Computers & Operations Research 183 (2025) 107085

Contents lists available at ScienceDirect

Computers and Operations Research

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

Replication and sequencing of unreliable jobs on m parallel machines: New results

Alessandro Agnetis [a], Mario Benini [a], Paolo Detti [a], Ben Hermans [b], Marco Pranzo [a], Kevin Schewior [c,d,*]

[a] Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Università di Siena, Italy
[b] Research Center for Operations Research & Statistics (ORSTAT), KU Leuven, Belgium
[c] Department of Mathematics and Computer Science, University of Cologne, Germany
[d] Department of Mathematics and Computer Science, University of Southern Denmark, Denmark

ARTICLE INFO

Keywords:
Scheduling
Approximation algorithms
Unreliable jobs
Unrecoverable breakdowns
Job replication
Submodular optimization

ABSTRACT

This paper gives new results for the problem of sequencing m copies of n unreliable jobs (i.e., jobs that have a certain probability of being successfully carried out) on m parallel machines (one copy per machine). A job is carried out if at least one of its copies is successfully completed, in which case a certain revenue is earned. If the copy of a job fails, the corresponding machine is blocked and cannot perform the subsequently scheduled job copies. The problem is to sequence the n copies of each job on each of the m machines in order to maximize the expected revenue. For the case of $m = 2$, Agnetis et al. (2022) proposed a metaheuristic approach and some upper bounding schemes. Here we address the general m-machine problem, giving a simple $(1 - 1/e)$-approximation algorithm, an even simpler algorithm, for which we show a tight logarithmic approximation guarantee, and an additional heuristic as well as an additional metaheuristic. We provide computational results, which, along with a new upper-bounding scheme, establish the effectiveness of our approaches in practice.

1. Introduction

In this paper we address a scheduling problem, characterized by the fact that *unrecoverable machine breakdowns* may occur. The problem has been first introduced in Agnetis et al. (2022). By "breakdown" we mean that, while processing a job, a processing resource becomes unavailable, whatever the reason (e.g., a real machine failure, or the machine is withdrawn by some higher-level process). In any case, upon a breakdown, the current job and all subsequently scheduled jobs on the machine are lost. In this context, which has been addressed in the computer-science literature (Benoit et al., 2013), the scheduling objective is related to whether or not each job is successfully carried out.

This paper addresses the following scenario. Consider a set $N = \{1, \dots, n\}$ of $n \in \mathbb{N}$ jobs and m identical machines. Jobs are *unreliable*: When a job $j \in N$ is performed on a certain machine, then it succeeds independently with a *success probability* $p_j \in (0, 1) \cap \mathbb{Q}$. If the job fails, a machine breakdown occurs, and the machine cannot perform any further jobs. We say that a job $j \in N$ is *successful on a machine* if and only if all jobs preceding j on that machine as well as job j itself succeed. A specific feature of our problem is that jobs can be *replicated* (Benoit et al., 2013). That is, we assume that there are m copies of each job, such a copy of each job can be attempted on each machine. A job is then said to be *carried out* if and only if *at least one* of its copies is successfully completed on the corresponding machine, which is what one is interested in (e.g., a complex scientific computation independently launched on different machines). Failures are assumed to occur independently, both between jobs and between their copies. Given a revenue $r_j \in \mathbb{Q}_{>0}$ for each job $j \in N$ that is attained if and only if the job is carried out, we study the problem of finding a sequence of the n job copies for each of the m machines so as to maximize the expected revenue.

The problem addressed in this paper is motivated by a context which is close to the one described by Benoit et al. (2013). Consider a certain computing workload and m (identical) machines (e.g., computers). The machines are subject to interruptions of known probability that destroy all work in progress. As an example, in manufacturing, machine failures may depend on the deterioration of the tools (Yang et al., 2024), and the probability that a machine fails during the processing of a job j can

* Corresponding author at: Department of Mathematics and Computer Science, University of Cologne, Germany.
E-mail address: schewior@cs.uni-koeln.de (K. Schewior).

https://doi.org/10.1016/j.cor.2025.107085
Received 14 June 2024; Received in revised form 27 March 2025; Accepted 30 March 2025
Available online 20 May 2025

be estimated by knowing the use time of each tool required by j and the conditions of each tool at the beginning of the unsupervised shift. In a computational environment, such probabilities can possibly be estimated using machine learning; indeed, applying machine learning to estimate properties of code has been explored by Sharma et al. (2024). Benoit et al. (2013) assume that the workload is "divisible", i.e., it is a continuous stream of work whose granularity can be adjusted arbitrarily, but there is essentially no cost in replicating (parts of) the workload on multiple machines. The problem is then how to share the workload among the machines so as to maximize the expected amount of work completed. Here we address a discrete version of this problem, i.e., we consider that granularity is not arbitrary, but rather there are n (indivisible) jobs, which can be individually replicated on all m machines, and the problem is how to sequence the jobs on each machine. Once a job is successfully completed—on any machine—, the corresponding work is saved and its reward is obtained. No further reward derives from completing other copies of the same job. Benoit et al. (2013) consider that the failure process of the computer(s) follows a known distribution, so, in their model, the failure probability changes over time, while we consider that the probability of the computer failing during the execution of a job j is a given value $(1-\pi_j)$. However, we note here that in the relevant case of exponential failures with rate λ, the probability that the machine does not fail during the execution of a job j having processing time q_j is given by $\pi_j = e^{-\lambda q_j}$, regardless its start time. So, exponential failures are captured by our model. A different scenario, addressed by Benoit et al. (2013), is when risk increases linearly with time, which we briefly discuss in Section 9. Moreover, unlike (Benoit et al., 2013), we consider general job rewards, not necessarily given by the corresponding job processing times.

Practical applications where job failures lead to machine breakdowns can arise in manufacturing systems where task execution is subject to an "execution risk". For instance, consider a manufacturing process in a pull production system, where items are produced overnight in an unsupervised environment to fulfill orders for the following day (this scenario is often referred to as *lights-out manufacturing* (Nadimpalli et al., 2025)). Prior to the start of the unsupervised shift, materials are loaded into the feeders, and production proceeds automatically. However, if an item becomes physically stuck in the machine, the machine is blocked, preventing the production of subsequent items. A viable strategy to mitigate this risk is to produce multiple copies of each item, increasing the likelihood of completing the required items by the end of the shift. While the reward is granted only for the first item of each type, possible surplus copies may either be recycled, e.g., in the context of 3D printing (Tian et al., 2017), or be stored in the warehouse to fulfill future orders.

The problem addressed in this paper has been introduced by Agnetis et al. (2022), and is referred to as the *Expected Revenue Maximization* (ERM) problem. When referring to the problem in which m is a constant, we write ERMm. We note here that, if we consider the revenues r_j as monetary amounts, since such revenues are gained as a result of a probabilistic mechanism, a decision maker might prefer to run a job yielding lower expected revenue but higher chances of success. However, as long as we assume that the decision maker follows the classical axioms of rationality (French, 1986), this simply implies that we should regard the values r_j as the *utility* of the monetary amount attained when job j is successfully completed, and more precisely the objective function is to maximize expected utility. For the sake of simplicity, throughout the paper we still use "expected revenue" instead of "expected utility" (or equivalently, if we view the r_j as monetary, we assume that the decision maker is risk-neutral).

In Agnetis et al. (2022), it is proved that ERM2 is NP-hard, and a metaheuristic approach is proposed for ERM2. No worst-case bound is provided, and the quality of the heuristic solution is assessed by means of some upper bounds obtained via mathematical programming. Here we address the more general ERM problem, for which we provide the following new results.

- We analyze the worst-case approximation bound of two simple greedy-like algorithms. For the first algorithm, which runs in time $O(n \log n)$, the bound equals H_m/m, where H_m is the mth harmonic number, and we show that the bound is tight. For the second algorithm, which runs in time $O(mn \log n)$, the bound is $(1 - 1/e)$.
- We devise an additional heuristic based on mutual best replies (local search) as well as an additional metaheuristic based on tabu search.
- We propose a new, easy-to-compute upper bound by solving a variant of ERMm. This upper bound is used to assess the quality of heuristic solutions. We also show that the bound overestimates the optimum value by at most a factor of 2.
- We perform extensive computational experiments to assess the performance of all algorithms introduced. Our results show that it is possible to quickly produce solutions with a small optimality gap even for large values of m, and that the approximation bounds are rather pessimistic in practice.

1.1. Literature review

Machine breakdowns have been studied in quite a few varieties in the scheduling literature (Schmidt, 2000). In contrast to the unrecoverable breakdowns considered here, there are, however, many models in which the breakdowns have a certain (possibly random) duration and, after that, the machine can either continue where it left off (*preempt-resume* models), or it has to restart the last job from scratch (*preempt-repeat* models) (Adiri et al., 1989). In these contexts, the problem is to devise scheduling strategies that optimize the expected value of some classical objective function, e.g., weighted sum of completion times. For instance, (Birge et al., 1990) show that when jobs have (deterministic) processing times p_j and weights w_j, and machine downtimes are identically, exponentially distributed (with parameter λ), in a preempt-repeat setting on a single machine, the expected total weighted completion time is minimized by scheduling the jobs by nonincreasing values of the ratios $w_j/(e^{\lambda p_j}-1)$. Interestingly, this rule coincides exactly with the *Z-rule* introduced in Section 2 when the success probability of a job is simply the probability of the machine not breaking during its execution. Cai et al. (2009) generalized several results for the preempt-repeat model to the case of incomplete information on processing times. Qi et al. (2006) investigate in which cases using an SPT rule is optimal even in a preempt-repeat setting.

The basic problem of maximizing the total expected revenue on a single machine with unreliable jobs and unrecoverable breakdowns has been considered by Stadje (1995). There are two avenues for generalizing this problem to multiple machines. The first avenue is allowing each job to only be scheduled on a single machine. The emerging problem becomes NP-hard even for $m = 2$ (Agnetis et al., 2009). There is a list-scheduling algorithm that is a 0.8535-approximation algorithm for $m = 2$ (Agnetis et al., 2014) and a 0.8531-approximation algorithm for arbitrary m (Agnetis and Lidbetter, 2020). The other avenue is allowing job replication like in the present work. This direction was inspired by Benoit et al. (2013) and, in the present context, initiated by Agnetis et al. (2022). For a more extensive literature review, we refer to Agnetis et al. (2022).

1.2. Overview

The structure of the paper is as follows. In Section 2 we establish notation and review fundamental results. In Section 3, the upper-bounding scheme is described. Sections Section 4, 5, and 6 deal with three different heuristic approaches, while Section 7 presents a metaheuristic approach. Computational experiments are described and discussed in Section 8. Finally, in Section 9, some conclusions are drawn.

2. Preliminaries

In what follows, we represent a *sequence* by a bijection $\sigma : N \to \{1, \dots, n\}$, such that job $i \in N$ comes before job $j \in N$ in the sequence σ if and only if $\sigma(i) < \sigma(j)$. Here and below, we do not distinguish between a job and its copies when it is unlikely to cause confusion. The probability that job j is successful on some machine under the sequence σ then equals

$$P_j(\sigma) = \prod_{i \in N \,:\, \sigma(i) \le \sigma(j)} p_i.$$

A *solution* for ERM will be represented by a multiset $S = \{\sigma_1, \dots, \sigma_m\}$ of m sequences (i.e., one for each machine). The corresponding expected revenue then equals

$$\mathrm{ER}(S) := \sum_{j \in N} \left[1 - \prod_{\sigma \in S} \left(1 - P_j(\sigma)\right) \right] r_j. \tag{1}$$

The number of arithmetic operations required for computing $\mathrm{ER}(S)$ is therefore $O(nm)$, which is equal to the total number of entries in the m arrays with which S can be represented. Recall in this respect that job $j \in N$ is carried out, and thus leads to a revenue r_j, if and only if the job is successful on at least one machine. This latter event occurs with a probability equal to one minus the probability that the job is unsuccessful on all machines.

In the special case where there is only one machine (i.e., $m = 1$), a solution for ERM consists of a single sequence σ. Denoting the expected revenue of such a sequence by $\mathrm{ER}(\sigma)$, Eq. (1) simplifies to

$$\mathrm{ER}(\sigma) = \sum_{j \in N} \left[1 - \left(1 - P_j(\sigma)\right)\right] r_j = \sum_{j \in N} P_j(\sigma) r_j. \tag{2}$$

It is well known that for this special case an optimal sequence can be found by following the so-called *Z-rule* (Stadje, 1995; Agnetis et al., 2009). More specifically, define the *Z-ratio* of a job $j \in N$ as

$$Z_j = \frac{p_j r_j}{1 - p_j},$$

i.e., the expected revenue of the job per probability of failing. It holds that a sequence is optimal if and only if it schedules the jobs in non-increasing order of their Z-ratio. The proof uses a standard pairwise-interchange argument which for the sake of completeness is repeated in the appendix.

Theorem 1 (*Stadje, 1995; Agnetis et al., 2009*)**.** *Consider an ERM instance with a single machine and a sequence $\sigma^\star$. It then holds that $\mathrm{ER}(\sigma^\star) = \max_\sigma \mathrm{ER}(\sigma)$ if and only if $Z_i > Z_j$ implies $\sigma^\star(i) < \sigma^\star(j)$ for all $i, j \in N$.*

Finally, throughout the paper, we use $\mathbb{N} = \{1, 2, \dots\}$ and $\mathbb{N}_0 = \{0, 1, 2, \dots\}$. We give an overview of all frequently used notations in Appendix A.

