Restricted Adaptivity in Stochastic Scheduling

Guillaume Sagnol
Institut für Mathematik, TU Berlin, Germany

Daniel Schmidt genannt Waldschmidt
Institut für Mathematik, TU Berlin, Germany

Abstract

We consider the stochastic scheduling problem of minimizing the expected makespan on m parallel identical machines. While the (adaptive) list scheduling policy achieves an approximation ratio of 2, any (non-adaptive) fixed assignment policy has performance guarantee $\Omega\left(\frac{\log m}{\log\log m}\right)$. Although the performance of the latter class of policies are worse, there are applications in which non-adaptive policies are desired. In this work, we introduce the two classes of δ-delay and τ-shift policies whose degree of adaptivity can be controlled by a parameter. We present a policy – belonging to both classes – which is an $\mathcal{O}(\log\log m)$-approximation for reasonably bounded parameters. In other words, an exponential improvement on the performance of any fixed assignment policy can be achieved when allowing a small degree of adaptivity. Moreover, we provide a matching lower bound for any δ-delay and τ-shift policy when both parameters, respectively, are in the order of the expected makespan of an optimal non-anticipatory policy.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases stochastic scheduling, makespan minimzation, approximation algorithm, fixed assignment policy, non-anticipatory policy

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.79

Related Version *Full Version*: `https://arxiv.org/abs/2106.15393` [30]

Funding Guillaume Sagnol and Daniel Schmidt genannt Waldschmidt were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project ID: 390685689).

Acknowledgements We thank Thibault Juillard for helpful discussions on the topic of this paper. We also thank the anonymous referees for helpful comments.

1 Introduction

Load balancing problems are one of the most fundamental problems in the field of scheduling, with applications in various sectors such as manufacturing, construction, communication or operating systems. The common challenge is the search for an efficient allocation of scarce resources to a number of tasks. While many variants of the problem are already hard to solve, in addition one may have to face uncertainty regarding the duration of the tasks; one way to model this is to use stochastic information learned from the available data.

In contrast to the solution concept of a schedule in deterministic problems, we are concerned with *non-anticipatory policies* in stochastic scheduling problems. Such a policy has the ability to *react* to the information observed so far. While this adaptivity can be very powerful, there are situations where assigning resources to jobs prior to their execution is a highly desired feature, e.g. for the scheduling of healthcare services. This is especially true for the daily planning of elective surgery units in hospitals, where a sequence of patients is typically set in advance for each operating room. In this work, we present and analyze semi-adaptive policies, which allow one to control the level of adaptivity of the policy.

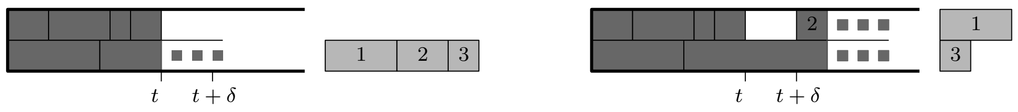

Figure 1 Snippets of the execution of a δ-delay policy: Realizations of jobs observed up to time t (left) and up to some time $> t+\delta$ (right) are depicted by rectangles in dark grey; the running job non-completed by the time of each snippet is indicated by squared dots in dark grey; jobs that did not start yet are depicted in light grey with the corresponding machine assignment.

The problem considered in this paper is the stochastic counterpart of the problem of minimizing the makespan on parallel identical machines, denoted by $P\,||\,\mathbb{E}[C_{\max}]$ using the three field notation due to Graham, Lawler, Lenstra and Rinnooy Kan [13]. The input consists of a set of n jobs $\mathcal{J}$ and a set of m parallel identical machines $\mathcal{M}$. Each job $j \in \mathcal{J}$ is associated with a non-negative random variable P_j representing the processing time of the job. The processing times are assumed to be (mutually) independent and to have finite expectation. In this work, it is sufficient to only know the expected processing times.

Roughly speaking, a non-anticipatory policy may, at any point in time t, decide to start a job on an idle machine or to wait until a later decision time. However, it may not anticipate any future information of the realizations, i.e., it may only make decisions based on the information observed up to time t. For further details we refer to the work by Möhring, Radermacher and Weiss [26]. The task is to find a non-anticipatory policy minimizing the expected makespan $\mathbb{E}[C_{\max}] := \mathbb{E}[\max_{j\in\mathcal{J}} C_j]$, where C_j denotes the (random) completion time of job j under the considered policy. An optimal policy is denoted by OPT. By slight abuse of notation we use Π for both the policy and the expected makespan of the policy.

