List scheduling is 0.8531-approximate for scheduling unreliable jobs on $m$ parallel machines

Alessandro Agnetis * Thomas Lidbetter †

Abstract

In this paper we analyze the worst-case performance of the list scheduling algorithm for the problem of scheduling unreliable jobs on $m$ parallel machines. Each job is characterized by a success probability and a reward earned in the case of success. In the case of failure, the jobs subsequently sequenced on that machine cannot be performed. The objective is to maximize the expected reward. We show the algorithm provides an approximation ratio of $\simeq 0.853196$, and that the bound is tight.

Keywords: Unreliable jobs; List scheduling; Approximation ratio.

1 Introduction

The following problem has been introduced in [2], called Unreliable Job scheduling Problem (UJP). A set of jobs $J = \{J_1, \ldots, J_n\}$ must be assigned to $m$ parallel, identical machines, $M_1, \ldots, M_m$. Each job must be assigned to a single machine and a machine can process one job at a time. Jobs are unreliable, i.e., while a job is being processed by a machine, a failure can occur, which implies losing all the work which was scheduled but not yet executed by the machine. Each job $J_i$ is characterized by a certain success probability $\pi_i$ (independent from other jobs) and a reward $r_i$, which is gained if the job is successfully completed. We assume that the values $\pi_i$ are rational numbers. The problem is to find an assignment of jobs to

* Dipartimento di Ingegneria dell’Informazione, Università di Siena, e-mail agnetis@dii.unisi.it
† Management Science and Information Systems Department, Rutgers Business School, Rutgers University, e-mail tlidbetter@business.rutgers.edu
the $m$ machines and a sequence on each machine that maximizes the total expected reward.

We may also frame UJP as a search and rescue problem, as in [5]. A single target, which must be rescued, is hidden in one of $n$ locations according to a known probability distribution. The probability that location $i$ contains the target is $r_i$, so the sum of the $r_i$ must be 1. When location $i$ is searched, there is a certain probability that the searcher will be captured herself, so that she can search no further locations. The target will be rescued only if a searcher searches its location before being captured. The probability that the searcher will not be captured when searching location $i$ is $\pi_i$, and these probabilities are independent. The problem is to distribute the locations among $m$ searchers, and choose an order of search for each searcher, so as to maximize the probability that the target is found. Further links between search and scheduling problems are considered in [3].

In this note we consider the list scheduling algorithm for UJP, and, for any instance $I$, give a bound on the ratio $\lambda(I)$ between the value produced by the list scheduling algorithm and the optimal value. Limited to the case $m = 2$, it was shown in [1] that $\lambda(I) \geq (2 + \sqrt{2})/4 \simeq 0.8535...$. Here we extend the result to any value of $m$, showing that $\lambda(I) \geq 0.85319...$

In Section 2 we review some basic notions concerning UJP, while in Section 3 we provide the main result.

## 2 Unreliable Job scheduling Problem

Here we briefly review the main concepts and notation concerning UJP. Let $S^h$ be a sequence of $K$ jobs assigned to machine $M_h$, and let $S^h(k)$ be the job in $k$-th position in $S^h$. A feasible solution $S = \{S^1, S^2, \ldots, S^m\}$ for UJP is an assignment and sequencing of the $n$ jobs on the $m$ machines. If $K$ jobs are assigned to $M_h$, the expected reward of sequence $S^h$ is given by

$$ER[S^h] = \pi_{S^h(1)}r_{S^h(1)} + \pi_{S^h(1)}\pi_{S^h(2)}r_{S^h(2)} + \ldots + \pi_{S^h(1)}\ldots\pi_{S^h(K-1)}\pi_{S^h(K)}r_{S^h(K)},$$

and the total expected reward is therefore

$$ER[S] = ER[S^1] + ER[S^2] + \ldots + ER[S^m].$$

UJP consists in finding a solution $S_{OPT} = \{S_{OPT}^1, S_{OPT}^2, \ldots, S_{OPT}^m\}$ that maximizes the total expected reward. A key role in our developments is played...
by the following quantity associated with each job $j$, called the $Z$-ratio:

$$Z_j = \frac{\pi_j r_j}{1 - \pi_j}.$$  \hfill (2)