3. An upper bound for ERM

Consider the relaxation of ERM in which we are allowed to schedule the machines sequentially. More specifically, only when a certain machine gets blocked by a job failure, we need to decide how to schedule the jobs on the next machine. We call this variant the *sequential ERM*. A solution for the sequential ERM is represented by a policy that specifies, for every possible outcome of the jobs on the previous machines, which schedule to follow on the current machine. Observe that in case there is only a single machine, the sequential ERM is equivalent to the original ERM.

The optimal solution value for the sequential ERM provides an upper bound on the optimal value for the original ERM. To see this, observe that, for every ERM solution, one possible policy for the sequential ERM would be to follow on each machine the corresponding schedule specified by the ERM solution, regardless of the outcome of the jobs on the previous machines. That is, even if a job is already carried out by a previous machine and therefore cannot generate any additional revenue, it is still included on the current machine according to the schedule described by the original ERM solution. This static policy leads to the same expected revenue for the sequential ERM as the original solution for the traditional ERM. By consequence, every schedule for the traditional ERM defines a policy for the sequential ERM with the same objective value, and therefore the maximum value for the sequential ERM is at least as large as the maximum value for the original ERM. We refer to the maximum objective value for the sequential ERM as the *sequential upper bound*.

In the remainder of this section, we first show that for the sequential ERM it is an optimal policy to always schedule the jobs that have not yet been carried out in non-increasing order of their Z-ratio (Section 3.1). Next, in Section 3.2, we discuss a number of structural properties of the value of this upper bound that will be useful for the analysis in Section 4.

3.1. Optimality of following the Z-ratio

Theorem 2 below states that for the sequential ERM it is optimal to follow the Z-ratio. More specifically, it is optimal to always schedule next the remaining jobs (i.e., the ones that have not yet been carried out on a previous machine) in non-increasing order of their Z-ratio. Here, we assume without loss of generality that the jobs are indexed in non-increasing order of their Z-ratio. For every $i \in N$ and $k \in \{1, \dots, m\}$, the value $f(i, k)$ reflects the expected revenue when following the Z-ratio if the remaining jobs are $i, \dots, n$ and there are k remaining machines.

Theorem 2. *For an arbitrary instance $(n, m, (r_i, p_i)_{i \in N})$ with $Z_1 \ge \cdots \ge Z_n$, the sequential upper bounds equals $f(1, m)$ as determined by the recursion*

$$f(i, k) = p_i \left[r_i + f(i + 1, k) \right] + (1 - p_i) f(i, k - 1) \tag{3}$$

with boundary conditions $f(n + 1, k) = 0$ and $f(i, 0) = 0$ for all $i \in N$ and $k \in \{1, \dots, m\}$.

Eq. (3) can be interpreted as follows. Since job i has the maximum Z-ratio among all remaining jobs $i, \dots, n$, it is scheduled next. If it succeeds, which occurs with probability p_i, then we obtain revenue r_i and the value $f(i + 1, k)$, reflecting that we still have k machines available to perform the remaining jobs $i + 1, \dots, n$. On the other hand, if job i fails, which occurs with probability $(1 - p_i)$, then we obtain value $f(i, k - 1)$, reflecting that we lost one machine.

Proof. Define a value function $v : 2^N \times \{0, \dots, m\} \to \mathbb{R}$ such that for every $J \subseteq N$ and $k \in \{0, \dots, m\}$ the value $v(J, k)$ equals the maximum expected revenue that one can obtain from a set of remaining jobs J and k remaining machines. For every $J \subseteq N$ with $J \neq \emptyset$ and $k \in \{1, \dots, m\}$, the value function satisfies the optimality equation

$$v(J, k) = \max_{j \in J} \left\{ p_j (r_j + v(J \setminus \{j\}, k)) + (1 - p_j) v(J, k - 1) \right\}$$

with boundary conditions $v(J, 0) = 0$ and $v(\emptyset, k) = 0$.

Now consider an arbitrary $J \subseteq N$ with $J \neq \emptyset$ and $k \in \{1, \dots, m\}$. Moreover, let $i \in J$ be such that it has the maximum Z-ratio among all jobs in J, i.e.,

$$\frac{p_i r_i}{1 - p_i} = \max_{j \in J} \frac{p_j r_j}{1 - p_j}.$$

We then claim that

$$v(J, k) = p_i (r_i + v(J \setminus \{i\}, k)) + (1 - p_i) v(J, k - 1), \tag{4}$$

such that following the Z-rule is indeed optimal. In particular, this would establish that $f(i, k) = v(\{i, \dots, n\}, k)$ for every $i \in N$ and $k \in \{1, \dots, m\}$, and that the recursion as specified by Eq. (3) is indeed correct. Hence, to complete the proof, it remains to be shown that Eq. (4) holds.

We prove Eq. (4) using induction on $k+|J|$. For $k=1$, we are in the single-machine case, and it is optimal to follow the Z-rule by Theorem 1. For $|J|=1$, the claim also holds trivially, since there is only a single job that can be performed. Hence, suppose that $k>1$ and $|J|>1$, and, as the induction hypothesis, that our claim holds for the case of $k-1$ remaining machines or $|J|-1$ remaining jobs. Consider an arbitrary job $j\in J\setminus\{i\}$ and observe that, by definition of the value function and the induction hypothesis,

$$\begin{aligned}
&p_i(r_i+v(J\setminus\{i\},k))+(1-p_i)v(J,k-1)\\
&\quad\geq p_i\left[r_i+p_j(r_j+v(J\setminus\{i,j\},k))+(1-p_j)v(J\setminus\{i\},k-1)\right]\\
&\qquad+(1-p_i)\left[p_j(r_j+v(J\setminus\{j\},k-1))+(1-p_j)v(J,k-2)\right]\\
&\quad=p_ir_i+p_jr_j+p_ip_jv(J\setminus\{i,j\},k)+p_i(1-p_j)v(J\setminus\{i\},k-1)\\
&\qquad+(1-p_i)p_jv(J\setminus\{j\},k-1)+(1-p_i)(1-p_j)v(J,k-2)
\end{aligned}$$

and

$$\begin{aligned}
&p_j(r_j+v(J\setminus\{j\},k))+(1-p_j)v(J,k-1)\\
&\quad=p_j\left[r_j+p_i(r_i+v(J\setminus\{i,j\},k))+(1-p_i)v(J\setminus\{j\},k-1)\right]\\
&\qquad+(1-p_j)\left[p_i(r_i+v(J\setminus\{i\},k-1))+(1-p_i)v(J,k-2)\right]\\
&\quad=p_ir_i+p_jr_j+p_ip_jv(J\setminus\{i,j\},k)+p_i(1-p_j)v(J\setminus\{i\},k-1)\\
&\qquad+(1-p_i)p_jv(J\setminus\{j\},k-1)+(1-p_i)(1-p_j)v(J,k-2).
\end{aligned}$$

From this, we obtain that

$$\begin{aligned}
&p_i(r_i+v(J\setminus\{i\},k))+(1-p_i)v(J,k-1)\\
&\quad\geq p_j(r_j+v(J\setminus\{j\},k))+(1-p_j)v(J,k-1),
\end{aligned}$$

showing that Eq. (4) holds, and therefore completing the proof. □

3.2. Structural properties of the upper bound

The next three lemmas provide insight into the structural properties of the optimal objective value for the sequential ERM, and they will help with developing the subsequent results in Section 4. Throughout, we consider an arbitrary instance $(n,m,(r_i,p_i)_{i\in N})$ with $Z_1\geq\cdots\geq Z_n$. It then follows from Theorem 2 that $f(i,k)$ equals the maximum expected revenue for the instance defined on job set $\{i,\dots,n\}$ and with $k\geq 1$ remaining machines. Lemma 1 states that for an instance with a single machine the maximum Z-ratio upper bounds the maximum expected revenue. Lemma 2, in turn, states that the average expected revenue per machine is non-increasing in the number of machines. Lemma 3, finally, implies that before performing a job with maximum Z-ratio the expected revenue is not smaller than when the job has been carried out. The proofs of the lemmas are reported in Appendix A.

Lemma 1. *$f(i,1)\leq Z_i$ for all $i\in N$.*

Lemma 2. *$\dfrac{f(i,k)}{k}\leq\dfrac{f(i,\ell)}{\ell}$ for all $i\in N$ and $k\geq\ell\geq 1$.*

Lemma 3. *$f(i,k)\geq f(i+1,k)$ for all $i\in N\setminus\{n\}$ and $k\geq 0$.*

3.3. A bound on the sequentiality gap

In this section, we prove a bound on the sequentiality gap (a concept akin to the adaptivity gap, see Dean et al., 2008), i.e., the smallest-possible ratio between the expected revenues of an optimal schedule for ERM and sequential ERM. While our proof is of algorithmic nature, the implied algorithmic result is superseded later. Indeed, we view the following as a result of structural nature.

Theorem 3. *For any instance $(n,m,(r_i,p_i)_{i\in N})$, there exists a solution S with*

$$\mathrm{ER}(S)\geq\frac{1}{2}\cdot f(1,m).$$

Proof. We will show that there exists a randomized solution $\tilde{S}$. To this end, we denote by $\mathbb{E}[X]$ the expected value of a random variable X and by $\Pr[\mathcal{E}]$ the probability of an event $\mathcal{E}$. The randomized solution $\tilde{S}$ will fulfill

$$\mathbb{E}[\mathrm{ER}(\tilde{S})]\geq\frac{1}{2}\cdot f(1,m), \tag{5}$$

implying that there exists a (deterministic) solution S with the desired property. We obtain $\tilde{S}$ by *simulating* the optimal policy for sequential ERM (i.e., following the Z-ratio as described in Section 3.1). That is, the successes of all copies of a job are evaluated in order to obtain $\tilde{S}$, and then again independently to evaluate $\mathrm{ER}(\tilde{S})$. Note that the optimal policy for sequential ERM may sequence only a suffix of the jobs $\{1,\dots,n\}$ on some machine because the other jobs have been successful already. In $\tilde{S}$, we move such jobs to the back of the sequence on the corresponding machines, in arbitrary order. In the analysis, we will not use their existence.

For any job j, we denote by $\mathcal{E}_{i,j}$ the event that j is successful for the first time on machine i, and by $\mathcal{E}'_{i,j}$ the event that it is successful in the *simulation* of the optimal policy for sequential ERM. (Note that, for the latter event, it does not make a difference whether we require j to be successful on i *for the first time*—in this simulation, j can only be successful once.) We will show that, for all jobs j and all $i\in\{0,\dots,m\}$,

$$\Pr[\mathcal{F}_{i,j}]\geq\frac{1}{2}\cdot\Pr[\mathcal{F}_{i,j}\vee\mathcal{F}'_{i,j}] \tag{6}$$

using the notation

$$\mathcal{F}_{i,j}:=\bigvee_{k=1}^{i}\mathcal{E}_{k,j}\quad\text{and}\quad\mathcal{F}'_{i,j}:=\bigvee_{k=1}^{i}\mathcal{E}'_{k,j}.$$

This will imply the claim: Noting that a weaker version of the inequality is $\Pr[\mathcal{F}_{i,j}]\geq 1/2\cdot\Pr[\mathcal{F}'_{i,j}]$, setting $i=m$ in this inequality, multiplying it with r_j, and summing the resulting inequality over all j, we obtain Inequality (5).

