Robust Scheduling Against Machine Failures

Zhenwei Liu , **Guochuan Zhang** and **Yifan Zhao**
Zhejiang University
{lzw98,zgc,3220104389}@zju.edu.cn

Abstract

We study a *robust scheduling* problem on identical machines in which machines may fail after the initial assignment. The goal is to compute an initial schedule together with a recovery strategy that minimizes the post-failure makespan, under the constraint that jobs on available machines cannot be moved. We consider two failure models: a *strong adversary*, which selects the failed machines, and a *weak adversary*, which selects only the number of failures. For up to k failures, we give algorithms with robustness ratios 1.618 (strong adversary) and 1.5 (weak adversary). For the single-failure case ($k = 1$), we obtain *best possible* ratios 1.387 and 1.281, respectively, matching our lower bounds.

1 Introduction

Scheduling is a fundamental problem in operations research and computer science. The classical makespan-minimization problem asks for an assignment of jobs to machines that minimizes the completion time of the last job. Modern large-scale systems often face incomplete information or uncertainty: jobs may arrive over time (online scheduling [Albers and Hellwig, 2017]), processing times may be revealed only upon completion (non-clairvoyant scheduling [Motwani *et al.*, 1994; Lindermayr and Megow, 2025]), or processing times may be random with known distributions (stochastic scheduling [Niño-Mora, 2009]).

Beyond job-side uncertainty, the machine environment may also be uncertain. Stein and Zhong [2020] considered a robust scheduling problem where the number of machines is unknown a priori. The scheduler first packs jobs into bags; once the number of machines is revealed, it assigns the bags to machines. A generalized model with different machine speeds was studied in [Eberle *et al.*, 2023; Minarík and Sgall, 2024].

Following this line of work, we investigate a *robust scheduling* problem motivated by unreliable machine environments. We adopt the framework of *Recoverable Robustness* [Liebchen *et al.*, 2009]. Specifically, we are given a set of m identical machines $\mathcal{M}$ and a set of n jobs $\mathcal{J}$, with a known processing time p_j for each job $j \in \mathcal{J}$, and a parameter k. The goal is to assign jobs to machines so as to minimize the makespan while anticipating potential machine failures. We call this assignment the *initial schedule*. The parameter k is a known upper bound on the number of failures: after the scheduler commits an initial schedule, up to k machines may fail simultaneously, as decided by an adversary. Throughout, we focus on the nontrivial range $1 \le k \le m-1$, so that at least one machine remains available after failures. The scheduler then reassigns only the jobs initially on the failed machines to the available machines, minimizing the makespan after reassignment. Jobs initially assigned to available machines remain fixed. See Figure 1 for an illustration. This captures settings where reassignment is costly (e.g., cloud computing, where migrating running virtual machines is expensive [Rani *et al.*, 2025]) or where changing jobs on available machines is undesirable (e.g., railway planning [Lusby *et al.*, 2018]).

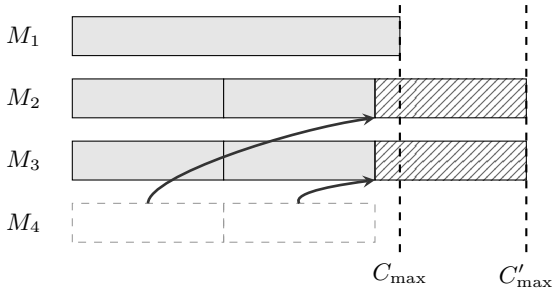

Figure 1: Reassignment after M_4 fails. $C_{\max}$ and $C'_{\max}$ are the makespans before and after the reassignment, respectively.

We formalize this setting through two adversary models.

The Strong Adversary. An adversary selects the *specific* set of failed machines $\mathcal{M}' \subset \mathcal{M}$. This captures settings where the initial schedule corresponds to a physical allocation of resources (e.g., machinery or hardware) that cannot be reconfigured freely.

The Weak Adversary. The adversary decides only the *number* of failed machines $k' \le k$, while the scheduler selects which machines to remove. This models *elastic scale-in* in cloud computing (e.g., AWS Auto Scaling), where a system manager must reduce the active cluster size by k' nodes but may choose the least disruptive nodes to drain and shut down.

In both models, a scheduler (or an algorithm) ALG consists of two components: an initial schedule and reassignment rules for each possible scenario of machine failures. We measure performance by a *robustness ratio*, analogous to the competitive ratio in online algorithms.

Evaluation with Robustness Ratio. Intuitively, the robustness ratio compares our recovered schedule with an optimal schedule that knows the failure scenario in advance. For each $1 \le i \le m$, let OPT_i be the optimal makespan when we use only i machines to complete all jobs. Fix an initial schedule S and a subset of machines $\mathcal{M}' \subset \mathcal{M}$. Let $\text{ALG}(S, \mathcal{M}')$ denote the makespan of a new schedule obtained from S by reassigning jobs from the failed machines $\mathcal{M}'$ to the available machines $\mathcal{M} \setminus \mathcal{M}'$ using the reassignment rules of ALG.

In the strong adversary model, an algorithm ALG is ρ-robust if for any instance, its initial schedule and its reassignment rules can guarantee that

$$\max_{\mathcal{M}' \subset \mathcal{M}: |\mathcal{M}'| \le k} \frac{\text{ALG}(S, \mathcal{M}')}{\text{OPT}_{m-|\mathcal{M}'|}} \le \rho. \tag{1}$$

That is, for every failure scenario, ALG can reassign jobs to the available machines and obtain a makespan at most ρ times the optimal makespan. When the reassignment rules are clear from context, we also say that the initial schedule S is ρ-robust.

Similarly, in the weak adversary model, an algorithm ALG is ρ-robust if for any instance, its initial schedule and its reassignment rules can guarantee that

$$\max_{0 \le k' \le k} \min_{\mathcal{M}' \subset \mathcal{M}: |\mathcal{M}'| = k'} \frac{\text{ALG}(S, \mathcal{M}')}{\text{OPT}_{m-|\mathcal{M}'|}} \le \rho. \tag{2}$$

That is, the adversary decides only the number of failed machines (the outer "max"), while the scheduler chooses which machines are treated as failed (the inner "min"). For each possible number of failed machines, ALG can choose which machines to treat as failed and then reassign jobs to achieve a makespan at most ρ times the optimal makespan.