When $m = 1$, the optimal solution is achieved by sequencing the jobs in non-increasing order of $Z_j$ [6,2]. Hence, UJP indeed consists in deciding how to partition the $n$ jobs among the $m$ machines, since on each machine the sequencing is then dictated by the priority rule (2). Since PRODUCT PARTITION can be polynomially reduced to UJP with $m = 2$ [2], and since PRODUCT PARTITION is strongly NP-hard [7], so is UJP, even for $m = 2$.

UJP bears various similarities with the classical problem of minimizing total weighted completion time on $m$ parallel machines, i.e., $Pm||\sum_j w_j C_j$. The single-machine problem $1||\sum_j w_j C_j$ is solved by the well-known Smith’s rule, i.e., sequencing the jobs in non-increasing order of the ratio $\rho_j = w_j/p_j$. For any $m \geq 2$, $Pm||\sum_j w_j C_j$ is NP-hard.

The list scheduling algorithm (LSA) for $Pm||\sum_j w_j C_j$ is the following: order the jobs by non-increasing $Z$ ratios $\rho_j$ and assign them in this order to the $m$ machines, allocating the next job in the list to the machine that frees up first. A schedule obtained in this way is also called Largest-Ratio-First (LRF) schedule. Kawaguchi and Kyan [4] showed that the worst-case error of any LRF schedule is $(1 + \sqrt{2})/2$. A simpler proof of this result has been provided by Schwiegelshohn [8].

In this paper we analyze the performance of LSA for UJP. In the following, while assigning the jobs to machines, we call the cumulative probability of a machine the product of the success probabilities of the jobs already scheduled on that machine. When a job $j$ is assigned to a machine, we use the notation $P_j$ to indicate the product of the success probabilities of all jobs scheduled on the machine up to job $j$ (included), and we refer to $P_j$ as the cumulative probability of job $j$.

The list scheduling algorithm for UJP works as follows. Order the jobs by non-increasing $Z_j$ and assign them in this order to the $m$ machines, allocating the next job in the list to a machine currently having maximum cumulative probability (ties are broken arbitrarily). A schedule obtained in this way is also called Largest-Z ratio-First (LZF) schedule. In this paper we investigate the worst-case performance of any LZF schedule.

In establishing our result, we follow a similar line of reasoning to the one in [8] for $Pm||\sum_j w_j C_j$. While our Lemmas 3.1 and 3.2 are an adaptation of Corollaries 1 and 3 in [8], Lemma 3.3 exploits features that are specific to UJP.
3 An approximation bound

The bound on the performance of an LZF schedule is proved by subsequently reducing the set of instances which need to be considered in order to detect the worst-case instance. This is done through three lemmas. In Lemma 3.1 we show that we can restrict to instances in which all jobs have \( Z_j = 1 \). In Lemma 3.2 we prove that it is sufficient to consider instances containing at most \( m - 1 \) jobs having a very large revenue (so-called second-stage jobs) and an arbitrary number of jobs having small revenue (low-value jobs). Lemma 3.3 shows that, furthermore, the worst-case situation occurs when the success probability of second-stage jobs is arbitrarily close to 0. Thereafter, the main result can be established.

As in [8], we extend the usual definition of an instance \( I \) of the problem to include an arbitrary LZF order for all the jobs. This order produces the primary LZF schedule \( S_{LZF}(I) \). In this way, an instance \( I \) has a unique primary LZF schedule, even if all jobs have the same \( Z_j \). For an instance \( I \), we let

\[
\lambda(I) = \frac{ER[S_{LZF}(I)]}{ER[S_{OPT}(I)]}.
\]

The following lemma is an adaption of Corollary 1 in [8].