So it remains to show Inequality (6). We do so by fixing a job j and inducing on i. The statement is trivial for $i=0$. Now assume that it holds for $i=0,\dots,i^\star-1$ where $i^\star\in\{1,\dots,m\}$; we show it for $i=i^\star$. It suffices to show

$$\Pr[\mathcal{E}_{i^\star,j}\mid\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\geq\frac{1}{2}\cdot\Pr[\mathcal{E}_{i^\star,j}\vee\mathcal{E}'_{i^\star,j}\mid\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})], \tag{7}$$

i.e., that $\mathcal{F}_{i^\star,j}$ covers a 1/2-fraction of the part of the probability space newly covered by $\mathcal{F}_{i^\star,j}\vee\mathcal{F}'_{i^\star,j}$. More formally, we indeed have

$$\begin{aligned}
\Pr[\mathcal{F}_{i^\star,j}]&=\Pr[\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}]\cdot\Pr[\mathcal{F}_{i^\star,j}\mid\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}]\\
&\quad+\Pr[\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\cdot\Pr[\mathcal{F}_{i^\star,j}\mid\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\\
&\geq\Pr[\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}]\cdot\Pr[\mathcal{F}_{i^\star-1,j}\mid\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}]\\
&\quad+\Pr[\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\cdot\Pr[\mathcal{E}_{i^\star,j}\mid\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\\
&\geq\Pr[\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}]\cdot\frac{1}{2}\cdot\Pr[\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}\mid\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}]\\
&\quad+\Pr[\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\cdot\frac{1}{2}\cdot\Pr[\mathcal{E}_{i^\star,j}\vee\mathcal{E}'_{i^\star,j}\mid\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\\
&=\Pr[\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}]\cdot\frac{1}{2}\cdot\Pr[\mathcal{F}_{i^\star,j}\vee\mathcal{F}'_{i^\star,j}\mid\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j}]\\
&\quad+\Pr[\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\cdot\frac{1}{2}\cdot\Pr[\mathcal{F}_{i^\star,j}\vee\mathcal{F}'_{i^\star,j}\mid\neg(\mathcal{F}_{i^\star-1,j}\vee\mathcal{F}'_{i^\star-1,j})]\\
&=\frac{1}{2}\cdot\Pr[\mathcal{F}_{i^\star,j}\vee\mathcal{F}'_{i^\star,j}],
\end{aligned}$$

where we use the law of total expectation in the first and last step, the definitions of $\mathcal{E}_{i,j}$ and $\mathcal{F}_{i,j}$ in the second and second-to-last step, and the induction hypothesis as well as Eq. (7) in the third step. Recall that condition imposed on both sides of Eq. (7) means that j is successful neither in reality nor in the simulation on any of the machines $1,\dots,i^\star-1$. Since, by construction, the sequence on machine $i^\star$ is identical in reality and in the simulation, $\mathcal{E}_{i^\star,j}$ and $\mathcal{E}'_{i^\star,j}$ occur with identical marginal probability under the imposed condition. Therefore, it holds that

Algorithm 1: Z-rule heuristic (ZRH).

input: An ERM instance $(n, m, (r_i, p_i)_{i\in N})$

for all j, compute $Z_j = p_j r_j/(1-p_j)$;
let σ be the sequence that schedules the jobs $j \in N$ in non-increasing order of Z_j;
let $S \leftarrow \{\sigma, \sigma, \dots, \sigma\}$ (m times);
return S;

$$\begin{aligned}\Pr[\mathcal{E}_{i^*,j} \mid \neg(\mathcal{F}_{i^*-1,j} \vee \mathcal{F}'_{i^*-1,j})] &= \frac{1}{2}\cdot\big(\Pr[\mathcal{E}_{i^*,j} \mid \neg(\mathcal{F}_{i^*-1,j} \vee \mathcal{F}'_{i^*-1,j})] \\ &\quad + \Pr[\mathcal{E}'_{i^*,j} \mid \neg(\mathcal{F}_{i^*-1,j} \vee \mathcal{F}'_{i^*-1,j})]\big) \\ &\geq \frac{1}{2}\cdot\big(\Pr[\mathcal{E}_{i^*,j} \mid \neg(\mathcal{F}_{i^*-1,j} \vee \mathcal{F}'_{i^*-1,j})] \\ &\quad + \Pr[\mathcal{E}'_{i^*,j} \mid \neg(\mathcal{F}_{i^*-1,j} \vee \mathcal{F}'_{i^*-1,j})] \\ &\quad - \Pr[\mathcal{E}_{i^*,j} \wedge \mathcal{E}'_{i^*,j} \mid \neg(\mathcal{F}_{i^*-1,j} \vee \mathcal{F}'_{i^*-1,j})]\big) \\ &= \frac{1}{2}\cdot\Pr[\mathcal{E}_{i^*,j} \vee \mathcal{E}'_{i^*,j} \mid \neg(\mathcal{F}_{i^*-1,j} \vee \mathcal{F}'_{i^*-1,j})].\end{aligned}$$

This completes the proof. □

We note that our proof also provides a randomized 2-approximation algorithm for ERM that runs in polynomial time. We will later give a simpler deterministic algorithm, for which we will show a stronger approximation guarantee.

We finish the section with observing that Inequality (5) is essentially tight for the schedule $\tilde{S}$ we are constructing. To see this, consider the instance in which $n = 2$, $r_1 = 1$, $p_1 = 1/\sqrt{m}$, $r_2 = p_2 = 0$ (note that $p_j = 0$ and $r_j = 0$ can easily be avoided by choosing a tiny probability and revenue, respectively), and $m \to \infty$. Note that $Z_1 > Z_2$, so

$$\lim_{m\to\infty} f(1,m) = \lim_{m\to\infty}\left(1-\left(1-\frac{1}{\sqrt{m}}\right)^m\right) = 1.$$

On the other hand, $\mathrm{ER}(\tilde{S})$ is the probability that job 1 is successful in $\tilde{S}$. Let X_1 and Y_1 be the number of trials job 1 needs to be successful in reality and in the simulation, respectively. Both values are i.i.d. draws from the same (negative binomial) distribution and possibly larger than m. Overall, we indeed have

$$\begin{aligned}\lim_{m\to\infty}\Pr[\text{job 1 successful in } \tilde{S}] &= \lim_{m\to\infty}\Pr[X_1 \leq m \wedge X_1 \leq Y_1] \\ &\leq \lim_{m\to\infty}\Pr[X_1 \leq Y_1] \\ &= \lim_{m\to\infty}(\Pr[X_1 < Y_1] + \Pr[X_1 = Y_1]) \\ &= \lim_{m\to\infty}(\Pr[X_1 < Y_1]) \\ &\leq \frac{1}{2},\end{aligned}$$

where we use in the second-to-last step that the success probability approaches 0 as $m \to \infty$ and in the last step that X_1 and Y_1 are i.i.d. draws.

4. The Z-rule heuristic

In this section we analyze the worst-case performance of the heuristic that schedules the jobs in non-increasing order of their Z-ratio on all machines (Algorithm 1), hence requiring $O(n \log n)$ time. We refer to this heuristic as the *Z-rule heuristic*, abbreviated as ZRH. For an arbitrary instance, we compare the objective value obtained by this heuristic with the sequential upper bound.

For every $m \geq 1$, let $H_m = \sum_{k=1}^{m} 1/k$ be the mth harmonic number, and denote $\alpha_m = H_m/m$. That is, α_m is the inverse harmonic mean of the integers $1, \dots, m$. Table 1 lists the value of α_m for $m \in \{1, \dots, 20\}$. In Section 4.1 we show that, for every instance with m machines, the Z-rule heuristic leads to an expected revenue that is at least α_m times the sequential upper bound. Next, in Section 4.2, we provide a family of instances showing that the derived bound between the heuristic and this sequential upper bound is asymptotically tight.

Table 1
Approximation guarantee α_m for ERMm with $m \in \{1, \dots, 20\}$.

m	α_m	m	α_m	m	α_m	m	α_m
1	1	6	0.4083	11	0.2745	16	0.2113
2	0.75	7	0.3704	12	0.2586	17	0.2023
3	0.611	8	0.3397	13	0.2446	18	0.1942
4	0.5208	9	0.3143	14	0.2323	19	0.1867
5	0.4567	10	0.2929	15	0.2212	20	0.1799

Since $\alpha_m = H_m/m$ decreases in m, the obtained approximation guarantee is only constant for a fixed number of machines. Observe in this respect that the approximation guarantee compares the objective value of the heuristic with the sequential upper bound, and not the true maximum expected revenue for the original ERM. Our result on the sequentially gap (Theorem 3), however, implies that the gap between the heuristic and the optimal solution is also $\Theta(m/\log m)$. Indeed, if there are more machines, then following the same sequence on every machine, as in the Z-rule heuristic, seems quite harmful.

4.1. Worst-case performance guarantee

Consider an arbitrary instance $(n, m, (r_i, p_i)_{i\in N})$ with $Z_1 \geq \cdots \geq Z_n$. The expected revenue of scheduling the jobs in non-increasing order of their Z-ratio on all machines then equals $h(1, m)$ as determined by the recursion

$$h(i,k) = \sum_{\ell=1}^{k}\binom{k}{\ell} p_i^{\ell}(1-p_i)^{k-\ell}\left[r_i + h(i+1,\ell)\right] \tag{8}$$

with boundary condition $h(n+1, k) = 0$, for all $i \in N$ and $k \in \{1, \dots, m\}$. The interpretation of Eq. (8) is as follows. For arbitrary $i \in N$ and $k \in \{1, \dots, m\}$, consider the instance defined on job set $\{i, \dots, n\}$ with k available machines. Since job i has maximal Z-ratio among all jobs $i, \dots, n$, it is scheduled first on all available machines, and only those machines on which job i succeeds remain available afterwards. The number of available machines to perform jobs $i+1, \dots, n$ therefore follows a binomial distribution with k trials and success probability p_i. Moreover, we receive the revenue r_i if and only if job i is successful on at least one machine. Hence, if job i is successful on exactly $\ell \in \{1, \dots, k\}$ machines, which occurs with probability $\binom{k}{\ell} p_i^{\ell}(1-p_i)^{k-\ell}$, then we receive the revenue r_i as well as the expected revenue $h(i+1, \ell)$ from the instance defined on job set $\{i+1, \dots, n\}$ with ℓ available machines. If job i fails on all k machines, on the other hand, then the revenue equals zero. Eq. (8) therefore indeed correctly specifies the expected revenue associated with the Z-rule heuristic.

In order to derive the worst-case performance guarantee, it is convenient to rewrite Eq. (8). Consider an arbitrary $i \in N$ and $k \in \{2, \dots, m\}$. Using Pascal's identity, i.e., the fact that $\binom{k}{\ell} = \binom{k-1}{\ell-1} + \binom{k-1}{\ell}$ for all $\ell = 1, \dots, k-1$, where $\binom{k-1}{0} = 1$, we can rewrite Eq. (8) as follows:

$$\begin{aligned}h(i,k) &= p_i^k\left[r_i + h(i+1,k)\right] \\ &\quad + \sum_{\ell=1}^{k-1}\left[\binom{k-1}{\ell-1} + \binom{k-1}{\ell}\right] p_i^{\ell}(1-p_i)^{k-\ell}\left[r_i + h(i+1,\ell)\right] \\ &= p_i^k\left[r_i + h(i+1,k)\right] + p_i\sum_{\ell=1}^{k-1}\binom{k-1}{\ell-1} p_i^{\ell-1}(1-p_i)^{k-\ell}\left[r_i + h(i+1,\ell)\right] \\ &\quad + (1-p_i)\sum_{\ell=1}^{k-1}\binom{k-1}{\ell} p_i^{\ell}(1-p_i)^{k-1-\ell}\left[r_i + h(i+1,\ell)\right] \\ &= p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell} p_i^{\ell}(1-p_i)^{k-1-\ell}\left[r_i + h(i+1,\ell+1)\right] + (1-p_i)h(i,k-1) \\ &= p_i r_i + p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell} p_i^{\ell}(1-p_i)^{k-1-\ell} h(i+1,\ell+1) + (1-p_i)h(i,k-1).\end{aligned} \tag{9}$$

Here, the third equality results from merging the first two terms of the second equality (where the indices of the summation have been shifted down one unit), and from applying Eq. (8) for $k-1$. The fourth equality, in turn, follows since

$$\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}=1, \tag{10}$$

because it is the total probability mass of a binomial distribution with $k-1$ trials and success probability p_i.

Theorem 4, which forms the main result of this section, states that for every instance with m machines, the Z-rule heuristic leads to an expected revenue that is at least $\alpha_m = H_m/m$ times the sequential upper bound, where $H_m = \sum_{\ell=1}^{m} 1/\ell$ is the mth harmonic number. To show this result, we use the following lemma proved in the appendix.

Lemma 4. *For every $k \in \{1,\dots,m\}$ and $p \in (0,1)$ it holds that*

$$(1-p)\left(1-\frac{1}{k}\right)+p\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p^{\ell}(1-p)^{k-1-\ell}H_{\ell+1}\geq pH_k. \tag{11}$$

We are now in the position to show the theorem.

Theorem 4. *Consider an arbitrary instance $(n,m,(r_i,p_i)_{i\in N})$ with $Z_1 \geq \cdots \geq Z_n$. For every $i \in N$ and $k \in \{1,\dots,m\}$ it then holds that*

$$\frac{h(i,k)}{f(i,k)}\geq \alpha_k=\frac{H_k}{k}=\frac{1}{k}\sum_{\ell=1}^{k}\frac{1}{\ell}. \tag{12}$$

In particular, $h(1,m)\geq \alpha_m f(1,m)$.

Proof. We establish Inequality (12) using induction on i and k. For $k=1$, it follows from expanding the recursions defined by Eqs. (3) and (8) that, for every $i \in N$,

$$h(i,1)=p_i\left[r_i+h(i+1,1)\right]=\sum_{j=i}^{n}\left(\prod_{u=i}^{j}p_u\right)r_j=f(i,1).$$

Since $\alpha_1 = 1$, this shows that Inequality (12) indeed holds if $k=1$. Analogously, for $i=n$, it follows from expanding Eqs. (3) and (8) that, for every $k=1,\dots,m$,

$$\begin{aligned}h(n,k)&=\sum_{\ell=1}^{k}\binom{k}{\ell}p_n^{\ell}(1-p_n)^{k-\ell}r_n=(1-(1-p_n)^k)r_n\\&=\sum_{\ell=1}^{k}(1-p_n)^{k-\ell}p_nr_n=f(n,k).\end{aligned}$$

Since $\alpha_k \leq 1$ for all $k \geq 1$, this shows that Inequality (12) indeed holds if $i=n$.