1.1 Our Contributions and Techniques

We study robust scheduling with up to k machine failures under both adversary models. Table 1 summarizes our results. Our algorithms use optimal makespans for i machines, for each $i = 1, \ldots, m$; computing these values is NP-hard in general [Garey and Johnson, 1979]. With exact optimum schedules, our algorithms achieve the stated robustness ratios; for a polynomial-time guarantee, we can replace them by $(1+\varepsilon)$-approximate schedules computed in $\mathcal{O}(\text{poly}_{1/\varepsilon}(n))$ time (see, e.g., [Hochbaum and Shmoys, 1988]), losing only the usual $(1+\varepsilon)$ factor.

We next outline the main algorithmic ideas. Suppose first that we use a schedule attaining OPT_m as the initial schedule. When machines fail, we reassign jobs by list scheduling: each job on a failed machine is moved to an available machine of minimum current workload. Let $k' \le k$ be the number of failed machines and let job j be the last completed job after reassignment. This ensures a makespan of $\text{OPT}_{m-k'} + p_j \le \text{OPT}_{m-k'} + \text{OPT}_m \le 2\text{OPT}_{m-k'}$, which implies 2-robustness. To improve upon this, we leave some machines idle in the initial schedule. When a machine fails, its jobs can be moved to these idle machines without increasing the makespan, allowing the schedule to tolerate more failures. However, leaving more idle machines may increase the makespan of the initial schedule, which is undesirable when the number of failed machines is small. Our algorithms balance these effects by choosing how many machines to leave idle. For the special case $k = 1$, a finer case analysis yields better ratios that match our lower bounds.

Value of k	Strong Adversary		Weak Adversary	
	Lower	Upper	Lower	Upper
Arbitrary k	1.387 Sec. 3.1, Thm. 1	1.618 Sec. 3.1, Thm. 2	1.281 Sec. 4.1, Thm. 4	1.5 Sec. 4.1, Thm. 5
$k = 1$	1.387 Sec. 3.1, Thm. 1	1.387 Sec. 3.2, Thm. 3	1.281 Sec. 4.1, Thm. 4	1.281 Sec. 4.2, Thm. 6

Table 1: Summary of robustness-ratio bounds. Each entry also gives the subsection and theorem reference.

In the weak adversary model, the scheduler has more flexibility because it can choose which machines are treated as failed. This yields the upper and lower bounds in Table 1 for both arbitrary k and $k = 1$.

1.2 Other Related Work

Buchem et al. [2024] introduced a stochastic scheduling problem with a stochastic number of machines. In their model, a probability distribution over the number of machines is given, and the goal is to minimize the expected makespan. They showed that the problem is NP-hard and presented a polynomial-time approximation scheme (PTAS). Epstein and Levin [2025] later improved the running time and obtained an efficient polynomial-time approximation scheme (EPTAS).

After a machine failure, reassignment reduces to scheduling jobs on machines that already have load; equivalently, machines can process newly assigned jobs only after their current work is completed. This setting is known as *scheduling with non-simultaneous machines*. Lee [1991] presented a $4/3$-approximation algorithm for the problem.

Our problem also falls under *Recoverable Robust Optimization* [Liebchen *et al.*, 2009; Buchheim and Kurtz, 2017], where one seeks an initial solution that remains good after a limited recovery step. Commonly studied settings include Knapsack [Büsing *et al.*, 2011], Network Flow [Bertsimas *et al.*, 2013], Scheduling [Bold and Goerigk, 2025], and Matroid Basis [Hommelsheim *et al.*, 2023].

2 Preliminaries

Notation. The *workload* of a job set is the sum of its processing times. For a job set $\mathcal{J}'$, let $p(\mathcal{J}') := \sum_{j \in \mathcal{J}'} p_j$ denote its total workload. Given a schedule S, the *workload of a machine* is the total processing time assigned to that machine. If no job is assigned to a machine, we say it is *idle*. The *makespan* of a schedule S is the maximum workload over all

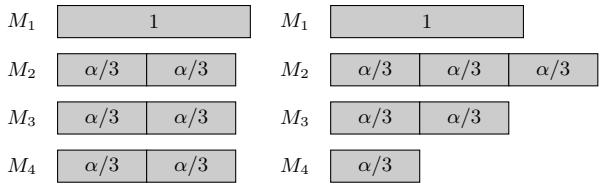

Figure 2: Two possible initial schedules of the instance.

machines. For each $1 \leq i \leq m$, let OPT_i be the optimal makespan when we use only i machines to complete all jobs, and let S_i be a schedule attaining OPT_i. Throughout this paper, we assume that $m \geq 3$; the remaining $m \in \{1, 2\}$ cases are degenerate for reassignment and can be handled separately.

Classical Scheduling Algorithms. The makespan minimization problem on m identical machines is NP-hard [Garey and Johnson, 1979]. For any $\varepsilon > 0$, we can compute a schedule with a makespan of at most $(1+\varepsilon)\text{OPT}_i$ in $\mathcal{O}(\text{poly}_{1/\varepsilon}(n))$ time (see, e.g., [Hochbaum and Shmoys, 1988]).

The *list scheduling* algorithm processes jobs in an arbitrary order and assigns each job to the currently least-loaded machine. If it runs on i machines and j is the last completed job, then the resulting makespan is at most $\text{OPT}_i + p_j$ since before the start time of job j, all machines are busy.

3 The Strong Adversary Model

In the strong adversary model, the algorithm first assigns each job to a machine. The adversary then fails a subset of machines $\mathcal{M}' \subset \mathcal{M}$ with $|\mathcal{M}'| \leq k$, and the algorithm reassigns the jobs on $\mathcal{M}'$ to the available machines $\mathcal{M} \setminus \mathcal{M}'$.

We give a general lower bound and algorithms with robustness ratios strictly below 2. For $k = 1$, our algorithm matches the lower bound and is best possible.

3.1 The General Case with Arbitrary k

Let $\alpha \approx 1.387$ be the positive root of the equation $3x^2 - 2x - 3 = 0$. We first show a lower bound α on the robustness ratio for any algorithm; the same construction already works for $k = 1$ and thus for every $k \geq 1$.