**Lemma 3.1** For every instance \( I \) of UJP, there is an instance \( I' \) with \( \lambda(I') \leq \lambda(I) \) and \( Z_j = 1 \) for all jobs \( j \in I' \).

Proof. Let \( \zeta_1 > \zeta_2 > \cdots > \zeta_d \) be the \( d \) different \( Z_j \) values of jobs in \( I \), and let \( \zeta_{d+1} = 0 \). We can write \( \zeta_i \) as \( \zeta_i = \sum_{k=i}^{d} (\zeta_k - \zeta_{k+1}) \). Letting \( i(j) \) denote the index of the \( Z \)-ratio of job \( j \), one has

\[
r_j = \zeta_{i(j)} \left( 1 - \frac{\pi_j}{\pi} \right) = \sum_{k=i(j)}^{d} (\zeta_k - \zeta_{k+1}) \frac{1 - \pi_j}{\pi_j}.
\]

Recalling the definition of \( P_j \), the expected revenue of an arbitrary schedule \( S \) for instance \( I \) can be written as

\[
ER[S] = \sum_{j \in I} r_j P_j = \sum_{j \in I} \left( \sum_{k=i(j)}^{d} (\zeta_k - \zeta_{k+1}) \frac{1 - \pi_j}{\pi_j} \right) P_j =
\]

\[
= \sum_{k=1}^{d} \left( \zeta_k - \zeta_{k+1} \right) \sum_{j : i(j) \leq k} \frac{1 - \pi_j}{\pi_j} P_j.
\]

(3)
Next, we define a sequence of instances of UJP, \( I_k = \{ j \in I : i(j) \leq k \}, \)
\( k = 1, \ldots, d. \) For these instances, we set the reward of job \( j \) in \( I_k \) as \( (1 - \pi_j) / \pi_j \) (hence, \( Z_j = 1 \)). It follows that any ordering of the jobs in \( I_k \) is an LZF order, so we can select an LZF order for \( I_k \) that is consistent with our LZF order for \( I \), ensuring that for any job \( j \in I_k \), the values of \( P_j \) in \( S_{LZF}(I_k) \) and in \( S_{LZF}(I) \) are identical. Letting \( P_j^* \) denote the value of \( P_j \) in an optimal schedule \( S_{OPT}(I_k) \), and observing that \( ER[S_{LZF}(I_k)] = \sum_{j \in I_k} ((1 - \pi_j) / \pi_j) P_j \), we can apply (3) to both \( S_{LZF}(I) \) and \( S_{OPT}(I) \) to obtain

\[
\lambda(I) = \frac{\sum_{k=1}^d (\zeta_k - \zeta_{k+1}) ER[S_{LZF}(I_k)]}{\sum_{k=1}^d (\zeta_k - \zeta_{k+1}) \sum_{j \in I_k} \frac{1 - \pi_j}{\pi_j} P_j^*}.
\]

By the optimality of \( S_{OPT}(I_k) \), we must have \( ER[S_{OPT}(I_k)] \geq \sum_{j \in I_k} ((1 - \pi_j) / \pi_j) P_j^* \), so that

\[
\lambda(I) \geq \frac{\sum_{k=1}^d (\zeta_k - \zeta_{k+1}) ER[S_{LZF}(I_k)]}{\sum_{k=1}^d (\zeta_k - \zeta_{k+1}) ER[S_{OPT}(I_k)]} \geq \min_{1 \leq k \leq d} \lambda(I_k).
\]

Hence, \( \lambda(I) \) is at least as large as the value it attains in an instance in which all jobs have Z-ratio equal to 1. \( \square \)