Now consider arbitrary $i \in N\setminus\{n\}$ and $k \in \{2,\dots,m\}$. As the induction hypothesis, assume that $h(i,k-1)\geq \alpha_{k-1}f(i,k-1)$ and $h(i+1,\ell)\geq \alpha_\ell f(i+1,\ell)$ for all $\ell=1,\dots,k$. Eq. (9) then yields

$$\begin{aligned}h(i,k)&\geq p_ir_i+p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}\alpha_{\ell+1}f(i+1,\ell+1)\\&\quad+(1-p_i)\alpha_{k-1}f(i,k-1)\\&=\alpha_kp_ir_i+(1-\alpha_k)p_ir_i+p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}\alpha_{\ell+1}f(i+1,\ell+1)\\&\quad+\alpha_k(1-p_i)f(i,k-1)+(\alpha_{k-1}-\alpha_k)(1-p_i)f(i,k-1).\end{aligned} \tag{13}$$

We now use Lemmas 1–3 to bound the right-hand side of the above inequality in terms of $f(i+1,k)/k$. In particular, Lemmas 1 and 2 imply that

$$p_ir_i=(1-p_i)Z_i\geq(1-p_i)Z_{i+1}\geq(1-p_i)f(i+1,1)\geq(1-p_i)\frac{f(i+1,k)}{k}. \tag{14}$$

Moreover, for every $\ell=0,\dots,k-1$, Lemma 2 and the fact that $\alpha_{\ell+1}=H_{\ell+1}/(\ell+1)$ yield

$$\alpha_{\ell+1}f(i+1,\ell+1)=H_{\ell+1}\frac{f(i+1,\ell+1)}{\ell+1}\geq H_{\ell+1}\frac{f(i+1,k)}{k}. \tag{15}$$

Next, observe that

$$\begin{aligned}\alpha_{k-1}-\alpha_k&=\frac{1}{k-1}\left(H_{k-1}-\frac{k-1}{k}H_k\right)\\&=\frac{1}{k-1}\left(\alpha_k+H_{k-1}-H_k\right)=\frac{1}{k-1}\left(\alpha_k-\frac{1}{k}\right)\end{aligned}$$

and, by Lemmas 2 and 3, that

$$f(i,k-1)\geq f(i+1,k-1)\geq(k-1)\frac{f(i+1,k)}{k}.$$

Hence, we also have that

$$(\alpha_{k-1}-\alpha_k)(1-p_i)f(i,k-1)\geq\left(\alpha_k-\frac{1}{k}\right)(1-p_i)\frac{f(i+1,k)}{k}. \tag{16}$$

Substituting Inequalities (14)–(16) into Inequality (13) and rearranging terms then yields

$$\begin{aligned}h(i,k)\geq\alpha_kp_ir_i+\alpha_k(1-p_i)f(i,k-1)+\Bigg[&\left(1-\alpha_k+\alpha_k-\frac{1}{k}\right)(1-p_i)\\&+p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}H_{\ell+1}\Bigg]\frac{f(i+1,k)}{k}.\end{aligned}$$

On the other hand, it follows from Eq. (3) that

$$\begin{aligned}\alpha_kf(i,k)&=\alpha_kp_ir_i+\alpha_kp_if(i+1,k)+\alpha_k(1-p_i)f(i,k-1)\\&=\alpha_kp_ir_i+\alpha_k(1-p_i)f(i,k-1)+p_iH_k\frac{f(i+1,k)}{k}.\end{aligned}$$

Hence, to establish that $h(i,k)\geq\alpha_kf(i,k)$, it suffices to show that

$$(1-p_i)\left(1-\frac{1}{k}\right)+p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}H_{\ell+1}\geq p_iH_k.$$

Lemma 4 states that this inequality indeed holds, which completes the proof. □

4.2. Asymptotic tightness of the performance guarantee

Consider the family of ERM instances in which all jobs have unit revenue and identical success probability. Observe that an instance $(n,m,(1,p)_{i\in N})$ from this family is completely specified by the parameters $n,m \in \mathbb{N}_0$ and $p \in (0,1)$. Let $F(n,p,m)$ and $H(n,p,m)$ be the corresponding objective value of the sequential upper bound and Z-rule heuristic, respectively.

Theorem 5 below shows that, for every $m \in \mathbb{N}_0$, the ratio between $H(n,p,m)$ and $F(n,p,m)$ tends to α_m as n grows to infinity and p tends to one. As such, it establishes that the worst-case performance guarantee derived in Section 4.1 is asymptotically tight. Before proving this result, we first characterize the values of $F(n,p,m)$ and $H(n,p,m)$ as n tends to infinity in the next lemma, whose proof is deferred to the appendix.

Lemma 5. *For arbitrary $m \in \mathbb{N}_0$ and $p \in (0,1)$, define*

$$F(p,m)=m\frac{p}{1-p} \tag{17}$$

and let $H(p,m)$ be as defined by the recursion, for $k=1,\dots,m$,

$$H(p,k)=\frac{p^k}{1-p^k}+\sum_{\ell=1}^{k-1}\binom{k}{\ell}\frac{p^{\ell}(1-p)^{k-\ell}}{1-p^k}\left[1+H(p,\ell)\right]. \tag{18}$$

It then holds that $\lim_{n\to\infty}F(n,p,m)=F(p,m)$ and $\lim_{n\to\infty}H(n,p,m)=H(p,m)$.

Now we are in the position to prove that the worst case result of Section 4.1 is asymptotically tight.

Theorem 5. *For every $m \in \mathbb{N}_0$ it holds that*

$$\lim_{p\to1}\lim_{n\to\infty}\frac{H(n,p,m)}{F(n,p,m)}=\alpha_m.$$

Proof. Given the result of Lemma 5, it suffices to show that

$$\lim_{p\to 1}\frac{H(p,m)}{F(p,m)}=\alpha_m \tag{19}$$

for every $m\in\mathbb{N}_0$. We show this using induction on m. Since $F(p,1)=H(p,1)$ for all $p\in(0,1)$, we have

$$\lim_{p\to 1}\frac{H(p,1)}{F(p,1)}=1=\alpha_1,$$

so Eq. (19) holds for $m=1$. Now consider arbitrary $m>1$ and, as the induction hypothesis, assume that

$$\lim_{p\to 1}\frac{H(p,k)}{F(p,k)}=\alpha_k$$

for all $k=1,\dots,m-1$. Eqs. (17)–(18) imply that

$$\begin{aligned}&\lim_{p\to 1}\frac{H(p,m)}{F(p,m)}\\&=\lim_{p\to 1}\left[\frac{p^m}{(1-p^m)F(p,m)}+\sum_{\ell=1}^{m-1}\binom{m}{\ell}\frac{p^\ell(1-p)^{m-\ell}}{1-p^m}\left(\frac{1}{F(p,m)}+\frac{H(p,\ell)}{F(p,m)}\right)\right]\\&=\lim_{p\to 1}\left[\frac{p^m(1-p)}{mp(1-p^m)}+\frac{1}{m}\sum_{\ell=1}^{m-1}\binom{m}{\ell}\frac{p^\ell(1-p)^{m-\ell}}{1-p^m}\left(\frac{1-p}{p}+\ell\frac{H(p,\ell)}{F(p,\ell)}\right)\right],\end{aligned} \tag{20}$$

where in the second equality we use that $F(p,m)=mp/(1-p)=(m/\ell)F(p,\ell)$ for all $\ell=1,\dots,m$. Now observe that, using l'Hopital's rule,

$$\lim_{p\to 1}\frac{p^m(1-p)}{p(1-p^m)}=\lim_{p\to 1}\frac{mp^{m-1}-(m+1)p^m}{1-(m+1)p^m}=\frac{1}{m}$$

and, for every $\ell=1,\dots,m-1$,

$$\begin{aligned}\lim_{p\to 1}\frac{p^\ell(1-p)^{m-\ell}}{1-p^m}&=\lim_{p\to 1}\frac{\ell p^{\ell-1}(1-p)^{m-\ell}-(m-\ell)p^\ell(1-p)^{m-\ell-1}}{-mp^{m-1}}\\&=\begin{cases}0 & \text{for } \ell=1,\dots,m-2;\\ \frac{1}{m} & \text{for } \ell=m-1.\end{cases}\end{aligned}$$

Substituting this into Eq. (20) and using the induction hypothesis, the definition of α_m, and the fact that $\lim_{p\to 1}(1-p)/p=0$, we obtain that

$$\lim_{p\to 1}\frac{H(p,m)}{F(p,m)}=\frac{1}{m^2}+\frac{m-1}{m}\alpha_{m-1}=\alpha_m,$$

which completes the proof. □

5. Modified Z-rule heuristic

One potential weakness of the Z-rule heuristic is that it leads to a solution that follows the same sequence on each machine. Because of this, jobs occurring early on in the sequence have a relatively high probability to be successful on multiple machines, whereas jobs occurring later on in the sequence have a relatively high probability not to be successful on any machine. In this section, we consider a construction heuristic based on the so-called modified Z-ratio. This ratio was introduced by Agnetis et al. (2022) for the special case of two machines and takes into account the probability that a job might already be successful on another machine.

Given a multiset $S=\{\sigma_1,\dots,\sigma_k\}$ of $k\in\{1,\dots,m\}$ sequences and a job $j\in N$, we define the *modified Z-ratio* of j given the partial solution S as

$$Z_j(S)=Z_j\prod_{\sigma\in S}\left(1-P_j(\sigma)\right)=\frac{p_jr_j}{1-p_j}\prod_{\sigma\in S}\left(1-P_j(\sigma)\right).$$

In words, the modified Z-ratio equals the expected revenue of performing job j on a single machine times the probability that j fails on all sequences in S, divided by j's failure probability. Observe that, if $S=\emptyset$, then the modified Z-ratio reduces to the original Z-ratio. If $|S|=1$, in turn, then it coincides with the modified Z-ratio as defined in Agnetis et al. (2022) for ERM2, i.e., ERM constrained to two machines.

Algorithm 2: Modified Z-rule heuristic (MZRH).

input: An ERM instance $(n,m,(r_i,p_i)_{i\in N})$

initialize $S\leftarrow\emptyset$;
while $|S|<m$, **do**
  let σ be a sequence that schedules the jobs $j\in N$ in non-increasing order of $Z_j(S)$;
  let $S\leftarrow S\uplus\{\sigma\}$;
return S;

Before describing the heuristic, we first state the following result that motivates its description. It shows that if $m-1$ sequences are fixed, then for the remainder machine it is optimal to schedule the jobs in non-increasing order of their modified Z-ratio. As such, it generalizes the corresponding result in Agnetis et al. (2022) for ERM2. Here and below, we use the additive union, with symbol $\uplus$, to merge two multisets such that the multiplicities of their elements are summed up. For example, with $S=\{\sigma\}$ for some arbitrary sequence σ, we have that $S\uplus\{\sigma\}=\{\sigma,\sigma\}$.

Theorem 6. *For a multiset $S=\{\sigma_1,\dots,\sigma_{m-1}\}$ of $m-1$ sequences and a sequence σ',*

$$\mathrm{ER}(S\uplus\{\sigma'\})=\max_{\sigma}\mathrm{ER}(S\uplus\{\sigma\})$$

if and only if $Z_i(S)>Z_j(S)$ implies $\sigma'(i)<\sigma'(j)$ for all jobs $i,j\in N$.

Proof. Eq. (1) yields that

$$\begin{aligned}\mathrm{ER}(S\uplus\{\sigma'\})&=\sum_{j\in N}\left[1-\prod_{\sigma\in S\uplus\{\sigma'\}}\left(1-P_j(\sigma)\right)\right]r_j\\&=\sum_{j\in N}\left[1\pm\prod_{\sigma\in S}\left(1-P_j(\sigma)\right)-\left(1-P_j(\sigma')\right)\prod_{\sigma\in S}\left(1-P_j(\sigma)\right)\right]r_j\\&=\sum_{j\in N}\left[1-\prod_{\sigma\in S}\left(1-P_j(\sigma)\right)\right]r_j\\&\quad+\sum_{j\in N}\left[1-\left(1-P_j(\sigma')\right)\right]r_j\prod_{\sigma\in S}\left(1-P_j(\sigma)\right).\end{aligned}$$

Denoting $C(S)=\sum_{j\in N}\left[1-\prod_{\sigma\in S}\left(1-P_j(\sigma)\right)\right]r_j$ and $r'_j=r_j\prod_{\sigma\in S}\left(1-P_j(\sigma)\right)$ for each $j\in N$, we obtain that

$$\mathrm{ER}(S\uplus\{\sigma'\})=C(S)+\sum_{j\in N}P_j(\sigma')r'_j.$$

Here, the quantity $\sum_{j\in N}P_j(\sigma')r'_j$ equals the expected revenue of following the sequence σ' for the corresponding ERM instance with a single machine and revenues $(r'_j)_{j\in N}$ instead of $(r_j)_{j\in N}$. Since $C(S)$ does not depend on σ', Theorem 1 yields that, for fixed S, the expected revenue $\mathrm{ER}(S\uplus\{\sigma'\})$ is maximized by σ' if and only if the jobs $j\in N$ are scheduled in non-increasing order of the ratio

$$\frac{p_jr'_j}{1-p_j}=\frac{p_jr_j}{1-p_j}\prod_{\sigma\in S}\left(1-P_j(\sigma)\right)=Z_j(S),$$

which establishes the result. □

Based on the above result, we propose the *modified Z-rule heuristic* (MZRH), summarized in Algorithm 2. The modified Z-rule heuristic builds a feasible solution such that, in each iteration, the next machine schedules the jobs in non-increasing order of their modified Z-ratio, given the partial solution formed by the previous machines' sequences. Notice that, given the values $P_j(S-\{\sigma\})$ and σ, computing $P_j(S)$ can be done in constant time (as $P_j(S)=P_j(S-\{\sigma\})+P_j(\sigma)-P_j(S-\{\sigma\})P_j(\sigma)$). Since there are m iterations and scheduling the jobs in non-increasing order of their modified Z-ratio takes time $O(n\log(n))$, the modified Z-rule heuristic runs in time $O(mn\log(n))$.