An alternative way of understanding non-anticipatory policies is that they maintain a queue of jobs for every machine. At any point in time t it may start the first job in the queue of a machine if it is idle or it may change the queues arbitrarily, using only the information observed up to time t. In this work we consider a policy to be adaptive if it has the ability to react to the observations by changing the queues arbitrarily. The important class of non-idling non-adaptive policies is called the class of fixed assignment policies. Such a policy assigns all jobs to the machines beforehand, in form of ordered lists, and each machine processes the corresponding jobs as early as possible in this order.

While the class of (fully adaptive) non-anticipatory policies and the class of (non-adaptive) fixed assignment policies can be considered as two extremes, the purpose of this paper is to introduce two classes of policies bridging the gap between them continuously.

▶ **Definition 1.1** (δ-delay and τ-shift policies). *A δ-delay policy for $\delta > 0$ is a non-anticipatory policy which starts with a fixed assignment of all jobs to the machines and which may, at any point in time t, reassign not-started jobs to other machines with a delay of δ: the reassigned jobs are not allowed to start before time $t+\delta$.*
A τ-shift policy for $\tau > 0$ is a non-anticipatory policy which starts with a fixed assignment of all jobs to the machines and which may reassign jobs to other machines, but only at times that are an integer multiple of τ.

Snippets of the execution of a δ-delay policy and a τ-shift policy can be found in Figure 1 and Figure 2, respectively. Observe that we recover the class of fixed assignment policies by letting δ or τ go to ∞, and the class of non-anticipatory in the limit when δ or τ goes to 0.

Related Work. Minimizing the makespan on parallel identical machines is a fundamental deterministic scheduling problem which dates back to the 60s. Graham [11] showed that the list scheduling algorithm computes a solution which is within a factor of $\left(2 - \frac{1}{m}\right)$ away from

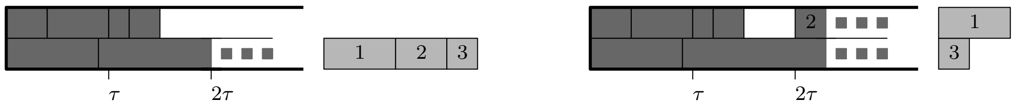

Figure 2 Snippets of the execution of a τ-shift policy: Realizations of jobs observed up to time 2τ (left) and some time $> 2\tau$ (right) are depicted by rectangles in dark grey; the running job non-completed by the time of each snippet is indicated by squared dots in dark grey; jobs that did not start yet are depicted in light grey with the corresponding machine assignment.

an optimal solution. When the jobs are arranged in LPT-order, i.e., in non-increasing order of their processing times, he showed that list scheduling gives a $\left(\frac{4}{3} - \frac{1}{3m}\right)$-approximation [12]. While $Pm||C_{\max}$, where the number of machines m is constant, and $P||C_{\max}$ are (weakly) and strongly NP-complete [9], respectively, Sahni [32] and Hochbaum and Shmoys [18] obtained a FPTAS and a PTAS, respectively. In subsequent work [3, 5, 17, 21, 22] the running time of the PTAS was improved. More general machine environments were also considered in the literature [19, 24].

The stochastic counterpart $P||\mathbb{E}[C_{\max}]$ where the processing times of the jobs are random and the objective is to minimize the expected makespan has also attracted attention. One can easily see that the list scheduling algorithm by Graham [11] also yields a 2-approximation compared to an optimal non-anticipatory policy for the stochastic problem, as its analysis can be carried over to any realization. While list scheduling can be considered as a very adaptive policy, some applications require rather restricted policies, e.g. when scheduling operating rooms at a hospital [7, 36]. A class of non-adaptive policies analyzed in the literature is comprised of fixed assignment policies, in which jobs must be assigned to the machines beforehand. Although more applicable, it is well known that the performance guarantee of an optimal fixed assignment is at least of the order $\Omega\left(\frac{\log m}{\log\log m}\right)$ with respect to an optimal non-anticipatory policy; see [14]. Much work was done in designing fixed assignment policies that are within a constant factor of an optimal fixed assignment policy. Kleinberg, Rabani and Tardos [23] obtain a constant factor approximation for this problem for general probability distributions. When the processing times are exponentially and Poisson distributed, PTASes were found [10, 6]. For the more general problem of makespan minimization on unrelated machines, Gupta, Kumar, Nagarajan and Shen [14] obtained a constant factor approximation. Closely related to the makespan objective, Molinaro [28] obtained a constant factor approximation for the ℓ_p-norm objective. In contrast to the literature for minimizing the makespan where approximative results were compared to an optimal fixed assignment policy, much work on the min-sum objective was done for designing approximative policies compared to an optimal non-anticipatory policy [27, 25, 34, 35, 15]. When minimizing the sum of weighted completion times, Skutella, Sviridenko and Uetz [35] showed that the performance ratio of an optimal fixed assignment policy compared to an optimal non-anticipatory policy can be as large as $\Omega(\Delta)$, where Δ is an upper bound on the squared coefficient of variation of the random variables. Lastly, Sagnol, Schmidt genannt Waldschmidt and Tesch [31] considered the extensible bin packing objective, for which they showed that the fixed assignment policy induced by the LEPT order has a tight approximation ratio of $1 + e^{-1}$ with respect to an optimal non-anticipatory policy.

