machine impact  $I(m)$  for problem  $Pm | p_j = 1 | \sum C_j$ . Notice that this is the only problem from the range under consideration for which a non-trivial lower bound on  $I(m)$  exists; for the rest of the problems the lower bound is 1 and is tight. For example, for problem  $Pm | p_j = 1 | C_{\max}$  for  $n = 4$  and  $m = 2$  we have that  $I(m) = 1$ . For problems with arbitrary processing times, the machine impact is 1 in the presence of a very long job.

Below we prove that for problem  $Pm | p_j = 1 | \sum C_j$  the bound on the machine impact

$$I(m) \geq \begin{cases} 1 + \frac{1}{3m-1}, & \text{if } n \text{ is even} \\ 1 + \frac{1}{3m-3}, & \text{if } n \text{ is odd} \end{cases} \quad (13)$$

holds.

We only prove the top inequality in (13) and demonstrate tightness of both bounds. Define the sequence

$$B(j) = (3m - 1) \left\lfloor \frac{j}{m} \right\rfloor - 3m \left\lfloor \frac{j}{m+1} \right\rfloor, \quad j \in N. \quad (14)$$

To prove the top inequality in (13), we show that

$$\sum_{j=1}^n B(j) \geq 0. \quad (15)$$

Similar to Lemma 2, we can prove that the sequence (14) is ‘quasiperiodic’.

**Lemma 4** *For each  $j \leq n - m(m+1)$  the equality  $B(j + m(m+1)) = B(j) + 2m - 1$  holds.*

The sequence of  $B(j)$ ,  $1 \leq j \leq n$ , can be split into blocks of  $m(m+1)$  elements each (plus, possibly, an incomplete last block) defined exactly as the blocks of the sequence  $A(j)$ ,  $1 \leq j \leq n$ , in Section 3.1.1. Furthermore, each complete block can be split into  $m$  patterns of  $m+1$  elements each, similar to the patterns of the sequence  $A(j)$ ,  $1 \leq j \leq n$ , in Section 3.1.1.

Suppose that all patterns in the first block with the elements  $1, 2, \dots, m(m+1)$  are complete. The proof of Lemma 3 can be modified to show that the sum of a pattern  $q$ ,  $1 \leq q \leq m$ , in the first block is equal to  $2q(m-1)$ , and the sum of the elements of the first block is equal to  $\sum_{q=1}^m 2(m-1)q = m^3 - m$ . Besides, it can be easily verified in light of Lemma 4, that each element of a block other than the first is strictly positive.

We only need to address a situation that the first block contains an incomplete pattern. As in Section 3.1.1, for the first block the  $q$ -th pattern starts with at most  $m-q+1$  negative entries, each equal to  $-q$ , where  $q$ ,  $1 \leq q \leq m$ . The first pattern in the first block must be complete, since otherwise  $n < m$ . If the first block contains an incomplete pattern  $q = v$ ,  $2 \leq v \leq m$ , then the negative contribution from the  $v$ -th pattern is compensated by the positive contribution from the preceding complete patterns, since  $\sum_{q=1}^{v-1} 2q(m-1) \geq (m-v+1)v$ , where the equality holds for  $v = 2$ .

Thus, the inequality (15) holds and the required lower bound is proved. To establish the tightness, notice that the reasoning above implies that the sum of the elements  $B(j)$ ,  $1 \leq j \leq n$ , is the smallest if its last element  $n$  is the  $(m-1)$ -th element of the second pattern of the first block, i.e.,  $n = (m+1) + (m-1) = 2m$ . If  $n$  is even, then

$$\sum_{j=1}^n B(j) = 0,$$

and the impact factor  $I(m)$  is equal to  $1 + \frac{1}{3m-1}$ . The behavior of the machine impact and its bounds is illustrated by Figure 1.

If  $n$  is odd, we can apply a similar reasoning to demonstrate that  $I(m)$  is bounded from below by  $1 + \frac{1}{3m-3}$ , the equality holds for  $n = 2m - 1$ . Indeed, for this instance we have  $\lfloor \frac{n}{m} \rfloor = \lfloor \frac{n}{m+1} \rfloor = 1$  and by (9) we deduce that

$$\begin{aligned} F(S^*(m)) &= 2 \left( (2m-1) - \frac{m}{2} \right) = 3m - 2; \\ F(S^*(m+1)) &= 2 \left( (2m-1) - \frac{m+1}{2} \right) = 3m - 3, \end{aligned}$$

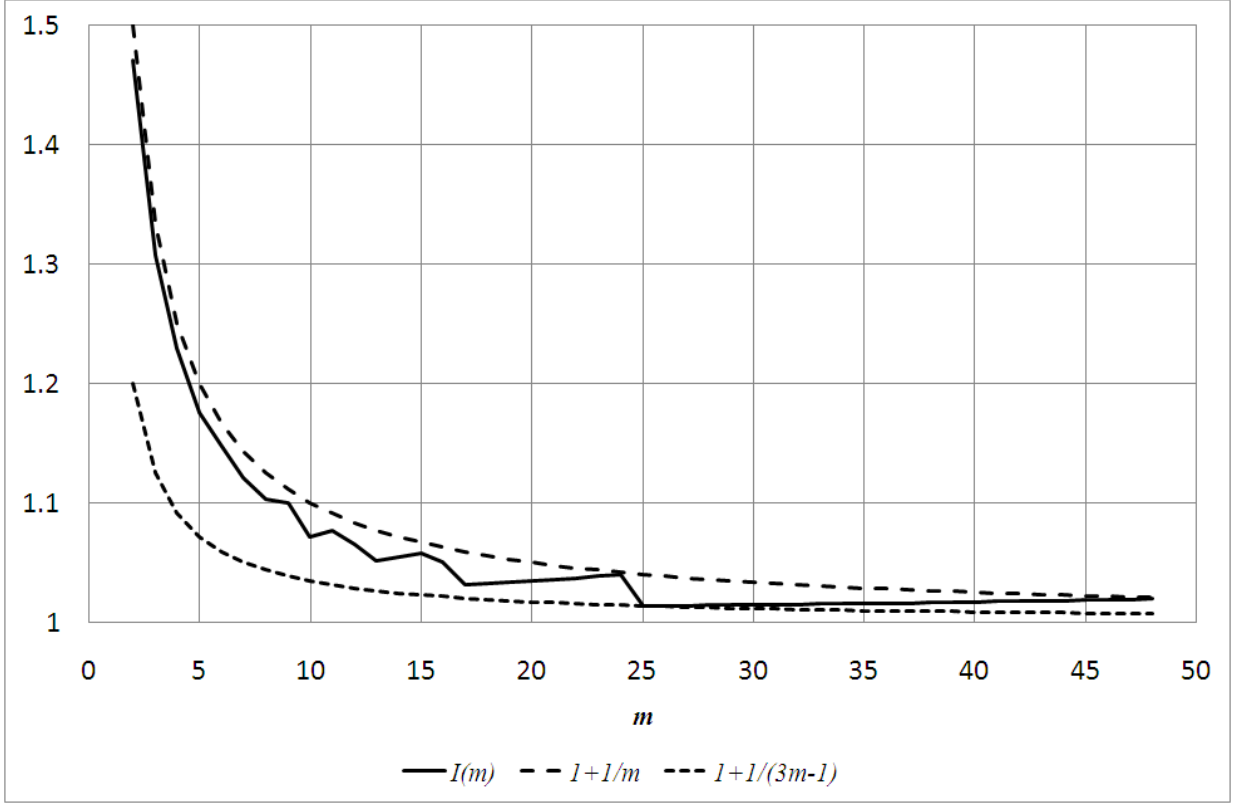


Figure 1: The graphs of  $I(m)$  and its lower and upper bounds for problem  $Pm | p_j = 1 | \sum C_j$  with  $n = 50$

as required.