In the remainder of this section, an approximation result for the Modified Z-rule heuristic is presented. More precisely, we show that this heuristic produces a solution whose value is at least $1-(\frac{m-1}{m})^m$ times the optimal value. Such a result follows from an approximation study of Nemhauser et al. (1978) establishing an approximation bound of a "greedy" heuristic for the maximization of submodular set functions.

Definition 1. Given a finite set M, a set function $f: 2^M \to \mathbb{R}$ is submodular if for every $\rho \in M$ and $S \subseteq T \subseteq M$ it holds that $f(S \cup \{\rho\}) - f(S) \geq f(T \cup \{\rho\}) - f(T)$.

In words, for a submodular function f, the marginal benefit of including an additional element ρ is non-increasing in the set of already included elements.

Given a finite set M, a submodular function $f: 2^M \to \mathbb{R}$, and an integer $m \geq 1$, consider the problem $\max\{f(S): |S| \leq m, S \subseteq M\}$ and the greedy algorithm that, starting from the empty set $S = \emptyset$, at each step adds to S an element $\sigma \in M \setminus S$ maximizing $f(S \cup \{\sigma\}) - f(S)$. Nemhauser et al. (1978) proved that such a greedy algorithm is $1-(\frac{m-1}{m})^m$ approximate.

Below, we use the result of Nemhauser et al. (1978) to argue that the modified Z-rule heuristic has the same approximation guarantee. To do so, we first show that the expected revenue is submodular in the included sequences and, next, argue that the modified Z-rule heuristic coincides with the greedy algorithm described by Nemhauser et al. (1978). Since we represent a solution by a multiset instead of a set, however, we first need to introduce some additional notation.

Let $\mathcal{P}$ denote the set of all permutations (i.e., sequences) of the jobs in N, and let $M = \mathcal{P} \times \{1, \dots, m\}$ be the set containing m copies of each permutation. Hence, M contains $n! \cdot m$ job permutations. For an arbitrary subset $S \subseteq M$, denote by $\bar{S}$ the multiset obtained from S by including each permutation $\sigma \in \mathcal{P}$ with the number of times that it appears in S. Additionally, define the set function $f: 2^M \to \mathbb{R}$ with $f(S) = \mathrm{ER}(\bar{S})$ for every $S \subseteq M$. It then follows that there is a one-to-one correspondence between subsets $S \subseteq M$ with $|S| = m$ and solutions $\bar{S}$ of our ERM problem, modulo permuting machines. The correspondence is in such a way that, for such a subset S, $f(S)$ equals the expected revenue of $\bar{S}$. The following lemma shows that f is submodular, i.e., that the additional expected revenue thanks to scheduling an additional machine according to a given permutation is non-increasing in the multiset of sequences scheduled on other machines.

Lemma 6. *The set function $f: 2^M \to \mathbb{R}$ with $f(S) = \mathrm{ER}(\bar{S})$ for every $S \subseteq M$ is submodular.*

Proof. Consider arbitrary $S \subseteq T \subseteq M$ and $(\rho, k) \in M \setminus T$. From Eq. (1),

$$f(S \cup \{(\rho,k)\}) = \mathrm{ER}(\bar{S} \uplus \{\rho\}) = \sum_{j \in N} \left[1 - \left(1 - P_j(\rho)\right) \prod_{\sigma \in \bar{S}} \left(1 - P_j(\sigma)\right)\right] r_j$$
$$= \mathrm{ER}(\bar{S}) + P_j(\rho) \sum_{j \in N} \left[\prod_{\sigma \in \bar{S}} \left(1 - P_j(\sigma)\right)\right] r_j$$

and similarly

$$f(T \cup \{(\rho,k)\}) = \mathrm{ER}(\bar{T}) + P_j(\rho) \sum_{j \in N} \left[\prod_{\sigma \in \bar{T}} \left(1 - P_j(\sigma)\right)\right] r_j.$$

Since $S \subseteq T$ and $0 \leq 1 - P_j(\sigma) \leq 1$ for every $\sigma \in M$ and $j \in N$, it follows that

$$f(S \cup \{(\rho,k)\}) - f(S) = \mathrm{ER}(\bar{S} \uplus \{\rho\}) - \mathrm{ER}(\bar{S})$$
$$= P_j(\rho) \sum_{j \in N} \left[\prod_{\sigma \in \bar{S}} \left(1 - P_j(\sigma)\right)\right] r_j$$
$$\geq P_j(\rho) \sum_{j \in N} \left[\prod_{\sigma \in \bar{T}} \left(1 - P_j(\sigma)\right)\right] r_j$$
$$= \mathrm{ER}(\bar{T} \uplus \{\rho\}) - \mathrm{ER}(\bar{T}) = f(T \cup \{(\rho,k)\}) - f(T).$$

By Definition 1, this shows that f is submodular. □

Now observe that, with the definition of M and f as described above, our ERM problem corresponds exactly to the problem $\max\{f(S): |S| \leq m, S \subseteq M\}$. Moreover, it follows from Theorem 6 that, starting from the empty set $S = \emptyset$, the modified Z-rule heuristic adds at each iteration $t = 1, \dots, m$ a permutation $\sigma \in \mathcal{P}$ such that

$$f(S \cup \{(\sigma,t)\}) - f(S) = \max_{(\rho,k) \in M \setminus S} \{f(S \cup \{(\rho,k)\}) - f(S)\}.$$

As such, our modified Z-rule heuristic is identical to the greedy algorithm described by Nemhauser et al. (1978) applied to f. Moreover, by Lemma 6, the function f is submodular. If we let S_H be the solution produced by the Modified Z-rule heuristic, and S^* an optimal solution, we thus obtain the following result.

Theorem 7 (*Nemhauser et al., 1978*)**.** *For every $m \in \mathbb{N}_0$ it holds that*

$$\frac{\mathrm{ER}(S_H)}{\mathrm{ER}(S^*)} \geq 1 - \left(\frac{m-1}{m}\right)^m \geq 1 - \frac{1}{e}.$$

6. Mutual-best-reply heuristic

In this section we introduce a third heuristic algorithm, called the *mutual-best-reply heuristic* (MBRH) and summarized in Algorithm 3. Here, for a solution S and a sequence $\sigma \in S$, we denote by $S - \{\sigma\}$ the multiset obtained by decreasing the multiplicity of σ in S by one. We also define

$$W(S) = \{\sigma \in S : \exists\, i, j \in N \text{ with } \sigma(i) < \sigma(j) \text{ and } Z_i(S - \{\sigma\}) < Z_j(S - \{\sigma\})\}$$

as the set of sequences $\sigma \in S$ that do not schedule the jobs $j \in N$ in non-increasing order of their modified Z-ratios within the partial solution $S - \{\sigma\}$. The mutual-best-reply heuristic then iteratively checks whether there is a machine whose sequence disagrees with the modified Z-rule given the partial solution formed by all other machines. If so, then the machine is rescheduled according to the modified Z-rule and we proceed to the next iteration. Hence, when the procedure terminates, it follows from Theorem 6 that every machine sequences the jobs optimally if we consider the other machines' sequences to be fixed. That is, the sequences form a mutual best reply to each other.

It follows from Theorem 6 that whenever a sequence disagrees with the modified Z-rule, then rescheduling leads to a strict increase in the expected revenue. Eq. (21) in the proof of Theorem 1 implies that this increase is lower bounded by

$$\gamma = \min_{i,j \in N\,:\, Z_i > Z_j} \left\{(1 - p_i)(1 - p_j)(Z_i - Z_j) \prod_{k \in N \setminus \{i,j\}} p_k\right\}.$$

Since the expected revenue of an arbitrary solution lies in the interval $\left[0, \sum_{j \in N} r_j\right]$, the mutual-best-reply heuristic thus terminates in at most $\sum_{j \in N} r_j / \gamma$ iterations. This is finite, but not polynomial (or even pseudo-polynomial) in the input size. Establishing the exact computational complexity of the mutual-best-reply heuristic still forms an open problem. As for the approximation ratio of MBRH, notice that it cannot

Algorithm 3: Mutual-best-reply heuristic (MBRH).

input: An ERM instance $(n, m, (r_i, p_i)_{i \in N})$ and an initial solution $S = \{\sigma_1, \dots, \sigma_m\}$

while there exists a $\sigma \in W(S)$, **do**
  let $S \leftarrow S - \{\sigma\}$;
  let σ' be a sequence that schedules the jobs $j \in N$ in non-increasing order of $Z_j(S)$;
  let $S \leftarrow S \uplus \{\sigma'\}$;

return S;

Algorithm 4: Tabu Search for ERM.

input: an ERM instance $(n, m, (r_i, p_i)_{i\in N})$ and an initial solution $\{\sigma_1, \dots, \sigma_m\}$

initialize $S_{\text{best}} \leftarrow \{\sigma_1, \dots, \sigma_m\}$ and $t \leftarrow 0$;
for $k = 1, \dots, m$ **do**
  for $i, j \in N$ with $i < j$ **do**
    $\tau(i, j, k) \leftarrow 0$; // Initialize tabu list
while $t < T$, **do**
  let $B \leftarrow 0$;
  for $k = 1, \dots, m$ **do** // Loop over machines
    for $i, j \in N$ with $i < j$ and $\tau(i, j, k) \geq t$ **do** // Loop over non-tabu swaps
      swap i and j in σ_k to obtain a new sequence σ'_k;
      initialize $S' \leftarrow \{\sigma'\}$ and $R \leftarrow \{1, \dots, m\} \setminus \{k\}$;
      while $R \neq \emptyset$ **do** // Schedule remaining machines
        pick arbitrary $\ell \in R$ and let $R \leftarrow R \setminus \{\ell\}$;
        let σ'_ℓ be a sequence with jobs $u \in N$ in non-increasing order of $Z_u(S')$;
        let $S' \leftarrow S' \uplus \{\sigma'_\ell\}$;
      if $\text{ER}(S') > B$ **then** // Store best non-tabu neighbor
        $B \leftarrow \text{ER}(S')$;
        $(\hat{i}, \hat{j}, \hat{k}) \leftarrow (i, j, k)$;
        $(\hat{\sigma}_1, \dots, \hat{\sigma}_m) \leftarrow (\sigma'_1, \dots, \sigma'_m)$;
  if $B > \text{ER}(S_{\text{best}})$ **then** // Store overall best solution
    $S_{\text{best}} \leftarrow \{\hat{\sigma}_1, \dots, \hat{\sigma}_m\}$;
  else if $B < \text{ER}(\{\sigma_1, \dots, \sigma_m\})$ **then** // Make non-improving move tabu
    $\tau(\hat{i}, \hat{j}, \hat{k}) \leftarrow t + \bar{\tau}$;
  let $(\sigma_1, \dots, \sigma_m) \leftarrow (\hat{\sigma}_1, \dots, \hat{\sigma}_m)$;
  let $t \leftarrow t + 1$;
return S_{best};

exceed $1 - 1/e$, when the solution given by MZRH is used to initialize MBRH.

7. A tabu-search heuristic

In this section, a Tabu Search (TS) algorithm for ERM is presented. The overall structure is summarized in Algorithm 4. Starting from an initial solution $\{\sigma_1, \dots, \sigma_m\}$, in each iteration t of the TS scheme we select the best non-tabu neighbor solution $\{\hat{\sigma}_1, \dots, \hat{\sigma}_m\}$. The solution neighborhood is constructed according to the method explained below. If this best neighbor improves the best known solution S_{best}, we update S_{best}. If the best neighbor is worse than the solution with which we started the iteration, i.e., no improving solution is found in the neighborhood, then the move that led to the best neighbor is added to the tabu list τ. The size of the tabu list, denoted as $\bar{\tau}$, specifies the maximum number of moves it can hold. When the tabu list reaches its maximum size, new moves are added by removing the oldest entry, adhering to a First-In, First-Out (FIFO) policy.

Finally, we proceed to the next iteration with $\{\hat{\sigma}_1, \dots, \hat{\sigma}_m\}$ as our new current solution. The algorithm terminates when the maximum number of iterations T is reached.

Given a tuple $(\sigma_1, \dots, \sigma_m)$, we consider for every two jobs $i, j \in N$, with $i < j$, and for every machine $k \in \{1, \dots, m\}$ a *move* (i, j, k) that generates a *neighbor* $(\sigma'_1, \dots, \sigma'_m)$ as follows. First, we swap jobs i and j in sequence σ_k, i.e., the sequence followed on machine k, to obtain a new sequence σ'_k. Next, we run through the remaining machines in an arbitrary order and, at each step, the next machine ℓ is scheduled according to a sequence σ'_ℓ as specified by the modified Z-rule. The definition of a move (i, j, k) not only keeps track of which two jobs are interchanged, but also of the machine on which this is done. Indeed, to prevent cycling, we only need to prevent undoing a non-improving move on the relevant machine. For this reason, the tabu list is a list of the (at most) $\bar{\tau}$ most recent moves, i.e., triples (i, j, k), that could lead to cycling. As such, for the purpose of defining the neighborhood and the tabu list, the orders of the schedules on the different machines matter. We therefore represent a solution as an ordered tuple $(\sigma_1, \dots, \sigma_m)$ rather than as an unordered multiset $\{\sigma_1, \dots, \sigma_m\}$.