Closely related to the reassignment of jobs in δ-delay and τ-shift policies, various non-preemptive scheduling problems with migration were considered in offline and online settings. Aggarwal, Motwani and Zhu [1] examined the offline problem where one must perform budgeted migration to improve a given schedule. For online makespan minimization on parallel machines, different variants on limited migration, e.g. bounds on the processing volume [33] or bounds on the number of jobs [2], were studied. Another related online problem was considered by Englert, Ozmen and Westermann [8] where a reordering buffer can be used to defer the assignment of a limited number of jobs.

One source of motivation for this research is the aforementioned application to surgery scheduling. In this domain, a central problem is the allocation of patients to operating rooms. Although additional resource constraints exist, the core of the problem can be modeled as the allocation of jobs with stochastic durations to parallel machines [7]. In this field, committing to a fixed assignment policy is common practice in order to simplify staff management and reduce the stress level in the operating theatre [4, 29, 36]. Another obstacle to the introduction of sophisticated adaptive policies is the reluctance of computer-assisted scheduling systems among practitioners [20]. That being said, it is clear that resource reallocations do occasionally occur in operating rooms to deal with unforeseen events, hence, giving a reason to study some kind of semi-adaptive model. The proposed model of δ-delay is an attempt to take into account the organizational overhead associated with rescheduling decisions; the model of τ-shift policy by the fact that rescheduling decisions cannot be made at any point in time, but must be agreed upon in short meetings between the OR manager and the medical team. Moreover, we point out that the class of τ-shift policies encompasses the popular class of *proactive-reactive policies* used for the more general resource constrained project scheduling problem [16], in which a baseline schedule can be reoptimized after a set of predetermined decision points (these approaches typically consider a penalty in the objective function to account for deviations between the initial baseline schedule and the reoptimized ones).

Our Contribution. We introduce and analyze two new classes of policies (δ-delay and τ-shift policies) that interpolate between the two extremes of non-adaptive and adaptive policies. For the stochastic problem of minimizing the expected makespan on m parallel identical machines, we analyze the policy $\mathrm{LEPT}_{\delta,\alpha}$, which belongs to the intersection of both classes. This policy can in fact be seen as a generalization of the list policy LEPT, which waits for predefined periods of time before reassigning the non-yet started jobs, taking the delay of δ into account. While an optimal fixed assignment policy has performance guarantee of at least $\Omega\left(\frac{\log m}{\log\log m}\right)$ compared to an optimal non-anticipatory policy, we show that $\mathrm{LEPT}_{\delta,\alpha}$ is an $\mathcal{O}(\log\log m)$-approximation for some constant $\alpha > 0$ and all $\delta = \mathcal{O}(1)\cdot \mathrm{OPT}$. Therefore, we exponentially improve the performance of non-adaptive policies by allowing a small amount of adaptivity. Moreover, we provide a matching lower bound for δ-delay policies as well as for τ-shift policies if δ or τ are in $\Theta(\mathrm{OPT})$. This shows that there is no δ-delay or τ-shift policy beating the approximation ratio of $\mathrm{LEPT}_{\delta,\alpha}$ by more than a constant factor.

Organization. Section 2 is devoted for the upper bound on the performance guarantee of $\mathrm{LEPT}_{\delta,\alpha}$. A lower bound on optimal δ-delay policies as well as τ-shift policies is given in Section 3. At the end, we conclude and give possible future research directions. Useful results from probability theory and detailed proofs can be found in the appendix of the full version of this paper [30].

2 Upper Bound

In this section, we show that there exists $\alpha > 1$ such that the policy $\mathrm{LEPT}_{\delta,\alpha}$ (see Definition 2.5) has a performance guarantee doubly logarithmic in m if $\delta = \mathcal{O}(1)\cdot \mathrm{OPT}$.