Theorem 1. *Under the strong adversary model, no algorithm can achieve a robustness ratio better than $\alpha \approx 1.387$, where α is the positive root of the equation $3x^2 - 2x - 3 = 0$.*

Proof. Consider an instance with $m = 4$ machines, $n = 7$ jobs, and $k = 1$. The processing times are as follows. $p_1 = 1$ and $p_j = \alpha/3$ for $2 \leq j \leq 7$. Observe that $\text{OPT}_4 = 1$ and $\text{OPT}_3 = \alpha$. To achieve a robustness ratio better than α, the makespan of the initial schedule must be at most α. Hence, in the initial schedule, job 1 must be the only job assigned to some machine, say, machine 1. Otherwise, the makespan of the initial schedule is at least $1 + \alpha/3 > \alpha$. If the initial makespan is already at least α, then the lower bound follows. Thus, to beat α, the six jobs of size $\alpha/3$ must be distributed over the remaining three machines with load strictly below α on each machine, which leaves only the two cases below. See Figure 2 for an illustration.

- Job 1 is assigned to machine 1, and each of the other three machines has two jobs of processing time $\alpha/3$. In this case, if machine 1 fails, job 1 must be reassigned to another machine. However, the workload of any other machine is $2\alpha/3$ in the initial assignment. Thus the makespan after any reassignment is $1 + 2\alpha/3$, and the robustness ratio is at least $(1 + 2\alpha/3)/\alpha = \alpha$.
- Some machine has at least three jobs of processing time $\alpha/3$. In this case, the initial makespan is at least α, so the robustness ratio is at least α.

Thus no initial assignment can achieve a robustness ratio better than α. □

Next, we present a ϕ-robust algorithm, where ϕ is the positive root of $x^2 - x - 1 = 0$, i.e., the golden ratio $(1 + \sqrt{5})/2 \approx 1.618$. The ratio ϕ results from balancing the performance of our algorithm across two worst-case scenarios. We summarize our algorithm for computing an initial schedule in Algorithm 1, and then explain how to reassign jobs when machines fail.

Theorem 2. *Under the strong adversary model, there is a ϕ-robust algorithm for arbitrary k, where $\phi = (1 + \sqrt{5})/2 \approx 1.618$.*

Proof. For each $i = 1, \ldots, m$, we compute an optimal schedule S_i on i machines. By scaling, assume without loss of generality that $\text{OPT}_m = 1$. We use the monotonicity of the optimum throughout: if $r \leq s$, then $\text{OPT}_r \geq \text{OPT}_s$.

If $\text{OPT}_i \leq \phi$ for all $i = m - k, \ldots, m$, then we use S_{m-k} as the initial schedule, leaving k machines idle. Since the adversary can only fail up to k machines and we have k idle machines, each failed non-idle machine can be matched to a distinct available idle machine, and we reassign all its jobs there. In this way, we obtain a new schedule without increasing the makespan.

Otherwise, by monotonicity and $\text{OPT}_m = 1 \leq \phi$, there exists some $j \in \{m - k + 1, \ldots, m\}$ such that $\text{OPT}_j \leq \phi < \text{OPT}_{j-1}$. In this case, we use S_j as the initial schedule, leaving $m - j$ machines idle. We show that S_j is ϕ-robust. Let $k' \leq k$ be the number of failed machines. Let $\text{ALG}_{m-k'}$ denote the makespan after applying our reassignment rule when k' machines fail. We have the following two cases.

Case 1: $k' \leq m - j$**.** In this case, every failed non-idle machine can be paired with a distinct idle machine that remains available, so we reassign its jobs there without increasing the makespan. Hence, using $\text{OPT}_{m-k'} \geq \text{OPT}_m = 1$, $\text{ALG}_{m-k'} \leq \phi \leq \phi \cdot \text{OPT}_{m-k'}$.

Case 2: $k' > m - j$**.** In this case, $\text{OPT}_{m-k'} > \phi$. We reassign the jobs on failed machines to available machines using list scheduling. Since $\text{OPT}_m = 1$ implies that every job has processing time at most 1, the makespan of the resulting schedule is at most $\text{OPT}_{m-k'} + 1$. Hence, $\text{ALG}_{m-k'}/\text{OPT}_{m-k'} = 1 + 1/\text{OPT}_{m-k'} < 1 + 1/\phi = \phi$.

We conclude that our initial schedule S_j is ϕ-robust.

□

Algorithm 1 ϕ-robust algorithm for arbitrary k under the strong adversary model

Input: m machines, n jobs, and the parameter k
Output: An initial schedule S
for $i = 1$ to m **do**
  Compute optimal schedule S_i on i machines; denote its makespan OPT_i
end for
if $\text{OPT}_i \le \phi$ for all i in $\{m-k, \dots, m\}$ **then**
  return S_{m-k} (leave k machines idle)
else
  Find index j such that $\text{OPT}_j \le \phi < \text{OPT}_{j-1}$
  return S_j (leave $m-j$ machines idle)
end if

3.2 The Special Case with $k = 1$

For the special case with $k = 1$, we achieve a best-possible 1.387-robust algorithm, matching the lower bound in Theorem 1.

Theorem 3. *Under the strong adversary model, there is an $\alpha \approx 1.387$-robust algorithm for $k = 1$.*

Proof. By scaling, we assume $\text{OPT}_m = 1$.

Easy cases. If $\text{OPT}_{m-1} \le \alpha$, then we use S_{m-1} as the initial schedule, leaving one machine idle. No matter which machine fails, we reassign its jobs to the idle machine (or the failed machine is the idle machine), and the makespan remains $\text{OPT}_{m-1} \le \alpha$. Hence, the robustness ratio is at most α.

If $\text{OPT}_{m-1} \ge 2/\alpha$, then we use S_m as the initial schedule. No matter which machine fails, we reassign its jobs to an available machine, incurring a makespan of at most $\text{OPT}_m + 1 = 2 \le \alpha\text{OPT}_{m-1}$. Hence, the robustness ratio is at most α. The remaining analysis focuses on the case with $\alpha < \text{OPT}_{m-1} < 2/\alpha$.

A light machine. Consider the schedule S_m. Let L_i be the workload of machine i in S_m. Suppose that there exists some machine i with $L_i \le \alpha^2 - 1$. Then we use S_m as the initial schedule. If machine $i' \ne i$ fails, we can reassign the jobs on machine i' to machine i, and the resulting makespan is at most $1 + (\alpha^2 - 1) = \alpha^2 < \alpha\text{OPT}_{m-1}$. If machine i fails, we can reassign the jobs on machine i to any other machine $i' \ne i$; the resulting makespan is at most $1 + (\alpha^2 - 1) < \alpha\text{OPT}_{m-1}$. Hence, the robustness ratio is at most α. In the remaining analysis, we consider the case where $L_i > \alpha^2 - 1$ for all machines i.