Observe that, rather than running through the remaining machines in an arbitrary order after having performed a swap on a given machine, we could also make this order a part of the move. This order is relevant for future iterations because the machines differ with respect to which pairwise interchanges are allowed by the tabu list. By considering all possible orders, however, the size of the neighborhood increases substantially. Preliminary computational experiments have shown that the increase in solution quality thanks to this larger neighborhood does not outweigh the additional time needed to explore it. Therefore, we chose to rely on a random order as described in Algorithm 4.

For the specific case of two machines, the tabu search described in Algorithm 4 is equivalent to the one developed by Agnetis et al. (2022). Our current approach therefore directly generalizes that algorithm to an arbitrary number of machines.

8. Computational experiments

In this section, we assess the quality of the introduced upper bound and heuristics by means of computational experiments. To do so, we use the same instances as in Agnetis et al. (2022). Section 8.1 describes these instances and provides the implementation details of our experiments. Section 8.2 focuses on the special case with two machines, i.e., ERM2, so that we can directly compare the performance of our methods with the ones proposed in Agnetis et al. (2022). Next, in Section 8.3, we focus on ERM3 to assess the quality of our tabu search algorithm. Section 8.4, finally, demonstrates how the gap between our upper bound and heuristics scales in the number of machines.

8.1. Instance description and implementation details

Agnetis et al. (2022) generated 20 instances for each combination of the number of jobs $n \in \{10, 20, 30, 40, 50\}$ and intervals $I_p \in \{[0.9, 1], [0.5, 1], [0.1, 1]\}$ for the success probability. For each job $i = 1, \dots, n$ in a given instance, a success probability p_i was drawn uniformly at random from the corresponding interval I_p, while the revenue r_i was drawn from an integer uniform distribution on the interval $[10, 100]$. This led to $20 \times 5 \times 3 = 300$ instances in total.

Then, for ERM3 only, to compare the value of the sequential upper bound with that of the solution provided by Tabu Search and other heuristics, as well as with the optimal value, we generated 10 further small instances for $n \in \{5, 6, 7\}$ across each of the three probability intervals and solved them to optimality by complete enumeration. It is important to note that, to this aim, we only need to evaluate all possible sequences for *two* machines. In fact, for each pair of sequences on two machines, the Z-rule is then applied to optimally schedule the other machine. However, we observed that for $n > 7$, the computational effort becomes prohibitive.

The tabu search has been run as a Python 3.7 program on a single 1.7 GHz CPU core with 8 GB RAM.

The experiments were conducted with $T = 100$ iterations of the Tabu Search, as preliminary experiments indicated that a higher number of iterations would not significantly improve solution quality. The size of the tabu list $\bar{\tau}$ was set to 20% of the number of jobs for each instance.

Table 2
Average percentage gap and cpu time in seconds for ERM2 instances.

		ZRH	MZRH	MBRH	Tabu Search		3AP bound	
n	I_p	% gap	% gap	% gap	% gap	cpu	% gap	cpu
10	[0.1, 1]	8.21	6.96	6.53	6.27	0.24	−8.10	0.22
10	[0.5, 1]	8.66	6.94	5.89	5.52	0.25	−7.52	0.94
10	[0.9, 1]	1.38	0.88	0.73	0.65	0.24	−0.78	0.97
20	[0.1, 1]	10.52	8.41	7.29	6.72	3.76	−12.72	1.05
20	[0.5, 1]	10.85	8.60	7.08	6.37	3.31	−19.83	1.02
20	[0.9, 1]	3.37	2.35	1.88	1.60	3.64	−2.64	1.07
30	[0.1, 1]	10.90	8.51	7.40	6.54	11.80	−20.25	1.60
30	[0.5, 1]	11.96	9.40	7.56	6.77	11.47	−27.10	1.52
30	[0.9, 1]	5.82	3.81	2.98	2.47	10.84	−5.84	1.51
40	[0.1, 1]	10.51	8.65	7.11	6.48	27.00	−28.06	4.44
40	[0.5, 1]	11.63	9.31	7.39	6.46	26.46	−31.26	3.98
40	[0.9, 1]	7.53	5.40	4.07	3.43	25.81	−7.95	4.81
50	[0.1, 1]	11.57	9.03	7.29	6.46	55.92	−27.60	13.95
50	[0.5, 1]	11.68	9.18	7.05	6.27	55.08	−36.39	9.78
50	[0.9, 1]	9.10	6.08	4.75	3.84	54.68	−12.56	10.66

All heuristics and the upper bound introduced in this paper have been run on a single 1.7 GHz CPU core equipped with 8 GB of RAM. The tabu search was implemented as a Python 3.7 program, whereas the other methods were implemented using the C++ programming language and compiled with Microsoft Visual C++ 14.0. The computational results for the methods introduced in Agnetis et al. (2022) were taken directly from that paper, so we refer to their description for the corresponding implementation details.

As a starting solution for MBRH and tabu search, we used the solution obtained by MZRH and MBRH, respectively.

8.2. Performance for ERM2

For the specific case of ERM2, Agnetis et al. (2022) proposed a quadratic integer program, a tabu search heuristic, and an upper bound based on a three-dimensional assignment problem (3AP). Although the quadratic integer program cannot be solved exactly within a reasonable computation time, it does provide a feasible solution and a valid upper bound. As such, it can be used as a heuristic whose performance can be evaluated by means of the obtained upper bound. Among these methods, the tabu search and the 3AP-based upper bound turned out to perform best. Therefore, we only compare our new upper bound and heuristics to these latter two methods.

Table 2 displays the results of this experiment on ERM2-instances. For a given value V (be it a lower bound obtained through a heuristic or the upper bound provided by 3AP), the percentage gap is computed relative to the sequential upper bound UB_s, presented in Section 3, as:

$$\frac{UB_s - V}{UB_s} \times 100\%.$$

In addition to the percentage gap, the table also reports on the average CPU time in seconds for the tabu search and the 3AP bound. All other heuristics needed less than 0.001 s to solve any of the instances. The averages are taken over all twenty instances for the corresponding value of n and interval I_p as indicated by the table's rows.

Table 2 clearly shows that the more advanced heuristics also result in better solutions. When comparing the corresponding CPU times, however, we observe that for larger instances the tabu search needs a relatively large amount of time to improve upon the solution obtained by MBRH.

The performance of all heuristics follows the same overall trend: with $I_p = [0.9, 1]$, the gap clearly increases in the number of jobs n, whereas it is much more stable for $I_p = [0.5, 1]$ or $I_p = [0.1, 1]$. Moreover, when comparing the different intervals of the success probability, we see that the gap increases between $I_p = [0.9, 1]$ and $I_p = [0.5, 1]$ and, perhaps somewhat surprisingly, the gap stabilizes between $I_p = [0.5, 1]$ and $I_p = [0.1, 1]$. Since we do not know the optimal solution value, however, it is not clear which part of the gap is caused by the heuristic and the sequential upper bound, respectively.

Table 3
Gaps of heuristics and upper bound with optimal solutions on small instances.

		ZRH	MZRH	MBRH	Tabu Search	Sequential UB
n	I_p	% gap	% gap	% gap	% gap	% gap
5	[0.1, 1]	−6.916	−0.819	−0.310	−0.092	9.260
5	[0.5, 1]	−12.309	−0.556	−0.326	−0.096	4.761
5	[0.9, 1]	−0.096	−0.008	−0.003	−0.001	0.044
6	[0.1, 1]	−7.952	−0.990	−0.396	−0.160	10.144
6	[0.5, 1]	−5.712	−0.872	−0.475	−0.194	6.114
6	[0.9, 1]	−0.146	−0.014	−0.006	−0.004	0.063
7	[0.1, 1]	−8.723	−0.887	−0.400	−0.136	10.528
7	[0.5, 1]	−6.683	−1.151	−0.520	−0.275	7.146
7	[0.9, 1]	−0.212	−0.022	−0.008	−0.006	0.092

With respect to the quality of the bound, we find that the sequential upper bound clearly dominates the one provided by 3AP, both in the quality of the bound as in the computation time. Indeed, for all considered instances, it took less than 0.001 s to compute the sequential upper bound, and its value was strictly smaller than the one provided by 3AP for all but one of the instances. As Table 2 shows, the average percentage gap of 3AP is always negative, and the sequential upper bound outperforms the 3AP bound especially when n is larger and $I_p = [0.5, 1]$.

8.3. Performance for ERM3

As discussed in Section 8.1, for a set of small instances involving 5, 6, and 7 jobs, the values of the sequential upper bound, Tabu Search solution and other heuristic solutions have been compared with the optimal solution value, obtained through complete enumeration. Table 3 presents the results, showing the average percentage gap relative to the optimal solution for each heuristic algorithm and for the sequential upper bound. Each row reports the average value across the 10 instances in the set. Note that the gap is negative for the heuristics, being their solution values smaller than or equal to the optimal ones, and positive for the upper bound since its solution values are higher. The results indicate that Tabu Search outperforms all other heuristics, consistently finding optimal or near-optimal solutions, with an overall absolute average gap of just 0.1%. The Modified Z-rule Heuristic (MZRH) and particularly the Mutual Best Reply Heuristic (MBRH) also perform well, with absolute average gaps of 0.27% for MBRH and 0.59% for MZRH. In contrast, the Z-rule Heuristic shows weaker performance, with an absolute average gap of 5.42% from the optimal solutions. Overall, the gap generally increases with the number of jobs, with the smallest gap occurring in the probability range $[0.9, 1]$ and the largest gap in the range $[0.5, 1]$. For the sequential upper bound, the overall average gap from the optimal solutions is 5.35%, but we note that it significantly deteriorates moving from $I_p = [0.9, 1]$ (0.07%) to $I_p = [0.1, 1]$ (9.98%). This deterioration is notably more pronounced than that observed for the heuristics. It is also noteworthy that the behavior of the sequential upper bound differs from that of the heuristics also in the fact that the gap always increases as the probability interval widens.

Table 4 provides the results for three-machine problems. (The 3AP-based upper bound by Agnetis et al. (2022) does not appear in the table since it only applies to ERM2.)

The computational results for ERM3 display a similar pattern as for ERM2. When comparing Tables 2 and 4, however, we see that the average gaps decrease for $I_p = [0.9, 1]$ and $n \leq 30$, while they increase for most other settings. One possible explanation for this is that when $I_p = [0.9, 1]$ and n is small, then finding a schedule in which most jobs are likely to be carried out is easier with $m = 3$ than with $m = 2$. As the

Table 4
Average percentage gap and cpu time in seconds for ERM3 instances.

		ZRH	MZRH	MBRH	Tabu Search	
n	I_p	% gap	% gap	% gap	% gap	cpu
10	[0.1, 1]	13.52	10.12	9.43	9.24	1.40
10	[0.5, 1]	12.68	8.29	7.51	7.26	1.49
10	[0.9, 1]	0.80	0.37	0.33	0.32	1.58
20	[0.1, 1]	17.00	11.67	10.53	10.38	12.88
20	[0.5, 1]	17.53	11.60	9.93	9.70	13.98
20	[0.9, 1]	2.81	1.43	1.19	1.18	15.62
30	[0.1, 1]	17.61	11.90	10.43	10.26	48.99
30	[0.5, 1]	19.29	12.89	10.86	10.61	55.83
30	[0.9, 1]	6.17	3.09	2.50	2.49	65.93
40	[0.1, 1]	17.12	11.82	10.21	10.04	128.16
40	[0.5, 1]	18.80	12.47	10.30	10.08	139.66
40	[0.9, 1]	9.03	4.92	3.94	3.94	174.16
50	[0.1, 1]	18.68	12.31	10.55	10.30	263.77
50	[0.5, 1]	19.06	12.49	10.26	10.05	301.88
50	[0.9, 1]	11.84	6.35	5.04	5.02	394.70

number of jobs increases or the range of success probabilities becomes wider, this effect dissipates and the gap increases, instead of decreasing, with the number of machines. Section 8.4 will demonstrate this effect even more clearly.

Especially for larger instances, we observe that the tabu search needs a relatively large amount of time to yield only a limited improvement over MBRH. It seems that, compared to the case of ERM2, the enlarged neighborhood harms the performance of the tabu search algorithm. Since this effect would only get worse as the number of machines increases, we did not test the tabu search for more than three machines.

8.4. Performance for ERMm

Fig. 1 provides an insight into how the gap between our heuristics (other than tabu search) and the sequential upper bound scales in m. The figure displays a similar pattern to what we observed in Section 8.3 when comparing Tables 2 and 4. For each fixed n, the gaps initially tend to increase in m, but, as m becomes larger, this growth slows down and, for most settings, eventually becomes negative. In fact, as the number of machines becomes sufficiently large, most jobs are likely to be carried out, leading to a smaller gap.

This empirical finding shows that the average percentage gap does not increase monotonically in the number of machines. This seems to conflict with the fact that, as derived in Section 4, the worst-case performance does deteriorate monotonically in m. However, in establishing the asymptotic tightness of this worst-case guarantee we considered a family of instances in which n tends to infinity. As such, our computational experiment suggests that it might be possible to derive a better performance guarantee by parameterizing it on the number of jobs n.