▶ **Theorem 2.1.** *There exists $\alpha > 1$ such that LEPT$_{\delta,\alpha}$ is an $\mathcal{O}(\log\log m)$-approximation for $\delta = \kappa \cdot OPT$ for any constant $\kappa > 0$.*

In the following, we show Theorem 2.1 for $\alpha = 33$. We note that we did not optimize the constants appearing in our calculation as our lower bound shows that $\log\log(m)$ is the correct order. Notice that it suffices to show the performance guarantee for m large enough as for $m = \mathcal{O}(1)$ the trivial policy assigning all jobs to a single machine is a constant factor approximation. To prove the main theorem, we proceed as follows: First, we define and discuss properties of the fixed assignment policy FLEPT as it lies at the heart of our policy called LEPT$_{\delta,\alpha}$. After we give the formal definition of LEPT$_{\delta,\alpha}$, we derive lower bounds on OPT needed to show its performance guarantee. The remaining part is devoted to show Theorem 2.1. The main idea of the proof is that the policy works over a sequence of reassignment periods; at the beginning of each period, there is a constant fraction of available machines with high probability. This can be used to show the following squaring effect: if the remaining volume of non-started jobs is $\epsilon \cdot m \cdot \text{OPT}$ in a period, it will be at most $\epsilon^2 \cdot m \cdot \text{OPT}$ in the next period, with high probability.

Recall that the List Scheduling algorithm due to Graham [11] with respect to a list of all jobs schedules the next job in the list on the next idle machine. Let us define the fixed assignment policy induced by list scheduling in LEPT order.

▶ **Definition 2.2** (The fixed assignment policy FLEPT)**.** *Let all jobs be arranged in non-increasing order of their expected processing times. FLEPT is the fixed assignment policy that assigns the jobs in this order to the same machines as List Scheduling would yield for the deterministic instance in which the processing times are replaced by their expected value.*

As shown by Sagnol, Schmidt genannt Waldschmidt and Tesch [31], FLEPT admits bounds on the expected load of any machine captured in the next lemma.

▶ **Lemma 2.3** ([31]: Section 3, Lemma 3)**.** *Given an assignment of jobs to machines induced by FLEPT, let ℓ_i denote the expected load of machine i, i.e., the sum of expected processing times of the jobs assigned to i. Moreover, let n_i denote the number of jobs assigned to i and let $\ell := \min_{i\in\mathcal{M}} \ell_i$. Then, for all $i \in \mathcal{M}$ we have $\ell \le \ell_i \le \frac{n_i}{n_i-1}\ell$, where $\frac{n_i}{n_i-1} = \frac{1}{0} := +\infty$ whenever $n_i = 1$.*

We immediately obtain by Lemma 2.3 the following structure on FLEPT.

▶ **Corollary 2.4.** *Given an assignment of jobs to machines induced by FLEPT, we can partition the set of machines into two types of machines: Either there is only a single job assigned to a machine or the expected load of a machine is bounded by 2ℓ. Moreover, ℓ can be bounded from above by the averaged expected load. In particular, if x denotes the total (remaining) expected load and m' is a lower bound on the total number of machines m, then $2\ell \le 2 \cdot \frac{x}{m'}$.*

Corollary 2.4 will play a central role in showing Theorem 2.1 as FLEPT constitutes an essential part of LEPT$_{\delta,\alpha}$, which we define now.

▶ **Definition 2.5** (Policy LEPT$_{\delta,\alpha}$)**.** *Let $\delta, \alpha > 0$, $k^* := \left\lfloor \log_2\left(\frac{2}{3}(\log_2(m)) + 1\right)\right\rfloor + 2$ and let $T := 2 \cdot \max\{\frac{1}{m}\sum_{j\in\mathcal{J}} \mathbb{E}[P_j], \max_j \mathbb{E}[P_j]\}$. Moreover, let $\tau_k := k(\delta + \alpha T)$ for $k \in [k^*+1]$. At the beginning the jobs are assigned according to FLEPT. For $k = 1, \dots, k^*+1$, LEPT$_{\delta,\alpha}$ reassigns the jobs that have not started yet before τ_k to the machines that have processed all jobs assigned at previous iterations $0, \dots, k-1$ by time $\tau_1, \dots, \tau_k$, respectively, according to FLEPT. The reassigned jobs may start at time $\tau_k + \delta$ at the earliest.*

We note that in practice it makes sense to use all available machines at each iteration instead of the machines that were available in each previous iteration. Although our policy is limited, we show that in its execution a constant fraction of machines is available in each iteration with high probability. It also simplifies our analysis and matches the bound shown in the next section. Furthermore, observe that $\mathrm{LEPT}_{\delta,\alpha}$ is both a δ-delay policy and a $(\delta+\alpha T)$-shift policy. Next, let us introduce some quantities which will turn out to be helpful to analyze $\mathrm{LEPT}_{\delta,\alpha}$.