We can use (9) to explain the behavior of function  $I(m)$ . Indeed, for the values of  $m$  larger than  $n/2$ , we derive that  $F(S^*(m)) = 2n - m$  and  $F(S^*(m + 1)) = 2n - m - 1$ , so that  $I(m)$  is monotone increasing. However, for  $m$  smaller than  $n/2$  the function  $I(m)$  is not monotone, which is due to ‘jumps’ in the values  $\lfloor \frac{n}{m} \rfloor$  and  $\lfloor \frac{n}{m+1} \rfloor$ , which contribute to  $F(S^*(m))$  and  $F(S^*(m + 1))$ , respectively. More precisely, it can be proved that when for some  $m = m'$  the inequalities  $I(m' - 1) > I(m')$  and  $I(m') < I(m' + 1)$  hold, i.e., when  $I(m')$  is a local minimum, then  $m' = \lceil n/r \rceil$  for some integer  $r$ , with the global minimum achieved for  $r = 2$ .

### 3.2 Arbitrary Processing Times

We now prove that  $\frac{m+1}{m}$  remains an upper bound on the machine impact for problem  $Pm || \sum C_j$  with arbitrary processing times. Recall that the jobs are numbered in the LPT order, in accordance with (7).

What we need to prove is the inequality

$$\sum_{j=1}^n p_j A_j \leq 0,$$

where  $A(j)$ ,  $1 \leq j \leq n$ , is the sequence defined by (11). Recall the latter sequence is periodic

by Lemma 2 and can be split into blocks, each of which is in turn split into patterns; see Section 3.1.1 for details.

We start with looking at a complete pattern of the first block.

**Lemma 5** *Let  $\{(q-1)m+1+1, \dots, q(m+1)\}$  be the  $q$ -th pattern of the first block of sequence (11), and that pattern is complete. Then*

$$\sum_{j=(q-1)(m+1)+1}^{q(m+1)} p_j A_j \leq -qp_{qm} < 0.$$

**Proof:** As follows from the proof of Lemma 3, each of the first  $m-q+1$  elements of the pattern of sequence (11) under consideration is equal to  $-q$ , while each of the remaining  $q$  elements is equal to  $m-q$ . Thus,

$$\sum_{j=(q-1)(m+1)+1}^{q(m+1)} p_j A_j = -q \sum_{j=(q-1)(m+1)+1}^{qm} p_j + (m-q) \sum_{j=qm+1}^{q(m+1)} p_j.$$

Due to the LPT numbering of the jobs, we deduce

$$\begin{aligned} \sum_{j=(q-1)(m+1)+1}^{q(m+1)} p_j A_j &\leq -q(m-q+1) \min\{p_j \mid (q-1)(m+1)+1 \leq j \leq qm\} + \\ &\quad + (m-q)q \max\{p_j \mid qm+1 \leq j \leq (q+1)m\} \\ &= -q(m-q+1)p_{qm} + (m-q)qp_{qm+1} \\ &\leq -q(m-q+1)p_{qm} + (m-q)qp_{qm} = -qp_{qm} \leq 0, \end{aligned}$$

as required. ■

Due to the periodic nature of the sequence (11) guaranteed by Lemma 2, we can extend Lemma 5 to any complete pattern of any block. If there is an incomplete pattern that finishes the sequence (11), then the sum product  $p_j A_j$  for  $j$  that belong to that pattern can easily be proved negative. As proved in Section 3.1.1, such a pattern of the sequence (11) either contains only negative values or all negative values and less positive values than it would if it were complete.

The main result of this section can be stated as follows.

**Theorem 3** *For problem  $Pm \parallel \sum C_j$ , the following bound holds*

$$I(m) = \frac{\sum C_j(S_{np}^*(m))}{\sum C_j(S_{np}^*(m+1))} \leq \frac{m+1}{m}$$

and this bound is tight.

In line with Theorem 2 derived for the makespan, we can extend the results of this section by deriving a bound on the ratio  $\frac{F(S^*(m))}{F(S^*(m'))}$  for any positive  $m' > m$ . Multiple application of

(10) gives us

$$\begin{aligned} F(S^*(m)) &\leq \frac{m+1}{m} F(S^*(m+1)) \leq \frac{m+1}{m} \frac{m+2}{m+1} F(S^*(m+2)) \\ &\leq \frac{(m+1)(m+2)\cdots m'}{m(m+1)\cdots(m'-1)} F(S^*(m')), \end{aligned}$$

so that

$$\frac{F(S^*(m))}{F(S^*(m'))} \leq \frac{m'}{m}.$$

To see that the latter bound is a tight, consider an instance of problem  $Pm|p_j = 1|\sum C_j$  with  $n = Wm(m+y)$  jobs of unit duration each, where  $y \geq 1$  and  $W$  is a large positive number. We compare  $F(S^*(m))$  and  $F(S^*(m'))$  for  $m' = m+y$ . For this instance, we use (9) to compute

$$\begin{aligned} F(S^*(m)) &= \frac{mW(m+y)(Wm+Wy+1)}{2}, \\ F(S^*(m+y)) &= \frac{(m+y)Wm(Wm+1)}{2}. \end{aligned}$$

Thus, as  $W \rightarrow \infty$  the ratio

$$\frac{F(S^*(m))}{F(S^*(m'))} = \frac{(Wm+Wy+1)}{(Wm+1)}$$

approaches  $\frac{m+y}{m} = \frac{m'}{m}$ .

## 4 Cost-Effective Choice of The Number of Machines: Makespan

In the previous sections, we have derived bounds on the machine impact that show how a scheduling performance measure  $F$  (either the makespan or the total flow time) is affected by the arrival of an extra machine. In reality, adding a machine cannot be seen free. From now on, we address the problem of making an optimal, cost-effective choice of the number of machines to minimize the function that captures a trade-off between the value of  $F$  and the cost of using machines.

Formally, assume that using a machine incurs a cost  $K$ , and we are interested in minimizing the total cost function

$$\Phi(m) = \alpha F(S^*(m)) + \beta Km, \quad (16)$$

where  $S^*(m)$  denotes an optimal schedule with  $m$  parallel machines to minimize an objective  $F \in \{C_{\max}, \sum C_j\}$ , while  $\alpha$  and  $\beta$  represent positive weights associated with the contribution of the scheduling objective and the machine cost, respectively.

In this section, we handle the problem with  $F$  being the makespan, for the preemptive and non-preemptive cases.

The problems we consider are related to those introduced by Imreh and Noga (1999), who study the non-preemptive problem of minimizing the function (16) for  $F = C_{\max}$  and

$\alpha = \beta = K = 1$  in the online settings. In their model, the decision-maker has no machines in the beginning and when a job is revealed may buy as many machines as needed. The jobs are either released according to a list (The List Model) or arrive over time (The Time Model). The focus is on deriving lower and upper bounds on the competitive ratios of the online algorithms. The best known bounds for the online non-preemptive problem are due Dósa and Tan (2010). The preemptive version of the List Model is studied by Jiang and He (2005). For the List Model, the semi-online scenarios are considered by He and Cai (2002) (no preemption allowed) and by Jiang and He (2006) (both preemptive and non-preemptive versions). In the semi-online scenarios, the decision-maker is either aware of the longest processing time of arriving jobs or of the total processing time  $P$ , and this leads to better bounds on the competitive ratios. A more general cost function for using the machines is analyzed by Imreh (2009). In the cited papers, analysis is often based on lower bounds on the value of  $\Phi(m)$  for the off-line variant of the problem, initially obtained by Imreh and Noga (1999), which holds for both the preemptive and non-preemptive versions. This is probably the closest link between the studies of the online models and the topic of this section.