Comparing the average percentage gaps for ZRH and MZRH with the related worst-case performance guarantees (i.e., H_m/m and $1-(\frac{m-1}{m})^m$, respectively), our computational results suggest that the average performance of these heuristics is considerably better than the worst-case performance. Moreover, the empirical performance can be further improved by using the MBRH heuristic, even if such an improvement is very small (the largest gap decrease attained by MBRH with respect to MZRH is by 1.3% in an instance with $m = 3$ and $n = 50$). We conjecture that such a small improvement is due to the fact that the quality of the solution returned by MZRH is already very high, though we are not able to certify it with the current upper bounds.

9. Conclusions

In this paper, we establish new results for the problem of maximizing the expected revenue of unreliable jobs on parallel machines when jobs can be replicated, as introduced by Agnetis et al. (2022). More specifically, we have proposed a novel upper bound (the sequential upper bound) and four heuristic approaches, namely the Z-rule heuristic, the modified Z-rule heuristic, the mutual-best-reply heuristic, and a tabu search.

Our main theoretical contribution is to show that there is a simple $(1-1/e)$-approximation algorithm for ERM, the modified Z-rule heuristic. We also show an asymptotically tight bound of $\Theta(m/\log m)$ on the approximation guarantee of the even simpler Z-rule heuristic. In addition, we show that the sequential upper bound overestimates the optimal expected revenue by a factor of at most 2.

Based on extensive computational experiments, we also find that our proposed methods display a good empirical performance. In particular, the sequential upper bound performs significantly better than the theoretical bound of 2 as well as the bound based on the three-dimensional assignment problem proposed in Agnetis et al. (2022) for ERM2. With respect to our heuristics, especially the modified Z-rule heuristic and the mutual-best-reply heuristic result in high-quality solutions in very limited computation time. On the other hand, the tabu search algorithm only seems practical for instances with two or three machines.

A number of open issues may be addressed by future research.

- As we were not able to provide tight examples, we cannot exclude that the actual worst-case performance ratio of MZRH can be improved. Concerning MBRH, it would be interesting to investigate whether it also leads to a better approximation guarantee than $1-1/e$.
- We cannot exclude either that our sequential upper bound overestimates the optimum value by less than a factor of 2. We did show that a different approach would be needed. It would be interesting to investigate this sequentiality gap further.
- Another direction for future research is to develop exact solution methods. Indeed, the proposed sequential upper bound and the fast MBRH can be used to devise exact implicit enumeration approaches (e.g., based on branch&bound and branch&cut schemes). In this context, the study of dominance rules is also important.
- In the tabu-search algorithm, for $m > 3$ the neighborhood becomes too large to be effectively explored. This suggests that further research can be devoted to address the definition of new neighborhoods that scale well in the number of machines, and the study of fast methods for neighborhood exploration.
- Another direction of further research deals with modeling issues. In our model, success probabilities are job-related and not machine-related. In fact, when failures are machine-related, the success probability may depend on the time when a job is started, which is closer to the view of Benoit et al. (2013). In the linear risk model in Benoit et al. (2013), a single computer is used to process the jobs through a time window of length T, and the objective is to maximize the amount of accomplished work. The computer is available at the beginning of the time window, but it will be withdrawn sometime during such a time window. Without any further information, the probability of the resource being still available at time t is $1-(t/T)$ (this is why it is a linear risk model). An adaptation of ERM to this risk model implies that each job j has a processing time q_j, which coincides with its revenue r_j. Letting $Q = \sum q_j$, and supposing that $T \geq Q$, when sequencing n jobs on a single machine, the expected revenue is given by

$$\sum_{j=1}^{n}\left(1-\frac{\sum_{i=1}^{j} q_i}{T}\right)q_j.$$

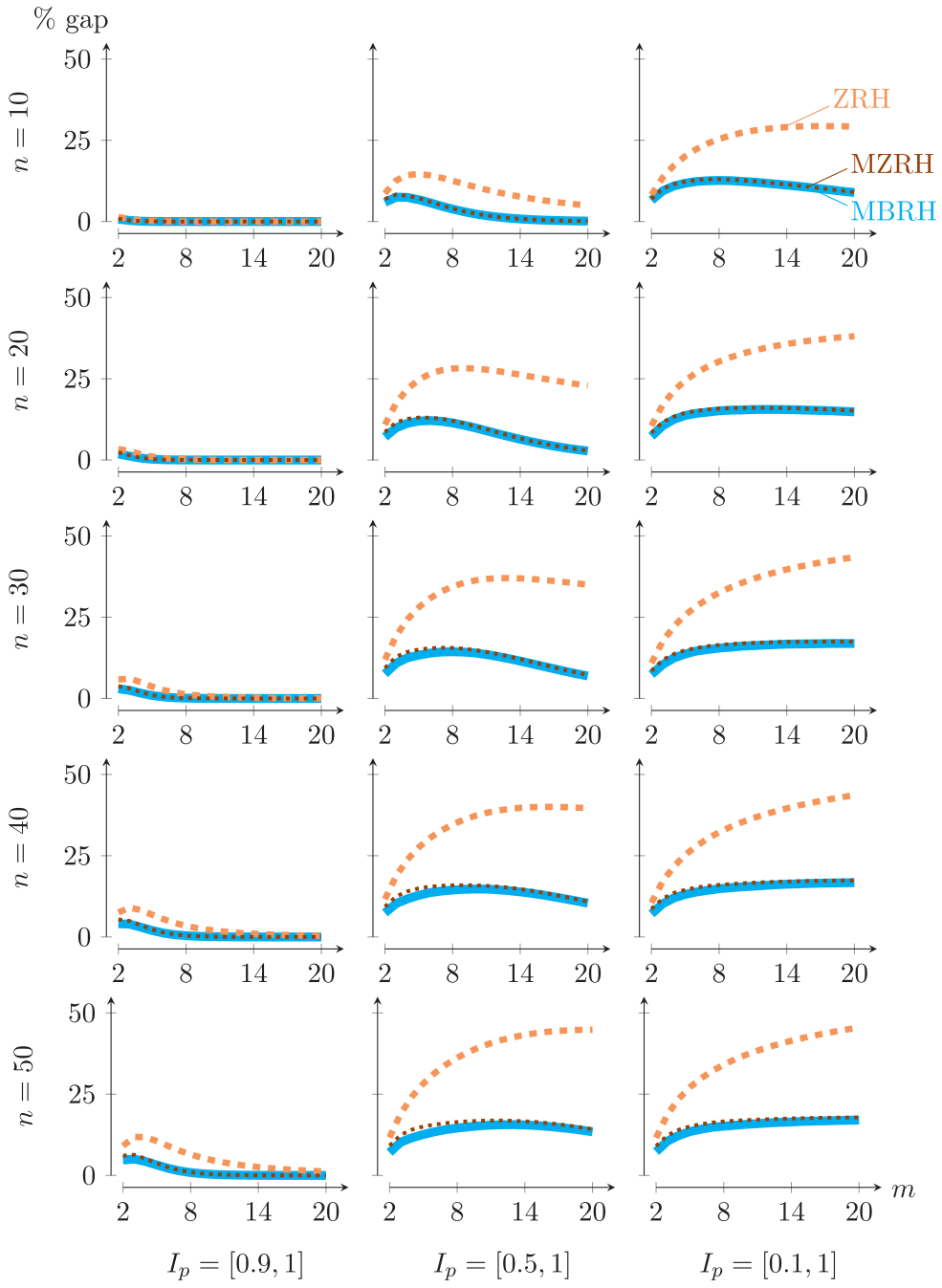

Fig. 1. Average percentage gap for ZRH, MZRH, and MBRH as a function of the number of machines m. The graph is repeated for different values of $n \in \{10, 20, \ldots, 50\}$ and intervals for the success probabilities $I_p \in \{[0.9, 1], [0.5, 1], [0.1, 1]\}$. Gaps are computed relative to the sequential upper bound UB_s as $(UB_s - V)/UB_s \times 100\%$ for the corresponding value V.

Since, for any n positive integers $q_1, q_2, \ldots, q_n$, one has

$$\sum_{i \neq j} q_i q_j = \frac{Q^2 - \sum_{j=1}^{n} q_j^2}{2},$$

it is easy to see that the expected revenue does not depend on the sequencing of the jobs and is given by

$$Q - \frac{1}{2T}(Q^2 + \sum_{j=1}^{n} q_j^2).$$

The analysis of the case with $m > 1$ machines (with or without job replication), possibly having different time windows, is an interesting topic for future research.

- We stated in the introduction that the values r_j may be interpreted as utilities rather than monetary values. However, risk neutrality can be an acceptable assumption if the range of values that success probabilities p_j may assume is limited (say, $p_{\max}/p_{\min} < \theta$ for some small θ). This might in turn yield stricter approximation bounds (even in the problem without job replication), which is a relevant topic for future research.
- There are also open questions with respect to the computational complexity of ERM. Agnetis et al. (2022) have established that ERM2 is NP-hard, but it is unclear whether it is weakly or strongly NP-hard. It is also still unknown how the complexity scales in the number of machines or whether special cases of the problem (e.g., with all jobs having an equal revenue) are easier to solve.

CRediT authorship contribution statement

Alessandro Agnetis: Writing – original draft. **Mario Benini:** Writing – original draft. **Paolo Detti:** Writing – original draft. **Ben Hermans:** Writing – original draft. **Marco Pranzo:** Writing – original draft. **Kevin Schewior:** Writing – original draft.

Acknowledgments

Alessandro Agnetis, Mario Benini, Paolo Detti and Marco Pranzo were supported by the Italian Ministry of University and Research, grant PNRR - Next Generation EU - THE - Spoke 10 - CUP: B63C22000 680007.

Ben Hermans was funded by a postdoctoral fellowship (grant number 12ZZI21N) of the Research Foundation – Flanders.

Kevin Schewior was supported by the Independent Research Fund Denmark, Natural Sciences (grant DFF-0135-00018B).

Appendix A. Table of notations

Notation	Meaning
$\mathbb{N}$	$\{1,2,\dots\}$
$\mathbb{N}_0$	$\{0,1,2,\dots\}$
N	set of jobs
n	number of jobs
m	number of machines
p_j	success probability of job j
sequence	bijection $\sigma : N \to \{1,\dots,n\}$
$P_j(\sigma)$	Probability that j is successful under sequence σ
solution for ERM	multiset S of m sequences
ER(S)	expected revenue of solution S
Z_j	Z-ratio of job j

Appendix B. Proofs from Section 2

Proof of Theorem 1. Consider a sequence $\sigma^\star$ and, for arbitrary jobs $i,j \in N$ with $\sigma^\star(j) = \sigma^\star(i)+1$, let $\sigma^\star_{ij}$ be the sequence obtained from $\sigma^\star$ by interchanging jobs i and j. Defining Q such that $P_i(\sigma^\star) = p_i Q$, it then holds that $P_j(\sigma^\star) = P_i(\sigma^\star_{ij}) = p_i p_j Q$ and $P_j(\sigma^\star_{ij}) = p_j Q$. For all other jobs $k \in N \setminus \{i,j\}$, in turn, we have that $P_k(\sigma^\star) = P_k(\sigma^\star_{ij})$. Eq. (2) then yields that

$$\begin{aligned}\mathrm{ER}(\sigma^\star) - \mathrm{ER}(\sigma^\star_{ij}) &= P_i(\sigma^\star)r_i + P_j(\sigma^\star)r_j - P_i(\sigma^\star_{ij})r_i - P_j(\sigma^\star_{ij})r_j \\ &= p_i Q r_i + p_i p_j Q r_j - p_i p_j Q r_i - p_j Q r_j \\ &= (1-p_j)p_i Q r_i - (1-p_i)p_j Q r_j \\ &= (1-p_i)(1-p_j)(Z_i - Z_j)Q, \end{aligned} \tag{21}$$

which is non-negative if and only if

$$Z_i = \frac{p_i r_i}{1-p_i} \geq \frac{p_j r_j}{1-p_j}.$$

Iterating this argument shows that $\sigma^\star$ is optimal if and only if $\sigma^\star(i) < \sigma^\star(j)$ for all $i,j \in N$. □

Appendix C. Proofs from Section 3

Proof of Lemma 1. We show the result by induction on i. For $i = n$, we have $f(n,1) = p_n r_n \leq Z_n$ since $0 < p_n < 1$ and $r_n > 0$. Now consider arbitrary $i < n$ and assume that $f(i+1,1) \leq Z_{i+1}$. Eq. (3) and our assumption that $Z_i \geq Z_{i+1}$ then yield

$$\begin{aligned} f(i,1) = p_i\left[r_i + f(i+1,1)\right] &\leq p_i\left(r_i + Z_{i+1}\right) \leq p_i\left(r_i + Z_i\right) \\ &= p_i\left(r_i + \frac{p_i r_i}{1-p_i}\right) = \frac{p_i r_i}{1-p_i} = Z_i, \end{aligned}$$

which completes the proof. □

Proof of Lemma 2. We show the result using induction on k and i. For $k = 1$ the result trivially holds. Now suppose that $k \geq 2$ and, as the first induction hypothesis, assume that

$$\frac{f(i,k-1)}{k-1} \leq \frac{f(i,\ell)}{\ell} \tag{22}$$

for all $i \in N$ and $\ell = 1,\dots,k-1$. Eq. (3) then yields that

$$\begin{aligned}(k-1)f(n,k) &= (k-1)\sum_{\ell=0}^{k-1}(1-p_n)^\ell p_n r_n = (k-1)f(n,k-1) \\ &\quad + (k-1)(1-p_n)^{k-1}p_n r_n \\ &\leq (k-1)f(n,k-1) + \sum_{\ell=0}^{k-2}(1-p_n)^\ell p_n r_n = k f(n,k-1).\end{aligned}$$

The first induction hypothesis, i.e., Inequality (22), then implies that

$$\frac{f(n,k)}{k} \leq \frac{f(n,k-1)}{k-1} \leq \frac{f(n,\ell)}{\ell}$$

for all $\ell = 1,\dots,k-1$. Since we also trivially have $f(n,k)/k \leq f(n,\ell)/\ell$ for $\ell = k$, we obtain that the result holds for $i = n$.