▶ **Definition 2.6.** *Let Ξ_k denote the random variable describing the total expected processing time of the remaining jobs which have not been started at time $< \tau_k$ divided by Tm. Moreover, let A_k denote the random variable describing the fraction of machines which are available at each time $\tau_1, \dots, \tau_k$, i.e., the machines have completed all jobs assigned in each iteration $0, \dots, k-1$.*

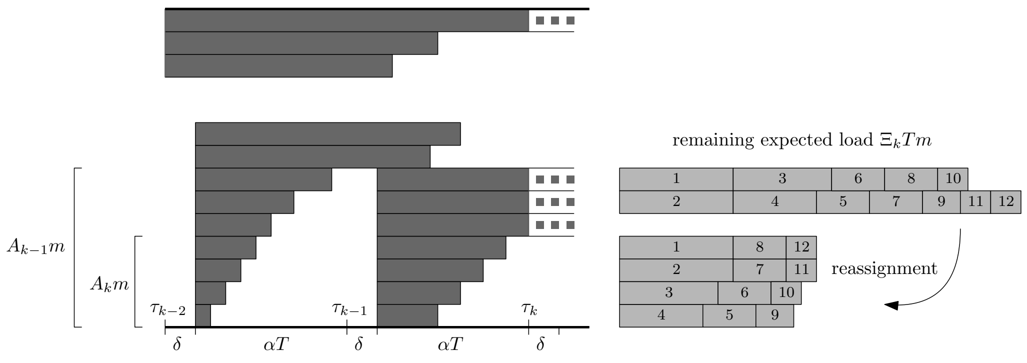

Figure 3 Snippet of $\mathrm{LEPT}_{\delta,\alpha}$: Realizations of jobs observed up to time τ_k are depicted by rectangles in dark grey; the running jobs non-completed by the time of the snippet are indicated by squared dots in dark grey; the expected processing times of the jobs that did not start yet are depicted in light gray. The remaining expected processing time of the twelve light gray jobs is $\Xi_k Tm$. These jobs are reassigned in an FLEPT fashion to the $A_k m$ machines at the bottom at time τ_k, and the first job of each newly formed queue will start at time $\tau_k + \delta$.

A snippet of $\mathrm{LEPT}_{\delta,\alpha}$ together with the introduced notation is illustrated in Figure 3.

Observe that the randomness of Ξ_k occurs only in the set of remaining jobs. We begin with some simple observations.

▶ **Observation 2.7.** *For any k, we have $\Xi_k \le \Xi_{k-1} \le 1$ and $A_k \le A_{k-1} \le 1$ almost surely.*

Next, we want to discuss lower bounds on the expected makespan of an optimal non-anticipatory policy. The first one justifies the use of T in the definition of $\mathrm{LEPT}_{\delta,\alpha}$.

▶ **Lemma 2.8.** *Let $T := 2\cdot\max\{\frac{1}{m}\sum_{j\in\mathcal{J}}\mathbb{E}[P_j], \max_j \mathbb{E}[P_j]\}$ and ℓ be defined as in Lemma 2.3. Then, we have $2\ell \le T \le 2\cdot OPT$.*

Proof. By Corollary 2.4 we immediately obtain the first inequality as ℓ is a lower bound on the averaged load $\frac{1}{m}\sum_{j\in\mathcal{J}}\mathbb{E}[P_j]$. Clearly, for each realization $\boldsymbol{p}$ the makespan is bounded from below by $\frac{1}{m}\sum_{j\in\mathcal{J}} p_j$. Hence, taking expectations we obtain $\mathrm{OPT} \ge \frac{1}{m}\sum_{j\in\mathcal{J}}\mathbb{E}[P_j]$. Lastly, in any non-anticipatory policy obviously all jobs must be scheduled non-preemptively. Therefore, $\max_j \mathbb{E}[P_j]$ is another lower bound on OPT. ◀

We obtain another lower bound when only at most m jobs have to be scheduled.

▶ **Lemma 2.9.** *We have* $\mathbb{E}\left[\max_{j\in\mathcal{J}} P_j\right] \le OPT$.