Notice that if \( Z_j = 1 \) for all jobs, the expected revenue of the jobs scheduled on a certain machine \( M_h \), from (1), is given by

\[
ER[S_h] = \pi_{S_h(1)} \left( \frac{1 - \pi_{S_h(1)}}{\pi_{S_h(1)}} \right) + \pi_{S_h(1)} \pi_{S_h(2)} \left( \frac{1 - \pi_{S_h(2)}}{\pi_{S_h(2)}} \right) + \ldots + \\
+ \pi_{S_h(1)} \ldots \pi_{S_h(K-1)} \pi_{S_h(K)} \left( \frac{1 - \pi_{S_h(K)}}{\pi_{S_h(K)}} \right) = 1 - \prod_{i=1}^K \pi_{S_h(i)}
\]

and hence, given a schedule \( S \), if \( P_h(S) = \prod_{i=1}^K \pi_{S_h(i)} \) is the cumulative probability of machine \( M_h \) in schedule \( S \), the expected reward \( ER[S] \) is given by

\[
ER[S] = m - \sum_{h=1}^m P_h(S).
\]

In view of Lemma 3.1, from now on we only consider instances of UJP in which all jobs have Z-ratio equal to 1.

Given an instance \( I \) and the corresponding primary schedule \( S_{LZF} \), let \( P_{\max}(I) = \max_h \{ P_h(S_{LZF}(I)) \} \). We can now establish the UJP counterpart of Corollary 3 in [8].
Lemma 3.2 For every instance $I$ of UJP, there is an instance $I'$ such that $\lambda(I') \leq \lambda(I)$ and every job has an arbitrarily high success probability if its cumulative probability in $S_{LZF}(I')$ is at least $P_{\text{max}}(I')$.

Proof. Consider an arbitrary instance $I$ and the corresponding LZF schedule $S_{LZF}(I)$. Now consider an instance $J$ obtained by replacing any job $j$ with two jobs $j_1$ and $j_2$ such that $\pi j_1 \pi j_2 = \pi j$, and consider the schedule obtained from $S_{LZF}(I)$ replacing $j$ with $j_1$ and $j_2$ consecutively scheduled in this order on the same machine, call it $S(J)$. Note that $ER[S_{LZF}(I)] = ER[S(J)]$, by $\Box$. Call $\bar{P}$ the cumulative probability of the jobs preceding $j$ on the same machine in $S_{LZF}(I)$. Due to the mechanism of the LZF algorithm, $\bar{P} \geq P_{\text{max}}(I)$. We choose $\pi j_1$ so that $\pi j_1 \bar{P} \geq P_{\text{max}}(I)$. In this case, $S(J)$ is still an LZF schedule. (This is not the case if $\pi j_1 \bar{P} < P_{\text{max}}(I)$, as a LZF schedule would have assigned $j_2$ on the machine that in $S_{LZF}(I)$ has cumulative probability $P_{\text{max}}(I)$.) Also, note that $ER[S_{OPT}(J)] \geq ER[S_{OPT}(J)]$. Hence, as long as $S(J)$ is a LZF schedule,

$$\lambda(I) = \frac{ER[S_{LZF}(I)]}{ER[S_{OPT}(I)]} \geq \frac{ER[S(J)]}{ER[S_{OPT}(J)]} = \lambda(J).$$

We can repeat this job splitting until all jobs $j$ such that $P_j \geq P_{\text{max}}(I)$ have an arbitrarily large success probability. Note all the jobs such that $P_j < P_{\text{max}}(I)$ are the last scheduled jobs on the various machines, and hence they can be at most $m - 1$. $\square$

The consequence of Lemma 3.2 is that we can restrict to instances satisfying the following. Each machine processes a large number of jobs with an arbitrarily high success probability, until its cumulative probability reaches $P_{\text{max}}(S_{LZF}(I))$. We call these jobs low-value jobs, since, if the success probability is high, then the reward is low. After these jobs, at most $m - 1$ machines process one more job. We call these jobs second-stage jobs. So, in summary, from now on we only consider instances which contain several low-value jobs followed by at most $m - 1$ second-stage jobs. In fact, by adding some jobs with success probability 1 and reward 0 to the end of $I$, we can assume that in $S_{LZF}(I)$, all machines process one second-stage job last (at least one of which has reward equal to 0). (This does not alter the expected reward of any schedule for $I$.)