For the preemptive case, we present a linear time algorithm for finding the number of machines that minimizes function (16). For the non-preemptive case, we analyze a worst-case behavior of an approximation algorithm that accepts the number of machines that is optimal for the preemptive counterpart.

#### 4.1 Preemption Allowed

We start with the preemptive version of the problem of minimizing the total cost. We denote the total cost function by  $\Phi_p(m)$ , provided that the scheduling objective is the makespan; the subscript “ $p$ ” is used to indicate preemption. Thus,

$$\Phi_p(m) = \alpha C_{\max}(S_p^*(m)) + \beta Km,$$

where  $S_p^*(m)$  is an optimal preemptive schedule on  $m$  machines. Let  $m_p^*$  denote the optimal number of machines, i.e.,  $\Phi_p(m_p^*) \leq \Phi_p(m)$  for all values of  $m \geq 1$ . Recall that for the optimal makespan for problem  $Pm|pmtn|C_{\max}$  the lower bounds (1) and (2) hold.

Define  $m_1$  as the smallest number of machines for which the duration of the longest job is either larger than or equal to the average machine load, i.e.,

$$m_1 = \left\lceil \frac{P}{p_{\max}} \right\rceil. \quad (17)$$

Thus, we can consider function  $\Phi_p(m)$  as

$$\Phi_p(m) = \begin{cases} \Gamma_1(m), & \text{for } m \geq m_1 \\ \Gamma_2(m), & \text{for } 1 \leq m < m_1, \end{cases}$$

where

$$\begin{aligned} \Gamma_1(m) & : = \alpha p_{\max} + \beta Km; \\ \Gamma_2(m) & : = \alpha \frac{P}{m} + \beta Km. \end{aligned} \quad (18)$$

Obviously,  $m_1$  minimizes  $\Gamma_1(m)$ , since adding another machine incurs extra cost without changing the makespan  $p_{\max}$ . Let  $m_2$  be such that

$$\Gamma_2(m_2) = \min \{ \Gamma_2(m) \mid 1 \leq m < m_1 \}.$$

If we have  $\Gamma_1(m_1) \leq \Gamma_2(m_2)$ , then  $m_p^* = m_1$ , otherwise,  $m_p^* = m_2$ .

Notice that function  $\Gamma_2(m)$  is exactly of the same shape as the total cost function of the EOQ model of inventory control, this topic being a part of any standard OR curriculum; see, e.g., Winston (1994). Function  $\Gamma_2(m)$  is convex and its global minimum is achieved for  $m = \mu$  found by the formula, similar to the famous “square root” EOQ formula:

$$\mu = \sqrt{\frac{\alpha P}{\beta K}}. \quad (19)$$

In what follows, we assume that the coefficients  $\alpha$ ,  $\beta$  and  $K$  are such that  $\mu < n$ ; otherwise function  $\Gamma_2(m)$  reaches its minimum for  $m = n$ . Notice that  $\alpha \frac{P}{\mu} = \beta K \mu$ , so that the smallest value of  $\Gamma_2(m)$  is equal to  $2\mu$ . This value, which becomes  $2\sqrt{P}$  for  $\alpha = \beta = K = 1$ , complies with the bound derived by Imreh and Noga (1999) and can be used as a lower bound on the total cost, as it is done in the online studies cited in Section 4.

Due to the convexity of  $\Gamma_2(m)$ , we observe that

$$\begin{aligned} \alpha \frac{P}{m} &< \beta K m, & m < \mu; \\ \alpha \frac{P}{m} &> \beta K m, & m > \mu. \end{aligned} \quad (20)$$

If  $\mu \geq m_1$ , then function  $\Gamma_2(m)$  decreases for  $m \in \{1, 2, \dots, m_1 - 1\}$ , so that  $\Gamma_2(m_1 - 1)$  might be smaller than  $\Gamma_1(m_1)$ . For  $\mu < m_1$ , if  $\mu$  happens to be integer, we take  $m_2 = \mu$ . Otherwise, due to the convexity of function  $\Gamma_2(m)$ , its integer minimum is delivered either by  $m' = \lfloor \mu \rfloor$  or  $m'' = \lceil \mu \rceil$ , depending on which of these two values returns a lower value of function  $\Gamma_2(m)$ .

Figure 2 illustrates how function  $\Phi_p(m)$  is formed. In the taken instance,  $\alpha = \beta = 1$ ,  $K = 2.8$ ,  $P = 108$  and  $p_{\max} = 25$ . In this case the graphs of  $\Gamma_1(m)$  and  $\Gamma_2(m)$  intersect at  $m = 4.32$ , while  $\mu = 6.21059$ . In this case,  $m_2 < m_1 < m' < m''$ . For other instances other relative orders of these four values are possible. Algorithm 1 below describes how to find the one that delivers the minimum of function  $\Phi_p(m)$ .

### Algorithm 1

- Step 1.** Determine the values of  $P$  and  $p_{\max}$ . Compute  $m_1$  by (17) and define  $m_p^* := m_1$ .
- Step 2.** Compute  $\mu$  by (19). If  $\lceil \mu \rceil < m_1$ , go to Step 3; otherwise define  $m_2 = m_1 - 1$ , if  $\Gamma_2(m_2) < \Gamma_1(m_1)$  redefine  $m_p^* := m_2$ . Go to Step 4.
- Step 3.** Compute  $m' := \lfloor \mu \rfloor$  and  $m'' := \lceil \mu \rceil$ . Define  $m_p^* := m'$  and, if  $\Gamma_2(m'') < \Gamma_2(m')$ , redefine  $m_p^* := m''$ .
- Step 4.** Output  $m_p^*$  and  $\Phi_p(m_p^*)$ . Stop.

The running time of the Algorithm 1 is  $O(n)$ , provided the square root operation takes constant time. The algorithm outputs the optimal number of machines  $m_p^*$  and the optimal total cost  $\Phi_p(m_p^*)$ . The corresponding optimal schedule  $S_p^*(m_p^*)$  can be found by McNaughton’s algorithm in  $O(n)$  time. For the instance illustrated in Figure 2 Algorithm 1 outputs  $m_p^* = m_2 = 4$  found in Step 2.

Thus, the following statement holds.