Proof. For any realization $\boldsymbol{p}$, a lower bound on the optimal makepsan for $\boldsymbol{p}$ is $\max_{j\in\mathcal{J}} p_j$. Taking expectations yields the statement. ◀

We have now set all necessary definitions and lower bounds on the cost of an optimal non-anticipatory policy to devote the remaining part of this section to prove Theorem 2.1. We first derive an upper bound on $\text{LEPT}_{\delta,\alpha}$ in terms of Ξ_{k^*+1}.

▶ **Lemma 2.10.** *We have that* $LEPT_{\delta,\alpha} \le \tau_{k^*+1} + \delta + OPT + \mathbb{E}[\Xi_{k^*+1}] \cdot Tm$.

Proof. Let $C(\boldsymbol{p})$ denote the first point in time in realization $\boldsymbol{p}$ in which all jobs that started before τ_{k^*+1} are completed. We consider an auxiliary policy Π which is identical to $\text{LEPT}_{\delta,\alpha}$ up to time τ_{k^*+1} and starts processing the remaining jobs at time $\max\{\tau_{k^*+1}+\delta, C(\boldsymbol{p})\}$ on an arbitrary single machine. Clearly, $\text{LEPT}_{\delta,\alpha} \le \Pi$ since $\text{LEPT}_{\delta,\alpha}$ starts the remaining jobs at time $\tau_{k^*+1}+\delta$, hence, not later than Π and uses at least as many machines as Π. For any realization $\boldsymbol{p}$ and the starting time of the remaining jobs $S(\boldsymbol{p})$ we have

$$S(\boldsymbol{p}) = \tau_{k^*+1} + \max\{\delta, C(\boldsymbol{p}) - \tau_{k^*+1}\} \le \tau_{k^*+1} + \max\{\delta, \max_{j\in\mathcal{J}} p_j\} \le \tau_{k^*+1} + \delta + \max_{j\in\mathcal{J}} p_j.$$

Hence, by Lemma 2.9 the expected starting time is at most $\tau_{k^*+1} + \delta + \text{OPT}$. By definition of Ξ_{k^*+1} the expected remaining load is exactly $\mathbb{E}[\Xi_{k^*+1}] \cdot Tm$. ◀

Due to the derived upper bound it only remains to bound $\mathbb{E}[\Xi_{k^*+1}]$. The next central lemma provides an upper bound on the probability that this quantity is large.

▶ **Lemma 2.11.** *There exists* $\alpha > 1$ *such that we have* $\mathbb{P}\left(\Xi_{k^*+1} > \frac{1}{m}\right) = o\left(\frac{1}{m}\right)$.

Let us assume for a moment that Lemma 2.11 is true. We then can prove the main theorem.

Proof of Theorem 2.1. By Lemmas 2.10 and 2.11 and by the law of total expectation we obtain

$$\begin{aligned}
\text{LEPT}_{\delta,\alpha} &\le \tau_{k^*+1} + \delta + \text{OPT} + \mathbb{E}[\Xi_{k^*+1}] \cdot Tm \\
&= \tau_{k^*+1} + \delta + \text{OPT} + \underbrace{\mathbb{P}\left(\Xi_{k^*+1} \le \frac{1}{m}\right)}_{\le 1} \cdot \underbrace{\mathbb{E}\left[\Xi_{k^*+1} \middle| \Xi_{k^*+1} \le \frac{1}{m}\right] \cdot Tm}_{\le T} \\
&\quad + \underbrace{\mathbb{E}\left[\Xi_{k^*+1} \middle| \Xi_{k^*+1} > \frac{1}{m}\right]}_{\le 1} \cdot \underbrace{\mathbb{P}\left(\Xi_{k^*+1} > \frac{1}{m}\right) \cdot m}_{=o(1)} \cdot T \\
&\le (\alpha T + \delta) \cdot \mathcal{O}(\log\log(m)) + \delta + \text{OPT} + T + o(1) \cdot T \\
&= \mathcal{O}(\log\log(m)) \cdot \text{OPT},
\end{aligned}$$

where the last step follows by Lemma 2.8 and the choice of δ and α. ◀

Let us return to the proof of Lemma 2.11. The high level idea is to use induction to show that in each iteration there is a constant fraction of available machines with high probability and hence, the remaining expected load after k^* iterations is small with high probability. The first lemma provides a stochastic dominance relation of Ξ_k and A_k to binomially distributed random variables in order to simplify calculations.