We now take the last fundamental step, which consists in showing that the most unfavourable situation occurs when the success probability of all second-stage jobs is either 1 or it is arbitrarily low. The proof of this lemma uses arguments that are specific of UJP and are not derived from $\Box$. In
order to simplify notation and streamline the proof, we assume that some 
jobs have success probability 0, as a shorthand for jobs having \( \pi_j = \varepsilon \) with 
\( \varepsilon \to 0 \). Notice that, from (4), the expected reward of a machine on which 
one such job is scheduled is 1.

**Lemma 3.3** For every instance \( I \) of UJP, there is an instance \( I' \) such that 
\( \lambda(I') \leq \lambda(I) \) and every job has an arbitrarily high success probability if its 
cumulative probability in \( S_{LZF}(I') \) is at least \( P_{\text{max}}(I') \). All other jobs have 
success probability equal to 0 or 1.

Proof. Given an instance \( I \), let \( p = P_{\text{max}}(I) \) and denote by \( t \in (0, m) \) 
the sum of the success probabilities of the second-stage jobs in \( I \). We will 
prove that we can construct an instance \( I' \) with the required properties in 
the case that \( t \) is an integer. In the case that \( t \) is not an integer, let \( q \) be 
an integer such that \( tq \) is an integer. Given \( I \), consider an instance \( I^q \) in 
which each job of \( I \) is replaced with \( q \) consecutive copies, and there are \( qm \) 
machines. In \( I^q \), we adopt a LZF order producing \( k \) identical copies of each 
machine schedule in \( S_{LZF}(I) \), so that \( ER[S_{LZF}(I^q)] = qER[S_{LZF}(I)] \). It 
is also clear that \( ER[S_{OPT}(I^q)] \geq qER[S_{OPT}(I)] \), since any schedule for \( I \) 
gives rise to a schedule for \( I^q \) whose expected reward is \( q \) times as large. 
Hence, \( \lambda(I^q) \leq \lambda(I) \), and it is sufficient to find a schedule \( I' \) such that 
\( \lambda(I') \leq \lambda(I^q) \). But the sum of the success probabilities of the second-stage 
jobs in \( I^q \) is an integer, so we may as well assume \( t \) is an integer (otherwise, 
the whole argument is applied to \( I^q \)).

Recall that in \( S_{LZF}(I) \) every machine processes a large number of low-
value jobs with cumulative probability \( p = P_{\text{max}}(I) \), possibly followed by a 
second-stage job. Let \( k = k(I) \) be the number of second-stage jobs in \( I \) with 
success probability not equal to 0 or 1. For every fixed \( m \), we will prove by 
induction on \( k \) that there is an instance \( I' \) such that \( \lambda(I') \leq \lambda(I) \) and all the 
second-stage jobs of \( \lambda(I') \) have success probability 0 or 1, which will prove 
the thesis.

This is evidently true for \( k = 0 \), because in that case \( I \) is already of the 
form \( I' \). Note that we cannot have \( k = 1 \), since in this case a single second-
stage job would have success probability \( 0 < \tilde{\pi} < 1 \) and \( t = m - 1 + \tilde{\pi} \) would 
be non-integer, contradicting the assumption that \( t \) is integer.