Let $\mathcal{J}_i$ be the set of jobs assigned to machine i in S_m. Suppose that for some machine i, there exists a subset of jobs $\mathcal{J}_i' \subseteq \mathcal{J}_i$ with $p(\mathcal{J}_i') \in [2-\alpha, \alpha^2 - 1]$. Then we modify S_m as follows: move the jobs in $\mathcal{J}_i \setminus \mathcal{J}_i'$ to another machine $i' \ne i$. Note that after this, the workload of machine i is in the interval $[2-\alpha, \alpha^2-1]$, and the workload of machine i' is at most $1 + (1 - (2-\alpha)) = \alpha$. We show that the robustness ratio of the new schedule is at most α.

If no machine fails, the makespan is at most α. If machine i fails, we can reassign the jobs on machine i to any other machine $i'' \notin \{i, i'\}$, and the resulting makespan is at most $1 + (\alpha^2 - 1) < \alpha\text{OPT}_{m-1}$. If machine $i'' \notin \{i, i'\}$ fails, we reassign the jobs on machine i'' to machine i, incurring a makespan of at most $1 + (\alpha^2 - 1) < \alpha\text{OPT}_{m-1}$. If machine i' fails, we reassign the jobs on machine i' as follows. We send the jobs moved from machine i to i' (with total workload at most $1 - (2-\alpha)$) to any machine $i'' \notin \{i, i'\}$, and send the other jobs on i' (with total workload at most 1) to machine i. After the reassignment, the workload of machine i is at most $\alpha^2 - 1 + 1 = \alpha^2 < \alpha\text{OPT}_{m-1}$, and the workload of machine i'' is at most $1 + (1 - (2-\alpha)) = \alpha$. Hence, the robustness ratio is at most α.

Algorithm 2 α-robust algorithm for $k = 1$ under the strong adversary model

Input: n jobs, m machines, and the parameter $k = 1$
Output: An initial schedule S
Compute optimal schedules S_{m-1} and S_m. Let L_i be the workload of machine i in S_m and $\mathcal{J}_i$ be the set of jobs assigned to machine i in S_m.
if $\text{OPT}_{m-1} \le \alpha$ **then**
  return S_{m-1} (with one idle machine).
else if $\text{OPT}_{m-1} \ge 2/\alpha$ or $\exists i, L_i \le \alpha^2 - 1$ **then**
  return S_m.
else if $\exists i, \mathcal{J}' \subseteq \mathcal{J}_i$ such that $p(\mathcal{J}') \in [2-\alpha, \alpha^2 - 1]$ **then**
  return Modified schedule from S_m by moving $\mathcal{J}_i \setminus \mathcal{J}'$ to machine $i' \ne i$.
else
  Let a_1, a_2, a_3 be the three smallest medium jobs (set $a_3 := \infty$ if fewer than three exist), and let b_1 be the smallest large job.
  if $a_1 + a_2 + a_3 \le a_2 + b_1$ or $a_1 + a_2 \le \alpha^2 - 1$ **then**
    return Modified schedule from S_m by swapping some jobs between machines of a_1 and a_2.
  else
    return Modified schedule from S_m by moving small jobs from the machine with b_1 to any other machine with two medium jobs.
  end if
end if

Structural case. It remains to consider cases with the following properties:

(a) $\alpha < \text{OPT}_{m-1} < 2/\alpha$;

(b) $L_i > \alpha^2 - 1$ for all machines i;

(c) In S_m, for every machine i, there is no subset of jobs $\mathcal{J}_i' \subseteq \mathcal{J}_i$ with $p(\mathcal{J}_i') \in [2-\alpha, \alpha^2 - 1]$.

To explore more structural properties, we classify jobs by their processing times into the following three categories. We say a job j is:

- small if $p_j < 2 - \alpha^2$;
- medium if $\alpha^2 + \alpha - 3 < p_j < 2 - \alpha$;
- large if $p_j > \alpha^2 - 1$.

Here $2 - \alpha^2 < \alpha^2 + \alpha - 3$, so the second forbidden interval in the claim below is nonempty; this follows from $3\alpha^2 - 2\alpha - 3 = 0$. See Figure 3 for an illustration. We first record the

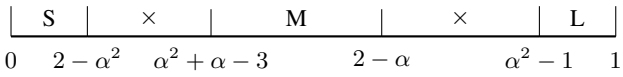

Figure 3: Job classification by size. "S", "M" and "L" correspond to small, medium, and large jobs. × means no jobs in these intervals.

following claim, which shows that these three classes cover all jobs.

Claim. Under Properties (a)–(c), every job must be either small, medium, or large; i.e., no job has processing time in the forbidden intervals

$$[2-\alpha, \alpha^2-1] \quad \text{or} \quad [2-\alpha^2, \alpha^2+\alpha-3].$$

Proof. Property (c) already forbids any job in $[2-\alpha, \alpha^2-1]$. Suppose, for contradiction, that some job j on machine i satisfies $p_j \in [2-\alpha^2,\ \alpha^2+\alpha-3]$. Then the subset $\mathcal{J}_i \setminus \{j\}$ has a workload of

$$p(\mathcal{J}_i) - p_j \ \geq\ (\alpha^2-1) - (\alpha^2+\alpha-3) \ =\ 2-\alpha,$$

and

$$p(\mathcal{J}_i) - p_j \ \leq\ 1-(2-\alpha^2) \ =\ \alpha^2-1.$$

Hence $p(\mathcal{J}_i \setminus \{j\}) \in [2-\alpha, \alpha^2-1]$, violating Property (c). ◁

Under Properties (a)–(c), each machine in S_m has either one large job and only small jobs, or two medium jobs and only small jobs. Indeed, a large and a medium job already exceed load 1; three medium jobs contain two jobs with total load in $[2-\alpha, \alpha^2-1]$; and a prefix argument rules out machines with only small jobs or with one medium job and small jobs.

Let ℓ be the number of machines that have two medium jobs, and the other $m-\ell$ machines that have one large job. We have the following cases.

Case 1: $\ell = 0$. In this case, there are m large jobs. So in any schedule with $m-1$ machines, some machine has at least 2 large jobs, which implies $\textsc{Opt}_{m-1} > 2(\alpha^2-1)$. This violates Property (a), so this case will not happen.