▶ **Lemma 2.12.** *For all $u, \xi \in (0,1)$, for all $a \in \left[\frac{1}{2}, 1\right]$ and for any iteration k we have,*

$$\mathbb{P}(\Xi_{k+1} \leq u | \Xi_k \leq \xi, A_k \geq a) \ \geq \ \mathbb{P}\left(\frac{2\xi}{am} Y \leq u\right), \ \textit{where} \quad Y \sim \text{Bin}\left(m, \frac{2\xi}{a\alpha}\right) \tag{1}$$

and

$$\mathbb{P}(A_{k+1} \geq u | \Xi_{k-1} \leq \xi, A_k \geq a) \ \geq \ \mathbb{P}\left(\frac{1}{m} Z \geq u\right), \ \textit{where} \quad Z \sim \text{Bin}\left(\lceil am \rceil, 1 - \frac{2\xi}{a\alpha}\right). \tag{2}$$

Proof sketch. For (2) the key idea is that in each iteration k a machine is available with probability at least $\left(1 - \frac{2\xi}{a\alpha}\right)$ using Corollary 2.4 and Markov's inequality. To show (1) we additionally bound the (normalized) remaining expected load by $\frac{2\xi}{am}$ using Corollary 2.4. ◀

Using Lemma 2.12 we inductively prove probability bounds on Ξ_k and A_k without conditioning on the random variables of the previous iterations. The next lemma handles the base case of the induction stated in Lemma 2.14.

▶ **Lemma 2.13** (Base case of induction). *Let $\gamma_1 = 1$ and $\beta_2 = \frac{3}{4}$. Then, there exists $\epsilon = e^{-\Theta(m^{\frac{1}{3}})}$ such that*

$$\mathbb{P}(\Xi_1 \leq \gamma_1) \geq 1 - \epsilon, \tag{3}$$
$$\mathbb{P}\left(A_2 \geq \beta_2\right) \geq 1 - 3\epsilon. \tag{4}$$

Proof sketch. The first statement (3) is clear, as $\Xi_1 \leq 1$ almost surely. For the second statement, we first show that A_1 is large with high probability using Corollary 2.4 and the Chernoff bound. This bound can be used together with Lemma 2.12 and the Chernoff bound to show (4). ◀

We use the above statement as the base case of an induction to show the next lemma.

▶ **Lemma 2.14.** *Let $\gamma_1 = 1$, $\gamma_{k+1} = \frac{1}{2}\gamma_k^2$ $(\forall k \geq 1)$, $\beta_k = \frac{3}{4} - \frac{2}{\alpha}\sum_{h=1}^{k-2}\gamma_h$ $(\forall k \geq 2)$ and let $k^* := \left\lfloor \log_2\left(\frac{2}{3}(\log_2(m)) + 1\right)\right\rfloor + 2$. Then, there exists $\psi = \Theta\left(\frac{1}{\log\log(m))}\right)$ and $\epsilon = e^{-\Theta(m^{\frac{1}{3}})}$ such that*

$$\mathbb{P}(\Xi_k \leq \gamma_k) \geq 1 - (2^k - 1)\epsilon, \qquad \forall k = 1, \ldots, k^* \tag{5}$$
$$\mathbb{P}(A_k \geq \beta_k - (k-2)\psi) \geq 1 - (2^k - 1)\epsilon, \qquad \forall k = 2, \ldots, k^*. \tag{6}$$

Proof sketch. The doubly exponential decrease of γ_k and the choice of α implies that $\beta_k > \beta_\infty > \frac{5}{8}$. Additionally, the choice of ψ yields $\beta_\infty - (k-2)\psi \geq \frac{1}{2}$. Thus, we can assume that at each iteration with high probability half of the machines are available. For the induction step we make use of Lemma 2.12, the Chernoff bound and the union bound. ◀

By Lemmas 2.12–2.14 we can now show the probability bound on the remaining expected load at iteration $(k^* + 1)$.

Proof of Lemma 2.11. Let $u = \frac{1}{m}, \xi = m^{-\frac{2}{3}}$ and $a = \frac{1}{2}$. Lemma 2.12 (1) implies

$$\mathbb{P}(\Xi_{k^*+1} \leq u | \Xi_{k^*} \leq \xi, A_{k^*} \geq a) \ \geq \ \mathbb{P}\left(Y \leq \frac{1}{4} m^{\frac{2}{3}}\right),$$

where $Y \sim \text{Bin}\left(m, \frac{4}{\alpha} m^{-\frac{2}{3}}\right)$. As $\mathbb{E}[Y] = \frac{4}{\alpha} m^{\frac{1}{3}}$ we obtain by applying the Chernoff bound for $\zeta = \frac{\alpha}{16} m^{\frac{1}{3}} - 1 > 0$