Although not strictly necessary, we consider separately the case \( k = 2 \), 
since this is a good introduction to the structure of the general induction 
argument. In this case, in \( I \) there must be two second-stage jobs \( i \) and \( j \) 
with success probabilities \( \pi_i, \pi_j \in (0, 1) \) such that \( \pi_i + \pi_j = 1 \) (again, since \( t \) 
is integer). We define a new instance \( I' \) by replacing jobs \( i \) and \( j \) with jobs
\(i'\) and \(j'\) having success probabilities 0 and 1 respectively. Clearly this does not affect \(t\), so the expected reward of \(S_{LZF}(I)\) and \(S_{LZF}(I')\) are the same, and \(k(I') = 0\). We must show that there is a schedule for \(I'\) whose expected reward is at least that of \(ER[S_{OPT}(I)]\), so that \(\lambda(I') \leq \lambda(I)\).

Consider \(S_{OPT}(I)\), and define a schedule \(S'(I')\) for \(I'\) which is the same as \(S_{OPT}(I)\), but replacing \(i\) and \(j\) with \(i'\) and \(j'\). There are two possibilities. Either, in \(S_{OPT}(I)\), jobs \(i\) and \(j\) are processed on the same machine or on different machines. If they are processed on the same machine, then the expected reward of \(S'(I')\) must be greater than that of \(S_{OPT}(I)\), since the machine that processes \(i'\) and \(j'\) has expected reward 1, due to \(i'\) having success probability 0. Now suppose that \(i\) and \(j\) are processed on different machines in \(S_{OPT}(I)\), and let the cumulative probabilities of all the other jobs scheduled by \(S_{OPT}(I)\) on these two machines be \(p_1\) and \(p_2\), respectively. Assume, without loss of generality, that \(p_1 \geq p_2\) (otherwise we could replace \(i\) with \(j'\) and \(j\) with \(i'\) in \(S'(I')\)). Then the contribution of these two machines to the expected reward of \(S'(I')\) is \(2 - p_2\) and to \(S_{OPT}(I)\) is \(2 - p_1\pi_i - p_2\pi_j\). Recalling that \(\pi_i + \pi_j = 1\), the difference between these two contributions is \(p_1\pi_i + p_2(\pi_j - 1) = \pi_i(p_1 - p_2) \geq 0\).

Now consider any \(k \geq 3\), and assume that the induction hypothesis is true for all smaller values of \(k\). Let \(i\) and \(j\) be any two second-stage jobs in \(I\) with success probabilities \(\pi_i \in (0, 1)\) and \(\pi_j \in (0, 1)\). We consider two cases.

Case 1: \(\pi_i + \pi_j \leq 1\). This is similar to the case \(k = 2\). We define a new instance \(I'\) by replacing jobs \(i\) and \(j\) with \(i'\) and \(j'\) that have success probabilities 0 and \(\pi_i + \pi_j\) respectively. This does not affect \(t\), so the expected reward of \(S_{LZF}(I)\) and \(S_{LZF}(I')\) are the same, and also we have that \(k(I') < k(I)\). By the induction hypothesis, it is sufficient to show that there is a schedule for \(I'\) whose expected reward is at least that of \(S_{OPT}(I)\). As before, we define a schedule \(S'(I')\) for \(I'\) which is obtained from \(S_{OPT}(I)\) replacing \(i\) and \(j\) with \(i'\) and \(j'\). Again, there are two possibilities: either \(i\) and \(j\) are processed on the same machine in \(S_{OPT}(I)\) or not. In the former case, \(ER[S'(I')] \geq ER[S_{OPT}(I)]\) since the machine that processes \(i'\) and \(j'\) has expected reward 1 in \(S'(I')\). In the latter case, let the cumulative probabilities of all the other jobs scheduled by \(S_{OPT}(I)\) on these two machines be \(p_1\) and \(p_2\), respectively. Assume again, without loss of generality, that \(p_1 \geq p_2\). Then the contribution of these two machines to \(ER[S'(I')]\) is \(2 - p_2(\pi_i + \pi_j)\) and to \(ER[S_{OPT}(I)]\) is \(2 - p_1\pi_i - p_2\pi_j\). The difference between these contributions is \(-p_2(\pi_i + \pi_j) + p_1\pi_i + p_2\pi_j = \pi_i(p_1 - p_2) \geq 0\).