Case 2: $\ell = m$. In this case, we use S_m as our initial schedule. When one machine i fails, let j_1 and j_2 be its two medium jobs. We reassign j_1 to another machine i_1 and j_2 to another machine i_2, where i, i_1, i_2 are distinct. If machine i has small jobs of total workload W, let $T = \alpha^2+\alpha-3$. Since $W < 1-2T$, we split them into two groups of workload at most T: if $W > T$, take the first prefix above $W-T$, whose load is below $W-T+(2-\alpha^2) < T$. We send one group to i_1 and the other to i_2. Therefore, each of i_1 and i_2 starts with workload at most 1, receives at most $2-\alpha$ from the failed machine and at most $\alpha^2+\alpha-3$ from the split small jobs, and thus ends with workload at most $1+(2-\alpha)+(\alpha^2+\alpha-3) = \alpha^2 < \alpha\textsc{Opt}_{m-1}$.

Case 3: $1 \leq \ell \leq m-1$. Let the processing times of the 2ℓ medium jobs be $a_1 \leq a_2 \leq \cdots \leq a_{2\ell}$ and those of the $m-\ell$ large jobs be $b_1 \leq b_2 \leq \cdots \leq b_{m-\ell}$. For simplicity, we also refer to a_i or b_i as the corresponding job.

We show another lower bound on $\textsc{Opt}_{m-1}$. Consider a schedule with $m-1$ machines. If there is some machine with 2 large jobs, then we have $\textsc{Opt}_{m-1} \geq 2(\alpha^2-1)$, which violates Property (a). Similarly, if there is a machine with one large job and 2 medium jobs, then we have $\textsc{Opt}_{m-1} \geq 2/\alpha$, violating Property (a). Hence, in any schedule with $m-1$ machines, either there is some machine that has at least 3 medium jobs or there are at least two machines, each of which has one medium job and one large job. In the former case, we have $\textsc{Opt}_{m-1} \geq a_1+a_2+a_3$, where a_3 is defined as ∞ if there are only 2 medium jobs, and in the latter we have $\textsc{Opt}_{m-1} \geq a_2+b_1$. In any case, we have

$$\textsc{Opt}_{m-1} \geq \min\{a_1+a_2+a_3, a_2+b_1\}. \tag{3}$$

We perform a case analysis on Equation (3).

Case 3.1: $a_1+a_2+a_3 \leq a_2+b_1$. Note that we will not be in this case if there are only 2 medium jobs. Since $a_3 \geq a_2 \geq a_1$, we have $\textsc{Opt}_{m-1} \geq a_1+a_2+a_3 \geq 3/2(a_1+a_2)$. We obtain a new schedule S'_m from S_m as follows. Let i_1 (resp. i_2) be the machine where a_1 (resp. a_2) is assigned. If $i_1 = i_2$, then a_1 and a_2 are already on one machine, and the same reassignment argument applies without the swap below. Thus, we assume $i_1 \neq i_2$ in the construction below. Let a'_1 (resp. a'_2) be the other medium job assigned to i_1 (resp. i_2), and $\mathcal{J}_1$ (resp. $\mathcal{J}_2$) be the set of small jobs assigned to i_1 (resp. i_2). We move a'_1 and $\mathcal{J}_1$ to i_2, and a_2 to i_1. In this way, we obtain S'_m as the initial schedule. The two modified machines end up with workloads a_1+a_2 and at most $2-(a_1+a_2)$, respectively. The first is below α because $a_1+a_2 < 2(2-\alpha) < \alpha$, and the second is below α because $a_1, a_2 > \alpha^2+\alpha-3$, so $a_1+a_2 > 2(\alpha^2+\alpha-3) > 2-\alpha$. It suffices to show that when one machine fails, we can reassign the jobs so that the makespan after reassignment is at most $\alpha\textsc{Opt}_{m-1}$.

If a third machine $i_3 \notin \{i_1, i_2\}$ fails, send its jobs to i_1, whose load is at most a_1+a_2+1. If $\frac{3}{2}(a_1+a_2) \leq \alpha$, this is at most α^2; otherwise it is below $\alpha \cdot \frac{3}{2}(a_1+a_2)$. In both cases it is at most $\alpha\textsc{Opt}_{m-1}$. If i_1 fails, we can reassign all the jobs on i_1 to some $i_3 \notin \{i_1, i_2\}$ by the same arguments as above. If i_2 fails, we reassign a'_1 and $\mathcal{J}_1$ to i_1, and reassign a'_2 and $\mathcal{J}_2$ to some $i_3 \notin \{i_1, i_2\}$. After that, the workload of i_1 is at most $1+a_2 < \alpha^2$, and the workload of i_3 is at most $1+(1-a_2) < 2-(\alpha^2+\alpha-3) < \alpha^2$; both are below $\alpha\textsc{Opt}_{m-1}$. We conclude that S'_m has a robustness ratio of α.

Case 3.2: $a_1+a_2 \leq 2\alpha/3 = \alpha^2-1$. We use the same modification as in Case 3.1 and the same analysis, using the fact that $1+a_1+a_2 \leq 1+\alpha^2-1 = \alpha^2 \leq \alpha\textsc{Opt}_{m-1}$.

Case 3.3: $a_1+a_2+a_3 > a_2+b_1$ and $a_1+a_2 > 2\alpha/3$. Note that $a_2 \geq a_1$, which implies that $a_2 > \alpha/3$. We obtain a new schedule S'_m from S_m as follows. Let i be the machine where b_1 is assigned and $\mathcal{J}$ be the set of small jobs on i. We move $\mathcal{J}$ to some other machine $i' \neq i$ that has two medium jobs. In this way, we obtain S'_m. If no machine fails, the makespan of S'_m is at most $\max\{1, 2-b_1\} < \alpha$, since $b_1 > \alpha^2-1$. If $i'' \notin \{i, i'\}$ fails, we reassign all its jobs to i, increasing its workload to $1+b_1$. Since $a_2 > \alpha/3$ and $b_1 > \alpha^2-1 = 2\alpha/3$, we have $\alpha(a_2+b_1)-(1+b_1) > \alpha^2/3+(\alpha-1)2\alpha/3-1 = 0$; hence $1+b_1 < \alpha(a_2+b_1) \leq \alpha\textsc{Opt}_{m-1}$. If i fails, we can reassign all its jobs to $i'' \notin \{i, i'\}$ by the same arguments as above. If i' fails, send one medium job and all small jobs on i' to i, and the other medium job to $i'' \notin \{i, i'\}$. The original

small workload of i' is at most $1-(\alpha^2-1)$, since its two medium jobs total more than $a_1+a_2 > \alpha^2-1$. The makespan after reassignment is at most $1+2-\alpha+(1-(\alpha^2-1)) < \alpha^2 \le \alpha \textsc{Opt}_{m-1}$. □

4 The Weak Adversary Model

In the weak adversary model, the adversary reveals the number of failed machines $k' \le k$ after the algorithm fixes its initial schedule. The algorithm then chooses which k' machines are treated as failed. This weaker adversary allows better robustness ratios.