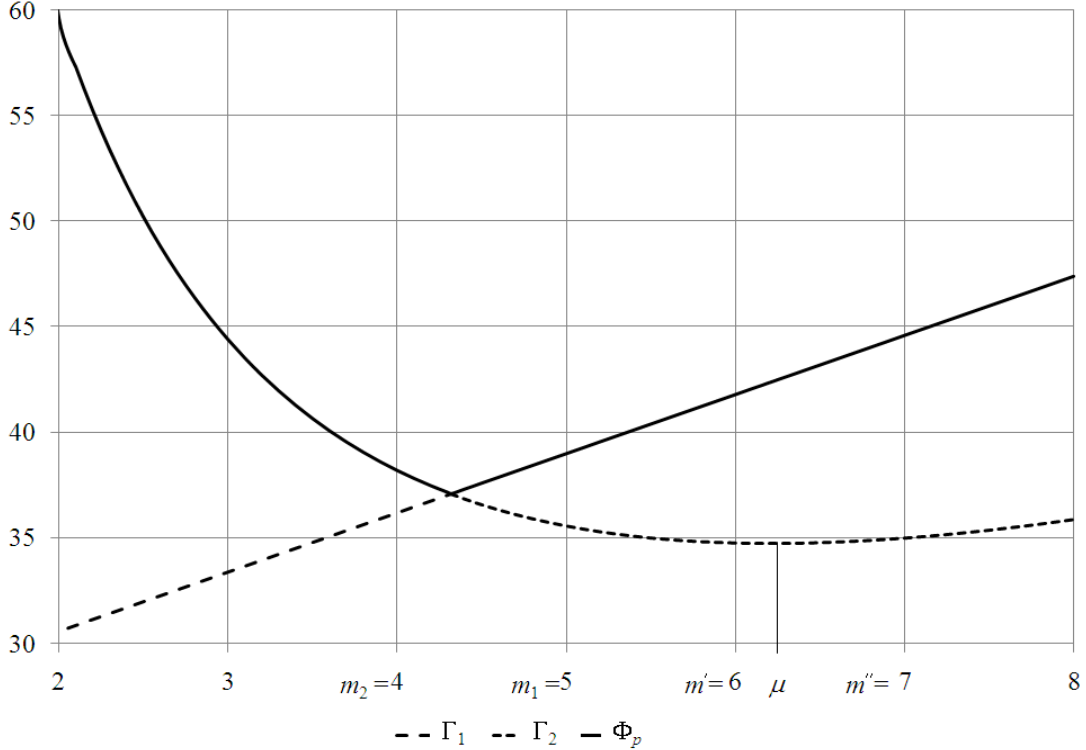


Figure 2: An example of graphs of the functions  $\Gamma_1(m)$ ,  $\Gamma_2(m)$  and  $\Phi_p(m)$

**Theorem 4** *Algorithm 1 finds the optimal number of machines  $m_p^*$ , the optimal cost  $\Phi_p(m_p^*)$  and the corresponding preemptive optimal schedule  $S_p^*(m_p^*)$  in  $O(n)$  time.*

## 4.2 No Preemption Allowed

Now we pass to the non-preemptive version of the problem. We only may look for an approximation algorithm, since problem  $Pm \parallel C_{\max}$  is NP-hard. We denote the total cost function by  $\Phi_{np}(m)$ ; the subscript “np” is used to indicate that no preemption is allowed. Thus,

$$\Phi_{np}(m) = \alpha C_{\max}(S_{np}^*(m)) + \beta K m,$$

where  $S_{np}^*(m)$  is an optimal non-preemptive schedule on  $m$  machines. Let  $m_{np}^*$  denote the optimal number of machines, i.e.,  $\Phi_{np}(m_{np}^*) \leq \Phi_{np}(m)$  for all values of  $m \geq 1$ .

As a part of our algorithm we need to be able to describe a procedure for finding a non-preemptive schedule with a given number of machines and compare the found makespan with the makespan of the optimal preemptive schedule with the same number of machines. This brings us to a discussion of the power of preemption in the context of scheduling of identical parallel machines.

Recall that for an instance of a scheduling problem to minimize the makespan  $C_{\max}$  on  $m$  identical parallel machines the *power of preemption* is defined as the maximum ratio  $C_{\max}(S_{np}^*(m))/C_{\max}(S_p^*(m))$  across all instances of the problem at hand. As independently

proved by Braun and Schmidt (2003) and Lee and Strusevich (2005),

$$\frac{C_{\max}(S_{np}^*(m))}{C_{\max}(S_p^*(m))} \leq 2 - \frac{2}{m+1}.$$

In particular, Braun and Schmidt (2003) show that the above bound holds if an optimal non-preemptive schedule  $S_{np}^*(m)$  is replaced by a schedule found by an LPT list scheduling algorithm which takes  $O(n \log n + nm)$  time. Lee and Strusevich (2005) describe a large class of non-preemptive schedules for which the above bounds holds.

Below we present an algorithm that in  $O(nm)$  time outputs a non-preemptive schedule that allows us to give better estimates of the power of preemption.

## Algorithm 2

**Step 1.** Compute  $P$ , and find  $p_m$ , the  $m$ -th largest value among  $p_j$ ,  $j \in N$ . Identify  $m-1$  values of the processing times for which  $p_j \geq p_m$  and renumber the jobs in such a way that

$$p_1 \geq p_2 \geq \dots \geq p_m,$$

while the remaining jobs are taken in an arbitrary order. Determine the index  $g$ ,  $0 \leq g \leq m-1$ , as the smallest index such that

$$p_{g+1} < \frac{P - \sum_{j=1}^g p_j}{m-g}$$

**Step 2.** For  $1 \leq j \leq g$ , taking the jobs in the order of the obtained numbering, assign job  $j$  to machine  $M_j$ . Complete the obtained partial schedule by assigning the remaining jobs to machines  $M_{g+1}, \dots, M_m$  by Algorithm LS. Call the resulting schedule  $S_{np}(m)$  and stop.

Finding  $p_m$  in Step 1 requires  $O(n)$  time by the median technique. Renumbering the jobs needs  $O(m \log m + n)$  time, determining  $g$  takes  $O(m)$  time and Step 2 needs  $O(nm)$  time. Thus, the overall running time of the algorithm is  $O(m(n + \log m)) = O(nm)$ , due to  $n \geq m$ .

**Theorem 5** *For the problem of scheduling  $n$  jobs on  $m$  parallel identical machines Algorithm 2 finds a non-preemptive schedule  $S_{np}(m)$  for which either  $C_{\max}(S_{np}(m)) = p_1$  or*

$$C_{\max}(S_{np}(m)) \leq \left(2 - \frac{2}{m-g+1}\right) \frac{P - \sum_{j=1}^g p_j}{m-g}, \quad (21)$$

*the latter bound being tight.*

The proof of Theorem 5 is given in Appendix B. For our purposes, we need the following statement that follows from Theorem 5.

**Corollary 1** *For the problem of scheduling  $n$  jobs on  $m$  parallel identical machines Algorithm 2 finds a non-preemptive schedule  $S_{np}(m)$  such that either*

$$C_{\max}(S_{np}(m)) \leq \left(2 - \frac{2}{m+1}\right) \frac{P}{m}, \quad (22)$$

if  $p_{\max} = p_1 \leq \frac{P}{m}$  and  $g = 0$ , or

$$C_{\max}(S_{np}(m)) \leq \left(2 - \frac{2}{m-g+1}\right) p_g, \quad (23)$$

if  $p_{\max} = p_1 > \frac{P}{m}$  and  $g \geq 1$ .

The proof of the corollary is moved to Appendix C. Since  $P/m$  and  $p_g$  are lower bounds on the value of  $C_{\max}(S_p(m))$ , the inequalities (22) and (23) give refined tight upper bounds on the power of preemption for identical machines.

Now we come back to designing an approximation algorithm for minimizing the total cost function  $\Phi_{np}(m)$ . Our algorithm will take the candidate number of machines found by Algorithm 1 and then use Algorithm 2 to find a non-preemptive schedule with the chosen number of machines.

### Algorithm 3

**Step 1.** Run Algorithm 1 to find the number of machines  $m_p^*$  that is optimal for the preemptive version of our problem. Define  $m^H := m_p^*$ .

**Step 2.** Run Algorithm 2 to find a non-preemptive schedule  $S_{np}(m^H)$ . Output  $m^H$ ,  $C_{\max}(S_{np}(m^H))$  and  $\Phi_{np}(m^H)$ .