Now consider arbitrary $i \in N \setminus \{n\}$ and, as the second induction hypothesis, assume that

$$\frac{f(i+1,k)}{k} \leq \frac{f(i+1,\ell)}{\ell} \tag{23}$$

for all $\ell = 1,\dots,k$. Eq. (3) and this second induction hypothesis applied to $\ell = k-1$ then imply that

$$\begin{aligned}(k-1)f(i,k) &= (k-1)p_i r_i + (k-1)p_i f(i+1,k) + (k-1)(1-p_i)f(i,k-1) \\ &\leq (k-1)p_i r_i + k p_i f(i+1,k-1) + (k-1)(1-p_i)f(i,k-1) \\ &= k\left[p_i(r_i + f(i+1,k-1)) + (1-p_i)f(i,k-2)\right] \\ &\quad - p_i r_i + (k-1)(1-p_i)f(i,k-1) - k(1-p_i)f(i,k-2) \\ &= k f(i,k-1) - p_i r_i + (1-p_i)\left[(k-1)f(i,k-1) - k f(i,k-2)\right].\end{aligned} \tag{24}$$

Now observe that it follows from the first induction hypothesis and Lemma 1 that

$$\begin{aligned}(k-1)f(i,k-1) - k f(i,k-2) &\leq (k-1)f(i,k-1) - \frac{k(k-2)}{k-1}f(i,k-1) \\ &= \frac{1}{k-1}f(i,k-1) \leq f(i,1) \leq Z_i.\end{aligned}$$

Here, we use the first induction hypothesis twice: once in the first inequality (by applying Inequality (22) to $\ell = k-2$) and again in the penultimate inequality (by applying Inequality (22) to $\ell = 1$). Substituting this into Inequality (24) then yields $(k-1)f(i,k) \leq k f(i,k-1)$. Hence, by Inequality (22),

$$\frac{f(i,k)}{k} \leq \frac{f(i,k-1)}{k-1} \leq \frac{f(i,\ell)}{\ell}$$

for all $\ell = 1,\dots,k-1$. Since we also trivially have $f(i,k)/k \leq f(i,\ell)/\ell$ for $\ell = k$, this completes the induction, and therefore the proof. □

Proof of Lemma 3. Consider an arbitrary $i \in N \setminus \{n\}$ and $k \geq 0$. Referring to the notation introduced in the proof of Theorem 2, it then holds that $f(i,k) = v(\{i,\dots,n\},k)$ and $f(i+1,k) = v(\{i+1,\dots,n\},k)$. Since having an additional job available can never decrease the maximum expected revenue, we know that $v(\{i,\dots,n\},k) \geq v(\{i+1,\dots,n\},k)$, which yields the result. □

Appendix D. Proofs from Section 4

Proof of Lemma 4. Consider an arbitrary $p \in (0,1)$ and $k \in \mathbb{N}_0$. Since one can easily verify that the inequality holds for $k = 1$, we assume that $k \geq 2$. Using Eq. (10), we can rewrite Inequality (11) as

$$p\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p^\ell(1-p)^{k-1-\ell}\left(H_k - H_{\ell+1}\right) \leq (1-p)\left(1-\frac{1}{k}\right),$$

which in turn is equivalent to

$$\sum_{\ell=0}^{k-2}\binom{k-1}{\ell}p^{\ell+1}(1-p)^{k-2-\ell}\left(H_k-H_{\ell+1}\right)\le\frac{k-1}{k}. \tag{25}$$

Now observe that for every $\ell=0,\dots,k-2$

$$H_k-H_{\ell+1}=\sum_{u=\ell+2}^{k}\frac{1}{u}\le\frac{k-1-\ell}{\ell+2}$$

and

$$\binom{k-1}{\ell}\cdot\frac{k-1-\ell}{\ell+2}=\frac{(k-1)!}{(\ell+1)!(k-2-\ell)!}\cdot\frac{\ell+1}{\ell+2}=\binom{k-1}{\ell+1}\cdot\frac{\ell+1}{\ell+2}.$$

Applying this to the left-hand side of Inequality (25) yields

$$\begin{aligned}
&\sum_{\ell=0}^{k-2}\binom{k-1}{\ell}p^{\ell+1}(1-p)^{k-2-\ell}\left(H_k-H_{\ell+1}\right)\\
&\quad\le\sum_{\ell=0}^{k-2}\binom{k-1}{\ell}p^{\ell+1}(1-p)^{k-2-\ell}\frac{k-1-\ell}{\ell+2}\\
&\quad=\sum_{\ell=0}^{k-2}\binom{k-1}{\ell+1}p^{\ell+1}(1-p)^{k-2-\ell}\frac{\ell+1}{\ell+2}\\
&\quad=\sum_{\ell=1}^{k-1}\binom{k-1}{\ell}p^{\ell}(1-p)^{k-1-\ell}\frac{\ell}{\ell+1}\\
&\quad\le\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p^{\ell}(1-p)^{k-1-\ell}\frac{k-1}{k}=\frac{k-1}{k},
\end{aligned}$$

where the second-to-last line follows from Eq. (10), and the final line follows since $\ell k\le(k-1)(\ell+1)$ for all $\ell=1,\dots,k-1$. This establishes Inequality (25) and therefore completes the proof. □

Proof of Lemma 5. Consider an arbitrary $p\in(0,1)$. We prove the result by induction on m. For $m=1$, it immediately follows from Eqs. (3), (8), (17), and (18) that

$$\lim_{n\to\infty}F(n,p,1)=\lim_{n\to\infty}H(n,p,1)=\sum_{j=1}^{\infty}p^j=\frac{p}{1-p}=F(p,1)=H(p,1).$$

Hence, the result holds for $m=1$. Now consider an arbitrary $m\ge2$ and, as the induction hypothesis, assume that $\lim_{n\to\infty}F(n,p,k)=F(p,k)$ and $\lim_{n\to\infty}H(n,p,k)=H(p,k)$ for all $k=1,\dots,m-1$. For arbitrary $n\in\mathbb{N}_0$, let $P(n)$ and $Q(n)$ denote the probability that all n jobs can be completed successfully in the solution obtained from the sequential upper bound and Z-rule heuristic, respectively. Since for given m and p the expected revenue for an instance with n jobs only differs from the one with $n-1$ jobs if all n jobs succeed, and since we consider unit revenues, it holds that $F(n,p,m)=F(n-1,p,m)+P(n)$ and $H(n,p,m)=H(n-1,p,m)+Q(n)$. Eq. (3) then yields that

$$\begin{aligned}
F(n,p,m)&=p\left[1+F(n-1,p,m)\right]+(1-p)F(n,p,m-1)\\
&=p\left[1+F(n,p,m)-P(n)\right]+(1-p)F(n,p,m-1)
\end{aligned}$$

and therefore that

$$F(n,p,m)=\frac{p\left[1-P(n)\right]}{1-p}+F(n,p,m-1).$$

Since $p<1$, we have $\lim_{n\to\infty}P(n)=0$. Combined with the induction hypothesis, we obtain

$$\lim_{n\to\infty}F(n,p,m)=\frac{p\left[1-\lim_{n\to\infty}P(n)\right]}{1-p}+\lim_{n\to\infty}F(n,p,m-1)=m\frac{p}{1-p},$$

which establishes that $\lim_{n\to\infty}F(n,p,m)=F(p,m)$. Analogously, Eq. (8) yields that

$$\begin{aligned}
H(n,p,m)&=\sum_{\ell=1}^{m}\binom{m}{\ell}p^{\ell}(1-p)^{m-\ell}\left[1+H(n-1,p,\ell)\right]\\
&=p^m\left[1+H(n,p,m)-Q(n)\right]\\
&\quad+\sum_{\ell=1}^{m-1}\binom{m}{\ell}p^{\ell}(1-p)^{m-\ell}\left[1+H(n-1,p,\ell)\right]
\end{aligned}$$

and therefore that

$$H(n,p,m)=\frac{p^m\left[1-Q(n)\right]}{1-p^m}+\sum_{\ell=1}^{m-1}\binom{m}{\ell}\frac{p^{\ell}(1-p)^{m-\ell}}{1-p^m}\left[1+H(n-1,p,\ell)\right].$$

Using $\lim_{n\to\infty}Q(n)=0$ and the induction hypothesis, we thus obtain

$$\begin{aligned}
\lim_{n\to\infty}H(n,p,m)&=\frac{p^m\left[1-\lim_{n\to\infty}Q(n)\right]}{1-p^m}\\
&\quad+\sum_{\ell=1}^{m-1}\binom{m}{\ell}\frac{p^{\ell}(1-p)^{m-\ell}}{1-p^m}\left[1+\lim_{n\to\infty}H(n-1,p,\ell)\right]\\
&=\frac{p^m}{1-p^m}+\sum_{\ell=1}^{m-1}\binom{m}{\ell}\frac{p^{\ell}(1-p)^{m-\ell}}{1-p^m}\left[1+H(p,\ell)\right].
\end{aligned}$$

This establishes that also $\lim_{n\to\infty}H(n,p,m)=H(p,m)$, and therefore completes the proof. □

Data availability

The data used for the experiments is available from the authors.

References

Adiri, I., Bruno, J.L., Frostig, E., Kan, A.H.G.R., 1989. Single machine flow-time scheduling with a single breakdown. Acta Inform. 26 (7), 679–696.

Agnetis, A., Benini, M., Detti, P., Hermans, B., Pranzo, M., 2022. Replication and sequencing of unreliable jobs on parallel machines. Comput. Oper. Res. 139, 105634.

Agnetis, A., Detti, P., Pranzo, M., 2014. The list scheduling algorithm for scheduling unreliable jobs on two parallel machines. Discrete Appl. Math. 165, 2–11.

Agnetis, A., Detti, P., Pranzo, M., Sodhi, M.S., 2009. Sequencing unreliable jobs on parallel machines. J. Sched. 12 (1), 45–54.

Agnetis, A., Lidbetter, T., 2020. The Largest-Z-ratio-First algorithm is 0.8531-approximate for scheduling unreliable jobs on m parallel machines. Oper. Res. Lett. 48 (4), 405–409.

Benoit, A., Robert, Y., Rosenberg, A.L., Vivien, F., 2013. Static strategies for worksharing with unrecoverable interruptions. Theory Comput. Syst. 53, 386–423.

Birge, J., Frenk, J., Mittenthal, J., Rinnooy Kan, A., 1990. Single-machine scheduling subject to stochastic breakdowns. Naval Res. Logist. 37, 661–677.

Cai, X., Wu, X., Zhou, X., 2009. Stochastic scheduling subject to preemptive-repeat breakdowns with incomplete information. Oper. Res. 57 (5), 1236–1249.

Dean, B.C., Goemans, M.X., Vondrák, J., 2008. Approximating the stochastic knapsack problem: The benefit of adaptivity. Math. Oper. Res. 33 (4), 945–964.

French, S., 1986. Decision Theory: An Introduction to the Mathematics of Rationality. Ellis Horwood.

Nadimpalli, C., Muttamsetty, L., Pamu, R., 2025. Dark Factories and Lights-Out Manufacturing: The Future of Production, in Altug Gunar (ed.), Economic and Political Consequences of AI: Managing Creative Destruction. IGI Global Scientific Publishing, pp. 233–266.

Nemhauser, G., Wolsey, L., Fisher, M., 1978. An analysis of approximations for maximizing submodular set functions - I. Math. Program. 14, 265–294.

Qi, X., Bard, J., Yu, G., 2006. Disruption management for machine scheduling: The case of SPT schedules. Int. J. Prod. Econ. 103, 166–184.

Schmidt, G., 2000. Scheduling with limited machine availability. European J. Oper. Res. 121 (1), 1–15.

Sharma, T., Kechagia, M., Georgiou, S., Tiwari, R., Vats, I., Moazen, H., Sarro, F., 2024. A survey on machine learning techniques applied to source code. J. Syst. Softw. 209, 111934.

Stadje, W., 1995. Selecting jobs for scheduling on a machine subject to failure. Discrete Appl. Math. 63 (3), 257–265.

Tian, X., Liu, T., Wang, Q., Dilmurat, A., Li, D., Ziegmann, G., 2017. Recycling and remanufacturing of 3D printed continuous carbon fiber reinforced PLA composites. J. Clean. Prod. 142, 1609–1618.

Yang, Q., Mishra, D., Awasthi, U., Bollas, G.M., Pattipati, K.R., 2024. Tool wear and remaining useful life estimation in precision machining using interacting multiple model. J. Manuf. Syst. 74.