4.1 The General Case with Arbitrary k

We start with a lower bound of $\gamma \approx 1.281$, where γ is the positive root of $x^2 - x/2 - 1 = 0$.

Theorem 4. *Under the weak adversary model, no algorithm can achieve a robustness ratio better than $\gamma \approx 1.281$, where γ is the positive root of $x^2 - x/2 - 1 = 0$.*

Proof. Consider an instance with $m = 6$ machines, $n = 12$ jobs, and $k = 1$. There are 6 large jobs of processing time $\gamma/2$ and 6 small jobs of processing time $1-\gamma/2$. Observe that $\textsc{Opt}_m = 1$ and $\textsc{Opt}_{m-1} = \gamma$. Indeed, a makespan of γ on five machines is feasible by placing the six large jobs as three pairs and the six small jobs as two triples; the latter have load $3(1-\gamma/2) \le \gamma$. Also, because there are six large jobs and only five machines, some machine must contain two large jobs, so $\textsc{Opt}_{m-1} \ge \gamma$ and hence $\textsc{Opt}_{m-1} = \gamma$.

To achieve a robustness ratio strictly smaller than γ, no machine can have two large jobs in the initial schedule; otherwise the makespan, and hence the robustness ratio, is at least γ. Thus each machine must contain exactly one large job. If some machine has at least two small jobs, then its workload is at least $\gamma/2 + 2(1-\gamma/2) = 2-\gamma/2 > \gamma$, which results in a robustness ratio greater than γ. Hence, the only possible initial schedule is that each machine has exactly one large job and one small job.

When one machine fails, one large job and one small job must be reassigned. Some machine receiving the large job then has workload at least $1+\gamma/2 = \gamma^2$. Since $\textsc{Opt}_{m-1} = \gamma$, the robustness ratio is $\gamma^2/\gamma = \gamma$. □

Next, we show a 1.5-robust algorithm for arbitrary k.

Theorem 5. *Under the weak adversary model, there is a 1.5-robust algorithm for arbitrary k.*

Proof. We use similar ideas as in Section 3.1 to decide an initial schedule, and then carefully select which machines to fail. Again, by scaling, we can assume that $\textsc{Opt}_m = 1$.

If $\textsc{Opt}_i \le 1.5$ for all i with $m-k \le i \le m$, we use S_{m-k} as the initial schedule and leave k machines idle. This is 1.5-robust: when failures occur, we select idle machines as the failed machines, so the makespan is unchanged.

Otherwise, there exists some $i \in \{m-k+1, \dots, m\}$ such that $\textsc{Opt}_i \le 1.5$ and $\textsc{Opt}_{i-1} > 1.5$. Then we use S_i as the initial schedule, leaving $m-i$ machines idle. We show this ensures a robustness ratio of 1.5. Let $k' \le k$ be the number of failed machines chosen by the adversary. The case $k' = 0$ is already covered by the first branch below and requires no special handling. If $k' \le m-i$, then we can select k' idle machines as the failed machines without affecting the makespan. Hence, it remains to consider the case where $m-i < k' \le k$. We first select the $m-i$ idle machines as the failed machines. It remains to choose the other $k'' := k' - (m-i)$ failed machines. Let $\textsc{Alg}_{i-k''}$ denote the makespan after applying our reassignment rule when the final number of machines is $i-k''$.

Call a job *large* if its processing time is at least 0.75 and *small* otherwise. Without loss of generality, we can assume that in S_i, the first i' machines contain a large job, the next $i-i'$ machines contain only small jobs, and the last $m-i$ machines are idle.

Case 1: $i-k'' \ge i'$. In this case, $k'' \le i-i'$, so we select any k'' machines among the $i-i'$ machines without large jobs as the failed machines. We reassign the jobs of these k'' machines to the remaining $i-k''$ machines using list scheduling. The makespan of the new schedule is at most $\textsc{Opt}_{i-k''} + 0.75$, which implies

$$\frac{\textsc{Alg}_{i-k''}}{\textsc{Opt}_{i-k''}} \le \frac{\textsc{Opt}_{i-k''} + 0.75}{\textsc{Opt}_{i-k''}} \le 1 + \frac{0.75}{1.5} = 1.5,$$

where the second inequality uses $\textsc{Opt}_{i-k''} \ge \textsc{Opt}_{i-1} > 1.5$.

Case 2: $i-k'' < i'/2$. Here the number of remaining machines $i-k''$ is so small that in any feasible schedule on $i-k''$ machines, some machine must contain at least 3 large jobs. This is just the pigeonhole principle: there are i' large jobs in total, but fewer than $i'/2$ remaining machines, so one machine must receive at least three large jobs. Hence, $\textsc{Opt}_{i-k''} \ge 3 \cdot 0.75 = 2.25$. We reassign the jobs of the failed k'' machines to the remaining $i-k''$ machines using list scheduling. The makespan of the new schedule is at most $\textsc{Opt}_{i-k''} + 1$, which implies

$$\frac{\textsc{Alg}_{i-k''}}{\textsc{Opt}_{i-k''}} \le \frac{\textsc{Opt}_{i-k''} + 1}{\textsc{Opt}_{i-k''}} \le 1 + \frac{1}{2.25} < 1.5.$$

Case 3: $i'/2 \le i-k'' < i'$. For each machine $r \in \{1, \dots, i\}$, let b_r be the processing time of the largest job on machine r. Without loss of generality, assume that $b_1 \ge b_2 \ge \cdots \ge b_i$. We keep the first $i-k''$ machines as the remaining machines, and the next k'' machines as the failed machines. Note that there are $i-k''+1$ jobs with a processing time of at least $b_{i-k''+1}$. Hence, in any feasible schedule with $i-k''$ machines, some machine has at least 2 jobs with a processing time of at least $b_{i-k''+1}$, which implies $\textsc{Opt}_{i-k''} \ge 2b_{i-k''+1}$. We reassign the jobs of the failed k'' machines to the remaining $i-k''$ machines using list scheduling, incurring a makespan of at most $\textsc{Opt}_{i-k''} + b_{i-k''+1}$, because every job on a failed machine has processing time at most $b_{i-k''+1}$. Therefore, we have

$$\frac{\textsc{Alg}_{i-k''}}{\textsc{Opt}_{i-k''}} \le \frac{\textsc{Opt}_{i-k''} + b_{i-k''+1}}{\textsc{Opt}_{i-k''}} \le 1 + \frac{b_{i-k''+1}}{2b_{i-k''+1}} = 1.5.$$

Thus the initial schedule is 1.5-robust. □

4.2 The Special Case with $k = 1$

For the special case with $k = 1$, we achieve a best-possible 1.281-robust algorithm.