The running time of this algorithm is  $O(nm)$ . It appears that the worst-case performance of Algorithm 3 depends in which step of Algorithm 1 the value  $m_p^*$  is found; in other words, that depends on the sign of the difference  $m_1 - \lceil \mu \rceil$ . This is why our analysis of Algorithm 3 is done in two separate statements.

**Theorem 6** For  $m^H = m_p^* \in \{m_1, m_2\}$  found in Step 2 of Algorithm 1 the bound

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq 2 - \frac{2}{m_1} \quad (24)$$

holds, and this bound is tight.

**Proof:** Since the makespan of the best non-preemptive schedule with  $m$  machines is no smaller than the makespan for the best preemptive schedule with the same number of machines, it follows that  $\Phi_{np}(m_{np}^*) \geq \Phi_p(m_{np}^*) \geq \Phi_p(m_p^*)$ .

If  $m_p^* = m_2 = m_1 - 1$  (see Figure 2 for an illustration), then  $C_{\max}(S_p^*(m_2)) = P/m_2 > p_{\max}$  and we deduce

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq \frac{\Phi_{np}(m_2)}{\Phi_p(m_2)} = \frac{\Phi_{np}(m_2)}{\Gamma_2(m_2)} = \frac{\alpha C_{\max}(S_{np}^*(m_2)) + \beta K m_2}{\alpha \frac{P}{m_2} + \beta K m_2}.$$

Algorithm 2 applied to  $m = m_2$  will find  $g = 0$ , so that (22) holds for schedule  $S_{np}(m_2)$ . Thus,

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq \frac{\alpha C_{\max}(S_{np}(m_2)) + \beta K m_2}{\alpha \frac{P}{m_2} + \beta K m_2} \leq \frac{\alpha \left(2 - \frac{2}{m_2+1}\right) \frac{P}{m_2} + \beta K m_2}{\alpha \frac{P}{m_2} + \beta K m_2}.$$

The last fraction decreases in  $\beta Km_2$ , so that we can replace it by zero to achieve

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq \frac{\alpha \left(2 - \frac{2}{m_2+1}\right) \frac{P}{m_2}}{\alpha \frac{P}{m_2}} = 2 - \frac{2}{m_2+1} = 2 - \frac{2}{m_1},$$

as required.

If  $m_p^* = m_1$  then  $C_{\max}(S_p^*(m_1)) = p_{\max} \geq P/m_1$  and we deduce

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq \frac{\Phi_{np}(m_1)}{\Phi_p(m_1)} = \frac{\Phi_{np}(m_1)}{\Gamma_1(m_1)} = \frac{\alpha C_{\max}(S_{np}^*(m_1)) + \beta Km_1}{\alpha p_{\max} + \beta Km_1}.$$

Algorithm 2 applied to  $m = m_1$  will find that  $g \geq 1$ , so that (23) holds for schedule  $S_{np}(m_1)$ . Therefore,

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq \frac{\alpha p_g \left(2 - \frac{2}{m_1-g+1}\right) + \beta Km_1}{\alpha p_{\max} + \beta Km_1}.$$

In the worst case,  $g = 1$  and  $p_g = p_{\max}$ . Replacing  $\beta Km_2$  by zero as above, we obtain

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq \frac{\alpha p_{\max} \left(2 - \frac{2}{m_1}\right)}{\alpha p_{\max}} = 2 - \frac{2}{m_1}.$$

To see that the bound (24) is tight, consider an instance with  $m+1$  jobs, that contains one job with the processing time  $m$  and  $m$  jobs with the processing time  $m-1$  each, where  $m$  is a positive integer. In the objective function, set  $\alpha = \beta = 1$ .

The cost of using a machine is  $K < 1$ . It follows that  $m_1 = m$  and  $\mu = \sqrt{m^2/K} > m$ , so that  $m_2 = m_1 - 1 = m - 1$ . Since  $\Gamma_1(m_1) = m + Km$  and  $\Gamma_2(m_2) = \frac{m^2}{m-1} + K(m-1)$ , we deduce that  $m_p^* = m_1 = m$  due to a small value of  $K$ . Taking  $m$  as  $m^H$  we find the best non-preemptive schedule on  $m$  machines in which each machine processes one job, except one machine that processes two jobs of duration  $m-1$  each, i.e.,  $C_{\max}(S_{np}^*(m)) = 2m-2$  and  $\Phi_{np}(m) = 2m-2 + Km$ . On the other hand, the total cost function reaches its minimum for  $m_{np}^* = m+1$ , so that in schedule  $S_{np}^*(m+1)$  each machine processes exactly one job. Thus,  $C_{\max}(S_{np}^*(m+1)) = m$  and  $\Phi_{np}(m+1) = m + K(m+1)$ . Thus,

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} = \frac{\Phi_{np}(m)}{\Phi_{np}(m+1)} = \frac{2m-2 + Km}{m + K(m+1)}.$$

As  $K$  approaches zero, the ratio goes to  $2 - \frac{2}{m}$ . This proves the theorem.  $\blacksquare$

The next statement establishes an improved performance of Algorithm 3, because for  $m_1 > \lceil \mu \rceil$  the machine cost  $\beta Km$  makes a larger contribution to the total cost function.

**Theorem 7** For  $m^H = m_p^* \in \{m', m''\}$  found in Step 3 by Algorithm 1 the bound

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq \frac{3}{2} - O\left(\frac{1}{m_p^*}\right) \quad (25)$$

holds, and this bound is tight.

**Proof:** Since the function  $\Gamma_2(m)$  of the form (18) is convex, we deduce that  $\Gamma_2(m) > \Gamma_2(m')$  for  $m < m'$  and  $\Gamma(m) > \Gamma(m'')$  for  $m > m''$ . Since a non-preemptive schedule has a makespan that is not smaller than that for a preemptive one for the same input data, we have that

- (i) for  $m_{np}^* \in \{m', m''\}$  the value  $m^H$  is optimal;
- (ii) for  $m_{np}^* < m'$ , the inequalities  $\Phi_{np}(m_{np}^*) \geq \Phi_p(m_{np}^*) = \Gamma_2(m_{np}^*) > \Gamma_2(m')$ ;
- (iii) for  $m_{np}^* > m''$ , the inequalities  $\Phi_{np}(m_{np}^*) \geq \Phi_p(m_{np}^*) = \Gamma_2(m_{np}^*) > \Gamma_2(m'')$ .

We know that  $\Phi_{np}(m^H) = \min \{\Phi_{np}(m'), \Phi_{np}(m'')\}$ , and if  $\mu$  is integer then  $m' = m''$ , while otherwise  $m' < \mu < m''$ , where  $m' = m'' - 1$ .