Case 2: \(\pi_i + \pi_j > 1\). In this case, we define a new instance \(I''\) by replacing jobs \(i\) and \(j\) in \(I\) with jobs \(i''\) and \(j''\) that have success probabilities \(\pi_i + \pi_j - 1\)
and 1 respectively. Again, from (5), \( ER[S_{LZF}(I)] = ER[S_{LZF}(I'')] \), and \( k(I'') < k(I) \), so by the induction hypothesis, it is sufficient to show that there is a schedule for \( I'' \) whose expected reward is at least that of \( S_{OPT}(I) \). We define a schedule \( S''(I'') \) for \( I'' \) which is obtained from \( S_{OPT}(I) \) replacing \( i \) and \( j \) with jobs \( i'' \) and \( j'' \). Again there are two subcases. The first is when \( i \) and \( j \) are processed on the same machine in \( S_{OPT}(I) \). In this case, let \( \hat{p} \) be the cumulative success probability of all the other jobs processed on this machine. Then the contribution of this machine to the expected reward of \( S''(I'') \) is \( 1 - \hat{p}(\pi_i + \pi_j - 1) \) and to \( S_{OPT}(I) \) is \( 1 - \hat{p}\pi_i\pi_j \). The difference between the two contributions is therefore

\[
\hat{p}(\pi_i\pi_j - \pi_i - \pi_j + 1) = \hat{p}(1 - \pi_i)(1 - \pi_j) > 0.
\]

The second subcase is when \( i \) and \( j \) are processed on different machines in \( S_{OPT}(I) \). Again, let the cumulative probabilities of all the other jobs scheduled by \( S_{OPT}(I) \) on these two machines be \( p_1 \) and \( p_2 \) respectively, with \( p_1 \geq p_2 \). Then the difference between the contributions of these two machines to \( ER[S'(I')] \) and \( ER[S_{OPT}(I)] \) respectively is

\[
(2 - p_1(\pi_i + \pi_j - 1) - p_2) - (2 - p_1\pi_i - p_2\pi_j) = (1 - \pi_j)(p_1 - p_2) \geq 0
\]

and the proof is complete. \( \square \)

We are now in the position of establishing the main result of this paper.

**Theorem 3.4** For any instance \( I \) of UJP, \( \lambda(I) \geq 0.853196... \)

Proof. From Lemmas 3.1, 3.2 and 3.3, we only need to consider instances \( I \) where, in \( S_{LZF}(I) \), each machine processes some low-value jobs first, until the cumulative success probability reaches some \( p \) on each machine, then \( m - t \) machines each process a job with success probability 0. Note that the optimal solution to such an instance \( I \) is that \( m - t \) machines each process one job of success probability 0, and the low-value jobs are evenly divided among the remaining \( t \) machines, so that each of these \( t \) machines has cumulative probability \( p^{m/t} \). Recalling that \( ER[S_{LZF}(I)] = m - tp \), \( \lambda(I) \) is given by

\[
\lambda(I) = \frac{m - tp}{m - tp^{m/t}} = \frac{m/t - p}{m/t - p^{m/t}} = f(m/t, p).
\]

The minimum of function \( f(m/t, p) \) (for \( p \in (0,1) \) and \( m/t \geq 1 \)) can be found by numerical methods and it has value 0.853195... \( \square \)
We observe that for \( m = 2 \) and \( t = 1 \), simple calculus shows that the minimum of \( f(2, p) \) is attained for \( p = 2 - \sqrt{2} \) and its value is \((2 + \sqrt{2})/4 \approx 0.8535...\), retrieving the result in \([1]\) for \( m = 2 \). One might expect that the minimum of \( f(m/t, p) \) for fixed \( m \) would be monotonically decreasing in \( m \), but this is in fact not the case. The approximate values of \( \min_{t,p} f(m/t, p) \) are shown in Table 1 for the first few values of \( m \).