Theorem 6. *Under the weak adversary model, there is a $\gamma \approx 1.281$-robust algorithm for $k = 1$.*

Proof. By scaling, assume $\textsc{Opt}_m = 1$.

We call a job j *large* if $p_j > \gamma/2 \approx 0.64$, and *small* otherwise. We first handle $m \geq 4$ and then treat $m = 3$ separately.

Case $m \geq 4$. If $\textsc{Opt}_{m-1} \leq \gamma$, then we use S_{m-1} as the initial schedule. If no machine fails, the makespan is $\textsc{Opt}_{m-1} \leq \gamma\textsc{Opt}_m$. If the adversary decides to fail one machine, we can pick the empty machine as the failed machine, without affecting the makespan. This proves that the robustness ratio is at most γ.

If $\textsc{Opt}_{m-1} > \gamma$, we use S_m as the initial schedule. Since $\textsc{Opt}_m = 1$, each machine contains at most one large job, and there are at most m large jobs. If there are exactly m large jobs, then any schedule on $m-1$ machines has some machine containing at least two large jobs. Hence, we have $\textsc{Opt}_{m-1} \geq 2b$, where b is the smallest processing time of large jobs. If the adversary decides to fail one machine, we pick the machine that has a large job j with $p_j = b$ as the failed machine. We assign job j to some machine $i' \neq i$, and the other jobs of machine i to another machine $i'' \notin \{i, i'\}$. After the reassignment, the makespan of the new schedule is at most $\max\{1+b, 1+(1-b)\} = 1+b$. Since $(1+b)/(2b) = 1/2+1/(2b) \leq 1/2+1/\gamma = \gamma$, the robustness ratio is at most γ.

If there are fewer than m large jobs, then some machine i in S_m has no large job. We treat machine i as the failed machine and partition its jobs using the following claim.

Claim. Any set of small jobs with total workload at most 1 can be split into at most 3 parts, each of workload at most $\gamma/2$.

Proof. If the total workload is at most $\gamma/2$, we are done with only one part. Otherwise, index the jobs as $1, \ldots, \ell$ and write $Z_i \coloneqq \sum_{j=1}^{i} p_j$ for each $1 \leq i \leq \ell$. If there exists $1 \leq i \leq \ell$ such that $1-\gamma/2 \leq Z_i \leq \gamma/2$, then we can partition the jobs into 2 parts, i.e., the first i jobs and the last $\ell - i$ jobs, and we are done. Otherwise, there must be some index $1 \leq i < \ell$ such that $Z_i < 1-\gamma/2$ and $Z_{i+1} > \gamma/2$. In this case, we partition the jobs into 3 parts, i.e., the first i jobs, the job $i+1$, and the last $\ell - i - 1$ jobs. The workload of the first i jobs is at most $1-\gamma/2 < \gamma/2$, and the workload of the last $\ell - i - 1$ jobs is at most $1-\gamma/2 < \gamma/2$. The workload of the job $i+1$ is at most $\gamma/2$ since there is no large job. ◁

Then we assign each part to a distinct machine, which is possible since $m \geq 4$, incurring a new makespan of at most $1+\gamma/2 = \gamma^2 \leq \gamma\textsc{Opt}_{m-1}$.

Case $m = 3$. The previous three-part argument may fail when $m = 3$. Again assume $\textsc{Opt}_{m-1} > \gamma$ and at most $m-1$ large jobs; use S_m and fix a machine without large jobs as the one that may fail. Let B be its workload. If some job has size above $2B/3$, assign this job and the remaining workload separately, giving makespan at most $1+\gamma/2 = \gamma^2 \leq \gamma\textsc{Opt}_{m-1}$. Thus every job has size at most $2B/3$, and we use the following claim.

Claim. If a job set $\mathcal{J}$ has total workload at most B and no job exceeds $2B/3$, then $\mathcal{J}$ can be split into two parts, each of workload at most $2B/3$.

Proof. If the total workload is at most $2B/3$, we are done with $\mathcal{J}_1 \coloneqq \emptyset$ and $\mathcal{J}_2 \coloneqq \mathcal{J}$. Otherwise, index the jobs as $1, \ldots, \ell$ and let $Z_i \coloneqq \sum_{j=1}^{i} p_j$ for each $1 \leq i \leq \ell$. Let i be the largest index with $Z_i \leq 2B/3$, and let $W \coloneqq Z_\ell$ be the total workload. If $W - Z_i \leq 2B/3$, then we are done with $\mathcal{J}_1 \coloneqq \{1, \ldots, i\}$ and $\mathcal{J}_2 \coloneqq \{i+1, \ldots, \ell\}$. Otherwise, we have $W - Z_i > 2B/3$, which implies $Z_i < B/3$ since $W \leq B$. By maximality of i, we have $Z_{i+1} > 2B/3$, so $p_{i+1} > 2B/3 - Z_i > B/3$. Then we set $\mathcal{J}_2 \coloneqq \{i+1\}$ and $\mathcal{J}_1 \coloneqq \mathcal{J} \setminus \{i+1\}$. We have $p(\mathcal{J}_2) = p_{i+1} \leq 2B/3$ and $p(\mathcal{J}_1) = W - p_{i+1} \leq B - B/3 = 2B/3$. ◁

Let the two resulting parts be labeled so that $p(\mathcal{J}_1) \leq p(\mathcal{J}_2)$. Note that $p(\mathcal{J}_1) \leq 0.5$ since $p(\mathcal{J}_1) \leq B/2 \leq 1/2$.