If  $\Phi_{np}(m_{np}^*) > \Gamma_2(m'')$  we deduce from (22) with  $m = m''$  that

$$\begin{aligned} \frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} &\leq \frac{\Phi_{np}(m'')}{\Gamma_2(m'')} = \frac{\alpha C_{\max}(S_{np}^*(m'')) + \beta K m''}{\alpha \frac{P}{m''} + \beta K m''} \\ &\leq \frac{\alpha C_{\max}(S_p^*(m'')) \left(2 - \frac{2}{m''+1}\right) + \beta K m''}{\alpha \frac{P}{m''} + \beta K m''}. \end{aligned}$$

Notice that due to the choice of  $m_2$  the equality  $C_{\max}(S_p^*(m_2)) = \frac{P}{m_2}$  holds, so that

$$\frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} \leq \frac{\alpha \frac{P}{m''} \left(2 - \frac{2}{m''+1}\right) + \beta K m''}{\alpha \frac{P}{m''} + \beta K m''}.$$

The last derived expression is decreasing in  $\beta K m''$ . Since  $m'' \geq \mu$ , it follows from (20) with  $m = m''$  that

$$\begin{aligned} \frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} &\leq \frac{\alpha \frac{P}{m''} \left(2 - \frac{2}{m''+1}\right) + \alpha \frac{P}{m''}}{2\alpha \frac{P}{m''}} \\ &= \frac{3 - \frac{2}{m''+1}}{2} = \frac{3}{2} - \frac{1}{m''+1} = \frac{3}{2} - O\left(\frac{1}{m_p^*}\right). \end{aligned}$$

Now, assume that  $\Phi_{np}(m_{np}^*) > \Gamma(m')$  for  $m' = m'' - 1$ . We deduce

$$\begin{aligned} \frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} &\leq \frac{\Phi_{np}(m')}{\Gamma(m')} = \frac{\alpha \left(2 - \frac{2}{m'+1}\right) \frac{P}{m'} + \beta K m'}{\alpha \frac{P}{m'} + \beta K m'} \\ &= \frac{\alpha \left(2 - \frac{2}{m'+1}\right) \frac{P}{m'} + \beta K m'}{\alpha \frac{P}{m'} + \beta K m'}. \end{aligned}$$

Applying (20) with  $m = m'' = m' + 1$ , we get

$$\alpha \frac{P}{m'+1} < \beta K (m' + 1),$$

which implies

$$\alpha \frac{m' P}{(m' + 1)^2} < \beta K m'.$$

Using this lower bound on the cost component, we obtain

$$\begin{aligned} \frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} &\leq \frac{\alpha \left(2 - \frac{2}{m'+1}\right) \frac{P}{m'} + \alpha \frac{m'P}{(m'+1)^2}}{\alpha \frac{P}{m'} + \alpha \frac{m'P}{(m'+1)^2}} \\ &= \frac{3}{2} - \frac{m' + \frac{3}{2}}{2(m')^2 + 2m' + 1} = \frac{3}{2} - O\left(\frac{1}{m_p^*}\right). \end{aligned}$$

To see that the bound (25) is tight, consider an instance with  $m + 1$  jobs, each with the processing time  $m$ , where  $m$  is a positive integer. In the objective function, set  $\alpha = \beta = 1$ . The cost of using a machine is  $K = \frac{m+1}{m}$ . It follows that  $m_1 = m + 1$  and  $m_2 = \mu = m' = m'' = m$ . Since  $\Gamma_1(m_1) = m + \frac{(m+1)^2}{m} = 2m + 2 + \frac{1}{m}$  and  $\Gamma_2(m_2) = 2(m + 1)$ , we deduce that  $m_p^* = m_2 = m$ . Taking  $m$  as  $m^H$  we find the best non-preemptive schedule on  $m$  machines in which each machine processes one job, except one machine that processes two jobs, i.e.,  $C_{\max}(S_{np}^*(m)) = 2m$  and  $\Phi_{np}(m) = 2m + (m + 1) = 3m + 1$ . On the other hand, the total cost function reaches its minimum for  $m_{np}^* = m + 1$ , so that  $C_{\max}(S_{np}^*(m + 1)) = m$  and  $\Phi_{np}(m + 1) = m + \frac{(m+1)^2}{m}$ . Thus,

$$\begin{aligned} \frac{\Phi_{np}(m^H)}{\Phi_{np}(m_{np}^*)} &= \frac{\Phi_{np}(m)}{\Phi_{np}(m + 1)} = \frac{3m + 1}{m + \frac{(m+1)^2}{m}} \\ &= \frac{3m^2 + m}{2m^2 + 2m + 1} = \frac{3}{2} - \frac{m + \frac{3}{2}}{2m^2 + 2m + 1} = \frac{3}{2} - O\left(\frac{1}{m}\right). \end{aligned}$$

This proves the theorem. ■

## 5 Cost-Effective Choice of The Number of Machines: Total Flow Time

In this section, we consider the problem of finding the optimal number of machines that minimizes the total cost function (16), provided that the scheduling objective  $F$  is the total cost.

Recall that for a given number of machines  $m$ , the optimal total flow time is defined by formula (8), provided that the jobs are numbered in accordance with the LPT rule (7). Define two sequences:

$$\begin{aligned} F(m) &= \sum_{j=1}^n p_j \left\lceil \frac{j}{m} \right\rceil, \quad m = 1, 2, \dots \\ T(m) &= Km, \quad m = 1, 2, \dots, \end{aligned}$$

where  $F(m)$  is the total flow time of processing the jobs on  $m$  machines and  $T(m)$  defines the total cost of using  $m$  machines. Thus, for the problem under consideration, the objective function can be written as

$$\Phi(m) = \alpha F(m) + \beta T(m).$$

Finding the value of  $m^*$  that minimizes function  $\Phi(m)$  can easily be done by direct enumeration: try all values of  $m$  from 1 to  $n$ , compute the components of the function and

select the best value of  $m$ . Such an approach requires  $O(n \log n)$  time for finding an LPT numbering of the jobs, and then  $O(n)$  time for computing function  $\Phi$  for each trial value of  $m$ ,  $1 \leq m \leq n$ . Thus, such a brute-force algorithm takes  $O(n^2)$  time.

Below we present an  $O(n \log n)$  algorithm for finding  $m^*$  that uses only  $\lceil \log_2 n \rceil$  trial values of  $m$ , due to our observation that the sequence  $\Phi(m)$ ,  $1 \leq m \leq n$ , is actually  $V$ -shaped with respect to  $m$ ,  $1 \leq m \leq n$ . Recall that a sequence  $A(m)$  is called  $V$ -shaped if there exists an  $m_0$ ,  $1 \leq m_0 \leq n$ , such that

$$A(1) \geq \dots \geq A(m_0 - 1) \geq A(m_0) \leq A(m_0 + 1) \leq \dots \leq A(n).$$

First, we show that sequence  $F(m)$ ,  $m = 1, 2, \dots, n$ , is convex. Recall that a sequence  $A(m)$ ,  $1 \leq m \leq n$ , is called *convex* if

$$A(m) \leq \frac{1}{2} (A(m-1) + A(m+1)), \quad 2 \leq m \leq n-1. \quad (26)$$

Our reasoning is based on the following property proved in our earlier paper Rustogi and Strusevich (2011a).

**Lemma 6** *For any non-decreasing function  $f$ , the sequence*

$$B(m) = \sum_{j=1}^q f \left( \left\lceil \frac{j}{m} \right\rceil \right), \quad 1 \leq m \leq n,$$

*is convex for each  $q$ ,  $1 \leq q \leq n$ .*

Below we need Lemma 6 in a weaker form, with  $f \equiv 1$ . Its application in the full form to another scheduling problem can be found in Rustogi and Strusevich (2011b).

**Lemma 7** *Let  $S^*(m)$  be a schedule that minimizes the total flow time on  $m$  identical parallel machines, so that  $F(m) = \sum_{j \in N} C_j(S^*(m))$ . Then the sequence of values  $F(m)$ ,  $1 \leq m \leq n$ , is convex.*

**Proof:** Due to (26), we need to prove that

$$F(m) \leq \frac{1}{2} (F(m-1) + F(m+1)), \quad 2 \leq m \leq n-1.$$

For a given  $j \in N$ , define

$$D_j(m) = 2 \left\lceil \frac{j}{m} \right\rceil - \left\lceil \frac{j}{m+1} \right\rceil - \left\lceil \frac{j}{m-1} \right\rceil, \quad 2 \leq m \leq n-1.$$

Notice that for a fixed  $m$  the values  $D_j(m)$  can be positive, zero or negative for different values of  $j$ .