Table 1: Approximate minimum values of \( f(m/t, p) \) for fixed \( m \).

| \( m \) | \( \min_{t,p} f(m/t, p) \) |
|-------|-----------------------------|
| 2     | 0.85355                     |
| 3     | 0.86179                     |
| 4     | 0.85355                     |
| 5     | 0.85541                     |
| 6     | 0.85355                     |

The minimum of function \( f(m/t, p) \) is achieved for \( m/t \approx 2.123105... \) and \( p \approx 0.589198... \). This can be used to build tight examples.

**Example 3.1** Consider an instance \( I \) of UJP with \( m = 21231 \). There are \( 2.1231 \cdot 10^8 \) low-value jobs, each having probability \( \hat{\pi} = (0.58919)^{1/10000} \), and \( 11231 \) jobs having 0 probability. Suppose that the primary schedule \( S_{LZF}(I) \) is the worst possible, consisting in first equally partitioning all low-value jobs among all the \( m \) machines, and then adding the 0 probability jobs in the end. Then, the product of the probabilities of all low-value jobs is \( \hat{\pi}^{21231} \) and hence \( P_{\text{max}}(I) = 0.58919 \). In \( S_{LZF}(I) \), the cumulative probability of 10000 machines is therefore equal to 0.58919, while it is 0 for the other 11231. Hence, \( ER[S_{LZF}(I)] = 21231 - 10000 \cdot 0.58919 = 15339.1 \).

The optimal schedule \( S_{OPT}(I) \) schedules the 0-probability jobs alone on 11231 machines, and evenly divides the low-value jobs among the other 10000 machines. As each of these machines receives 21231 low-value jobs, the cumulative probability of each of them is \( (0.58919)^{2,1231} \), and hence \( ER[S_{OPT}(I)] = 21231 - 10000 \cdot 0.58919^{2,1231} = 17978,41... \) The ratio is in this case \( \lambda(I) = ER[S_{LZF}(I)]/ER[S_{OPT}(I)] > 0.8531955... \). Building larger examples it is possible to get even closer to the bound of Theorem 3.4. \( \square \)
Acknowledgements

This work was supported by the National Science Foundation (NSF grant IIS-1909446).

References

[1] Agnetis, A., Detti, P., Pranzo, M., The list scheduling algorithm for scheduling unreliable jobs on two parallel machines, *Discrete Applied Mathematics*, 165, 2–11, 2014.

[2] Agnetis, A., Detti, P., Pranzo, M. and Sodhi M.S., Sequencing unreliable jobs on parallel machines, *Journal of Scheduling*, 12, 45–54, 2009.

[3] Fokkink, F., Lidbetter T., Végh, L. On submodular search and machine scheduling, *Mathematics of Operations Research*, https://doi.org/10.1287/moor.2018.0978.

[4] Kawaguchi, T., Kyan, S., Worst case bound of an LRF schedule for the mean weighted flow-time problem, *SIAM J. Computing*, vol. 15, 4, 1986.

[5] Lidbetter, T. Search and rescue in the face of uncertain threats, *arXiv preprint arXiv:1902.05432*.

[6] Mitten, L.G., An Analytic Solution to the Least Cost Testing Sequence Problem, *Journal of Industrial Engineering*, Vol. 11, 17, 1960.

[7] Ng, C.T., Barketau, M.S., Cheng, T.C.E., Kovalyov, M.Y., ”Product partition” and related problems of scheduling and systems reliability: Computational complexity and approximation, *European Journal of Operational Research*, 207, 601–604, 2010.

[8] Schwiegelshohn, U., An alternative proof of the Kawaguchi-Kyan bound for the Largest-Ratio-First rule, *Operations Research Letters*, 39, 255–259, 2011.