When this chosen machine fails, we reassign $\mathcal{J}_1$ and $\mathcal{J}_2$ to the remaining two machines as follows. We can assume that $p(\mathcal{J}_1) + p(\mathcal{J}_2) \geq \gamma/2$, as otherwise we can reassign both $\mathcal{J}_1$ and $\mathcal{J}_2$ to one remaining machine to ensure a makespan of at most $1+\gamma/2 = \gamma^2 \leq \gamma\textsc{Opt}_{m-1}$. If one remaining machine has workload at most $\gamma^2 - 2/3$, put $\mathcal{J}_2$ there and $\mathcal{J}_1$ on the other; their loads become at most γ^2 and $1+0.5 \leq \gamma^2$, respectively. Hence, the makespan of the new schedule is at most $\gamma^2 \leq \gamma\textsc{Opt}_{m-1}$. If both of the remaining two machines have a workload of at least $\gamma^2-2/3$, then we have $\textsc{Opt}_{m-1} \geq \gamma^2 - 2/3 + (p(\mathcal{J}_1) + p(\mathcal{J}_2))/2 = \gamma^2 - 2/3 + B/2$. In this case, we reassign $\mathcal{J}_1$ to one remaining machine and $\mathcal{J}_2$ to the other. This ensures that the makespan of the new schedule is at most $1 + p(\mathcal{J}_2) \leq 1 + 2B/3$. Set

$$f(B) \coloneqq \frac{1 + 2B/3}{\gamma^2 - 2/3 + B/2}.$$

Since $f'(B) = \frac{2\gamma^2/3 - 17/18}{(\gamma^2 - 2/3 + B/2)^2} > 0$, we have $f(B) \leq f(1) = \frac{5/3}{\gamma^2 - 1/6} < \gamma$. Thus $\frac{\textsc{Alg}_{m-1}}{\textsc{Opt}_{m-1}} < \gamma$. □

5 Conclusion

We studied robust scheduling under machine failures, where an initial schedule must be paired with reassignment rules that move only jobs on failed machines. For both the strong and weak adversary models, we gave robust algorithms and matching lower bounds for the single-failure case; for arbitrary k, we obtained ratios strictly below the baseline obtained by direct list-scheduling recovery. The main technical tradeoff is how much idle capacity to reserve for recovery while keeping the no-failure schedule close to optimal; the weak model further exploits the freedom to choose which machines are treated as failed. Open questions include designing algorithms with better robustness ratios for arbitrary k, establishing stronger lower bounds for $k \geq 2$, and finding better-than-2-robust algorithms without (approximately) computing $\textsc{Opt}_i$.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 12271477.

References

[Albers and Hellwig, 2017] Susanne Albers and Matthias Hellwig. Online makespan minimization with parallel schedules. *Algorithmica*, 78(2):492–520, 2017.

[Bertsimas *et al.*, 2013] Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. Robust and adaptive network flows. *Oper. Res.*, 61(5):1218–1242, 2013.

[Bold and Goerigk, 2025] Matthew Bold and Marc Goerigk. Recoverable robust single machine scheduling with polyhedral uncertainty. *J. Sched.*, 28(3):269–287, 2025.

[Buchem *et al.*, 2024] Moritz Buchem, Franziska Eberle, Hugo Kooki Kasuya Rosado, Kevin Schewior, and Andreas Wiese. Scheduling on a stochastic number of machines. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, London School of Economics, London, UK, August 28-30, 2024*, volume 317 of *LIPIcs*, pages 14:1–14:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Buchheim and Kurtz, 2017] Christoph Buchheim and Jannis Kurtz. Min-max-min robust combinatorial optimization. *Math. Program.*, 163(1-2):1–23, 2017.

[Büsing *et al.*, 2011] Christina Büsing, Arie M. C. A. Koster, and Manuel Kutschka. Recoverable robust knapsacks: the discrete scenario case. *Optim. Lett.*, 5(3):379–392, 2011.

[Eberle *et al.*, 2023] Franziska Eberle, Ruben Hoeksma, Nicole Megow, Lukas Nölke, Kevin Schewior, and Bertrand Simon. Speed-robust scheduling: sand, bricks, and rocks. *Math. Program.*, 197(2):1009–1048, 2023.

[Epstein and Levin, 2025] Leah Epstein and Asaf Levin. Efficient approximation schemes for scheduling on a stochastic number of machines. In Olaf Beyersdorff, Michal Pilipczuk, Elaine Pimentel, and Kim Thang Nguyen, editors, *42nd International Symposium on Theoretical Aspects of Computer Science, STACS 2025, Jena, Germany, March 4-7, 2025*, volume 327 of *LIPIcs*, pages 31:1–31:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[Garey and Johnson, 1979] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Hochbaum and Shmoys, 1988] Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17(3):539–551, 1988.

[Hommelsheim *et al.*, 2023] Felix Hommelsheim, Nicole Megow, Komal Muluk, and Britta Peis. Recoverable robust optimization with commitment. *CoRR*, abs/2306.08546, 2023.

[Lee, 1991] Chung-Yee Lee. Parallel machines scheduling with nonsimultaneous machine available time. *Discret. Appl. Math.*, 30(1):53–61, 1991.

[Liebchen *et al.*, 2009] Christian Liebchen, Marco E. Lübbecke, Rolf H. Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*, volume 5868 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2009.

[Lindermayr and Megow, 2025] Alexander Lindermayr and Nicole Megow. Permutation predictions for non-clairvoyant scheduling. *ACM Trans. Parallel Comput.*, 12(2):4:1–4:26, 2025.

[Lusby *et al.*, 2018] Richard Martin Lusby, Jesper Larsen, and Simon Bull. A survey on robustness in railway planning. *Eur. J. Oper. Res.*, 266(1):1–15, 2018.

[Minarík and Sgall, 2024] Josef Minarík and Jirí Sgall. Speed-robust scheduling revisited. In Amit Kumar and Noga Ron-Zewi, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, London School of Economics, London, UK, August 28-30, 2024*, volume 317 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Motwani *et al.*, 1994] Rajeev Motwani, Steven J. Phillips, and Eric Torng. Non-clairvoyant scheduling. *Theor. Comput. Sci.*, 130(1):17–47, 1994.

[Niño-Mora, 2009] J. Niño-Mora. Stochastic scheduling. In *Encyclopedia of Optimization*, pages 3818–3824. Springer, 2009.

[Rani *et al.*, 2025] S. Sheeja Rani, Oruba Alfawaz, and Ahmed M. Khedr. A robust fault-tolerant framework for VM failure predication and efficient task scheduling in dynamic cloud environments. *J. Netw. Comput. Appl.*, 244:104340, 2025.

[Stein and Zhong, 2020] Clifford Stein and Mingxian Zhong. Scheduling when you do not know the number of machines. *ACM Trans. Algorithms*, 16(1):9:1–9:20, 2020.