By Lemma 6, due to the convexity of the sequence  $B(m)$ ,  $1 \leq m \leq n$ , taking  $f \equiv 1$  we deduce that

$$\sum_{j=1}^q D_j(m) \leq 0 \quad (27)$$

for each  $m$ ,  $2 \leq m \leq n - 1$ , and all  $q$ ,  $1 \leq q \leq n$ .

In order to prove the lemma, we need to demonstrate that the inequality

$$\sum_{j=1}^n p_j D_j(m) \leq 0 \quad (28)$$

holds for each  $m$ ,  $2 \leq m \leq n - 1$ .

Fix an  $m$ ,  $2 \leq m \leq n - 1$ , and transform

$$\begin{aligned} \sum_{j=1}^n p_j D_j(m) &= p_n \left( \sum_{i=1}^n D_i(m) - \sum_{i=1}^{n-1} D_i(m) \right) + p_{n-1} \left( \sum_{i=1}^{n-1} D_i(k) - \sum_{i=1}^{n-2} D_i(k) \right) + \cdots + p_1 D_1(k) \\ &= p_n \sum_{i=1}^n D_i(k) + (p_{n-1} - p_n) \sum_{i=1}^{n-1} D_i(k) + (p_{n-2} - p_{n-1}) \sum_{i=1}^{n-2} D_i(k) + \cdots + p_1 D_1(k) \\ &= \sum_{j=2}^n \left[ (p_{j-1} - p_j) \sum_{i=1}^{j-1} D_i(k) \right] + p_n \sum_{i=1}^n D_i(k). \end{aligned}$$

The last obtained right-hand expression is non-positive due to (7) and (27), so that the desired inequality (28) holds and the sequence  $F(m)$ ,  $1 \leq k \leq n$ , is convex.  $\blacksquare$

Obviously, the sequence  $T(m)$  is also convex since  $T(m) = \frac{1}{2}(T(m-1) + T(m+1)) = Km$  for each  $m$ ,  $2 \leq m \leq n - 1$ .

It is easy to verify that the sum of two convex sequences is convex and any convex sequence is  $V$ -shaped; the analogies of these statements for convex functions are well-known. Thus, Lemma 7 immediately implies

**Theorem 8** *For the problem of minimizing the total cost, the sequence  $\Phi(m)$ ,  $1 \leq m \leq n$ , of values of the objective function is convex and  $V$ -shaped.*

Theorem 8 allows us to find an optimal schedule with an optimal number of machines  $m^*$  by performing binary search with respect to  $m$ . As a result at most  $\lceil \log_2 n \rceil$  values of  $m$  need to be tested, so that the running time of this method becomes  $O(n \log n)$ .

## 5.1 Implication for Safe Helicopter Pickup

We conclude this section with a brief interpretation of the results obtained for the problem of minimizing total flow time in terms of the problem of safe pickup of employees from off-shore installations. The latter problem arises in the oil and gas sector, and its study has been initiated by Qian et al. (2011). In the most general settings, the problem can be formulated as follows. Given a set  $N = \{1, 2, \dots, n\}$  of installations, the pickup demands  $p_j$ ,  $j \in N$ , i.e., the number of people to be taken from installation  $j$ , and the helicopter capacity  $Q$ , find a capacity-feasible flight schedule  $S$  that satisfies total pickup demand and minimizes the *total risk*, measured as the total number of people exposed to takeoffs and landings.

For the purpose of this paper, we ignore the capacity constraints, and assume that the helicopter is large enough to take all people on board. This problem is closely linked with problem  $Pm \parallel \sum C_j$ , in which the jobs are interpreted as installations, the processing times

as pickup demands, the machines as flights of the helicopter, and minimizing the total flow time is equivalent to minimizing the total risk. Additionally, the *non-split* pickup scenario is assumed, under which no installation is visited more than once and the helicopter picks up all  $p_j$  passengers from an installation  $j$ .

Thus,  $F(S^*(m))$  defined by (6) measures the total risk of an optimal flight schedule  $S^*(m)$  that is comprised of  $m$  flights. Clearly, if the number of flights is decreased, the risk increases, i.e.,  $F(S^*(m+1)) > F(S^*(m))$ . Thus, Theorem 3 implies that if the number of flights decreases from  $m+1$  to  $m$ , then the total risk  $F(S^*(m))$  of the resulting flight schedule can be up to  $\frac{m+1}{m}$  times larger than that for the best schedule with  $m+1$  flights. This is especially sensitive if  $m$  is small, which is typical in reality. Indeed, reducing the number of flights from 3 to 2 may increase the total risk up to 150%.

Theorem 8 leads to an  $O(n \log n)$ -time algorithm that determines the optimal number of flights that minimizes the cost function  $\Phi(m) = \alpha F(m) + \beta Km$ , where  $K$  represents the cost of one flight.

## 6 Conclusion

In this paper, we study the influence of adding an extra machine for the classical scheduling problems on identical parallel machines, to minimize the makespan and to minimize the total flow time. We present tight bounds on the machine impact and give algorithms for making a cost-effective choice of the number of the machines. These results may have applications to various areas, including computing and manufacturing. As an illustration, we discuss the implications of our results on the problem of safe helicopter transportation.

## References

- Braun, O., G. Schmidt. 2003. Parallel processor scheduling with limited number of preemptions. *SIAM J. Comput.* **32** 671–680.
- Chen, B. 2004. Parallel machine scheduling for early completion. In: J. Y. T. Leung (ed). *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, Chapman & Hall/CRC, London. 9-175–9-184.
- Conway, R. W, W. L. Maxwell, L. W. Miller. 1967. *Theory of Scheduling*. Addison-Wesley: Reading.
- Dósa, G., Z. Tan. 2010. New upper and lower bounds for online scheduling with machine cost. *Discrete Opti.* **7** 125–135.
- Graham, R. L. 1966. Bounds for certain multiprocessing anomalies. *Bell. Syst. Techn. J.* **45** 1563–1581.
- He, Y., S. Cai. 2002. Semi-online scheduling with machine cost. *J. Comput. Sci. Technol.* **17** 781–787.
- Imreh, C. 2009. Online scheduling with general machine cost functions. *Discrete Appl. Math.* **157** 2070–2077

- Imreh, C., J. Noga. 1999. Scheduling with machine cost. In: D. Hochbaum et al. (eds.). *Proceeding of RANDOM-APPROX'99, Lect. Notes Comput. Sci.* **1671** 168–176.
- Jiang, Y. W, Y. He. 2005. Preemptive online algorithms for scheduling with machine cost. *Acta Informatica.* **41** 315–340.
- Jiang, Y. W, Y. He. 2006. Semi-online algorithms for scheduling with machine costs. *J. Comput. Sci. Technol.* **21** 984–988.
- Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys. 1993. Sequencing and scheduling: algorithms and complexity. In: S. Graves, A.H.G. Rinnooy Kan, P. Zipkin (Eds.) *Handbooks in Operations Research and Management Science, Volume 4, Logistics of Production and Inventory*, North Holland: Amsterdam, 445–522.
- Lee, C. Y., V. A. Strusevich. 2005. Two-machine shop scheduling with an uncapacitated interstage transporter. *IIE Trans.* **37** 725–736.
- McNaughton, R. 1959. Scheduling with deadlines and loss functions. *Manag. Sci.* **6** 1–12.
- Qian, F., V. A. Strusevich, I. Gribkovskaia, Ø. Halskau. 2011. Minimization of passenger takeoff and landing risk in offshore helicopter transportation: Models, approaches and analysis. Report SORG-07-2011, University of Greenwich, London, UK.
- Rustogi, K., V. A. Strusevich. 2011. Convex and V-shaped sequences of sums of functions that depend on ceiling functions, *J. Integer Sequenc.* **14** Article 11.1.5, <http://www.cs.uwaterloo.ca/journals/JIS/VOL14/Strusevich/strusevich2.html>.
- Rustogi, K., V. A. Strusevich. 2011. Single machine scheduling with general positional deterioration and rate-modifying maintenance. *Omega*, In Press, doi:10.1016/j.omega.2011.12.007.
- Winston, W. L. 2003. *Operations Research: Applications and Algorithms*. Duxbury Press.

## APPENDIX A

### PROOF OF LEMMA 1

The proof is similar to that of Theorem 2 in the paper by Rustogi and Strusevich (2011a). Recall that

$$F(S^*(m)) = \sum_{j=1}^n \left\lceil \frac{j}{m} \right\rceil$$

due to (8). As in Section 3, assume that  $n = km + r$ , where  $0 \leq r \leq m - 1$ , and rewrite

$$\begin{aligned} F(S^*(m)) &= \sum_{k=0}^{\lfloor \frac{n}{m} \rfloor - 1} \sum_{r=1}^m \left\lceil \frac{km + r}{m} \right\rceil + \sum_{r=1}^{n - m \lfloor \frac{n}{m} \rfloor} \left\lceil \frac{\lfloor \frac{n}{m} \rfloor m + r}{m} \right\rceil \\ &= \sum_{k=0}^{\lfloor \frac{n}{m} \rfloor - 1} \sum_{r=1}^m \left( k + \left\lceil \frac{r}{m} \right\rceil \right) + \sum_{r=1}^{n - m \lfloor \frac{n}{m} \rfloor} \left( \left\lfloor \frac{n}{m} \right\rfloor + \left\lceil \frac{r}{m} \right\rceil \right). \end{aligned}$$

Since  $\lceil \frac{r}{m} \rceil = 1$ , we deduce

$$\begin{aligned} F(S^*(m)) &= \sum_{k=0}^{\lfloor \frac{n}{m} \rfloor - 1} \sum_{r=1}^m (k+1) + \sum_{r=1}^{n-m \lfloor \frac{n}{m} \rfloor} \left( \lfloor \frac{n}{m} \rfloor + 1 \right) \\ &= m \sum_{k=0}^{\lfloor \frac{n}{m} \rfloor - 1} (k+1) + \left( n - m \lfloor \frac{n}{m} \rfloor \right) \left( \lfloor \frac{n}{m} \rfloor + 1 \right), \end{aligned}$$

and the required formula (9) follows immediately.

## APPENDIX B

### PROOF OF THEOREM 5

The makespan of the obtained schedule  $S_{np}(m)$  is either equal to the largest load of machines  $M_1, \dots, M_g$  or to the the largest load of machines  $M_{g+1}, \dots, M_m$ . In the former case,  $C_{\max}(S_{np}(m)) = p_1$ , i.e., this schedule is optimal. Otherwise, let  $k$  be a job that terminates schedule  $S_{np}(m)$ . Assume that job  $k$  starts at time  $\tau$ , so that

$$C_{\max}(S_{np}(m)) = \tau + p_k.$$

Recall that the greedy nature of Algorithm LS implies that all  $m - g$  machines are permanently busy in the time interval  $[0, \tau]$ , so that

$$\tau \leq \frac{P - \sum_{j=1}^g p_j - p_k}{m - g}.$$

Thus, we deduce

$$C_{\max}(S_{np}(m)) \leq \frac{P - \sum_{j=1}^g p_j}{m - g} + \frac{m - g - 1}{m - g} p_k.$$

Furthermore,  $p_k \leq p_m \leq \dots \leq p_{g+1}$ , i.e.,

$$p_k \leq \frac{\sum_{j=g+1}^m p_j + p_k}{m - g + 1} \leq \frac{P - \sum_{j=1}^g p_j}{m - g + 1}.$$

Finally, we derive

$$\begin{aligned} C_{\max}(S_{np}(m)) &\leq \frac{P - \sum_{j=1}^g p_j}{m - g} + \left( \frac{m - g - 1}{m - g} \right) \frac{P - \sum_{j=1}^g p_j}{m - g + 1} \\ &= \left( 1 + \frac{m - g - 1}{m - g + 1} \right) \frac{P - \sum_{j=1}^g p_j}{m - g} = \left( 2 - \frac{2}{m - g + 1} \right) \frac{P - \sum_{j=1}^g p_j}{m - g}, \end{aligned}$$

which proves (21).

To see that the bound (21) is tight, take a positive integer  $m$  and a non-negative integer  $g$  less than  $m$ . Consider an instance with  $m$  machines and  $m + 1$  jobs, that contains jobs  $1, \dots, g$  with the processing time  $m - g + 1$  each, and jobs  $g + 1, \dots, m + 1$  with the processing

time  $m - g$  each, where  $m$  is a positive integer and  $g$  is a non-negative integer less than  $m$ . We have that  $P = g(m - g + 1) + (m - g + 1)(m - g) = m(m - g + 1)$  and  $p_{\max} = m - g + 1$ , so that  $C_{\max}(S_p^*(m)) = m - g + 1$ .

In an optimal non-preemptive schedule  $S_{np}^*(m)$  each job  $j, 1 \leq j \leq g$ , is assigned to an individual machine; without loss of generality, this can be machine  $M_j$ . The remaining  $m - g + 1$  jobs are processed on machines  $M_{g+1}, \dots, M_m$  and each of these machines will process exactly one job, except one machine that will process two jobs. Notice that this schedule is exactly schedule  $S_{np}(m)$  found by Algorithm 2. It can be checked that

$$\frac{C_{\max}(S_{np}(m))}{\frac{P - \sum_{j=1}^g p_j}{m-g}} = \frac{C_{\max}(S_{np}^*(m))}{C_{\max}(S_p^*(m))} = \left(2 - \frac{2}{m-g+1}\right).$$

Indeed,  $C_{\max}(S_{np}^*(m)) = C_{\max}(S_{np}(m)) = 2m - 2g$  and

$$\frac{P - \sum_{j=1}^g p_j}{m-g} = m - g + 1 = C_{\max}(S_p^*(m)),$$

so that

$$\left(2 - \frac{2}{m-g+1}\right) \frac{P - \sum_{j=1}^g p_j}{m-g} = 2m - 2g = C_{\max}(S_{np}(m)).$$

## APPENDIX C

### PROOF OF COROLLARY 5

The bound (22) emerges if Theorem 5 is applied with  $g = 0$ . On the other hand, from Theorem 5 for  $g \geq 1$ , we get

$$C_{\max}(S_{np}(m)) \leq \left(2 - \frac{2}{m-g+1}\right) \frac{P - \sum_{j=1}^g p_j}{m-g}.$$

Since by definition of  $g$  the inequality

$$p_g > \frac{P - \sum_{j=1}^{g-1} p_j}{m-g+1},$$

holds, we deduce that

$$p_g(m-g) > P - \sum_{j=1}^g p_j,$$

which leads to (23).