$$\mathbb{P}\left(Y \leq \frac{1}{4} m^{\frac{2}{3}}\right) = \mathbb{P}\left(Y \leq (1+\zeta) \cdot \mathbb{E}[Y]\right) \geq \exp\left(-\frac{\mathbb{E}[Y] \cdot \zeta^2}{2+\zeta}\right) = 1 - \exp(-\Theta(m^{\frac{2}{3}})) \geq 1 - \epsilon,$$

for m large enough. This yields

$$\begin{aligned}\mathbb{P}\left(\Xi_{k^*+1} \leq u\right) &\geq \mathbb{P}\left(\Xi_{k^*+1} \leq u \middle| \Xi_{k^*} \leq \xi, A_{k^*} \geq a\right) \cdot \mathbb{P}\left(\Xi_{k^*} \leq \xi, A_{k^*} \geq a\right) \\ &\geq (1-\epsilon) \cdot \left(1 - (2^{k^*} - 1)\epsilon - (2^{k^*} - 1)\epsilon\right) \\ &\geq 1 - \left(1 + 2 \cdot (2^{k^*} - 1)\right)\epsilon \\ &= 1 - (2^{k^*+1} - 1)\epsilon,\end{aligned}$$

where we used the law of total probability in the first inequality and for the second step we used the union bound and Lemma 2.14. Therefore, as $2^{k^*+1} = \Theta\left(\log(m)\right)$, we have $\mathbb{P}\left(\Xi_{k^*+1} > \frac{1}{m}\right) \cdot m \to 0$ as $m \to \infty$. ◀

3 Lower Bound

Throughout this section, we consider an instance I_N with $n = Nm$ jobs over m machines. Each job has processing time $P_j \sim \text{Bernoulli}\left(\frac{1}{N}\right)$, i.e. $P_j = 1$ with probability $\frac{1}{N}$, and $P_j = 0$ otherwise. The main result of this section is a $\Omega(\delta \log\log(m))$ lower bound on the performance of any δ-delay policy for large values of N. This matches the upper bound obtained in the previous section. Note that the hidden constant in the Ω notation does not depend on the value of $\delta > 0$. For $\delta = \Theta(\text{OPT})$, this implies that no δ-delay policy can improve on the $\log\log m$ performance guarantee of $\text{LEPT}_{\delta,\alpha}$ by more than some constant factor. At the end of the section we show that an analogous result holds for τ-shift policies as well.

▶ **Theorem 3.1.** *Let $\delta \leq 1$. For instance I_N let $OPT_\delta^{\text{DELAY}}$ and OPT denote the value of an optimal δ-delay policy and of an optimal non-anticipatory policy, respectively. Then, for $N = \Omega(\sqrt{m})$ we have*

$$\frac{OPT_\delta^{\text{DELAY}}}{OPT} = \Omega(\delta \cdot \log\log(m)).$$

The proof is split into two main lemmas. The first one relates the expected makespan of an optimal δ-delay policy to the expected makespan of an optimal 1-delay policy.

▶ **Lemma 3.2.** *Assume $\frac{1}{\delta} \in \mathbb{N}$. Then, we have $OPT_\delta^{\text{DELAY}} \geq \delta \cdot OPT_1$.*

The second lemma shows that OPT_1 grows doubly logarithmically with m.

▶ **Lemma 3.3.** *For $N = \Omega(\sqrt{m})$ it holds $OPT_1 = \Omega(\log\log(m))$.*

Let us assume for now that the above lemmas hold. Then, we simply need to show that $\text{OPT} = O(1)$ to prove the theorem.

Proof of Theorem 3.1. On the one hand, Lemma 3.2 and Lemma 3.3 imply $\text{OPT}_\delta^{\text{DELAY}} = \Omega(\delta \cdot \log\log(m))$. On the other hand, we can use the List Scheduling policy (LS) due to Graham [11] to obtain an upper bound on the value of an optimal non-anticipatory policy.

Whenever a machine becomes idle, LS schedules any non-scheduled job on it. For any fixed realization $\boldsymbol{p} = (p_j)_{j\in[Nm]}$ we obtain for its makespan $C^{LS}_{\max}(\boldsymbol{p}) = \left\lceil \frac{1}{m}\sum_{j\in[Nm]} p_j \right\rceil \leq 1 + \frac{1}{m}\sum_{j\in[Nm]} p_j$. As a result, taking expectations on both sides yields

$$\mathrm{OPT} \leq \mathbb{E}[C^{LS}_{\max}] \leq 1 + \frac{1}{m}\sum_{j\in[Nm]} \mathbb{E}[P_j] = 1 + \frac{1}{m}\cdot Nm \cdot \frac{1}{N} = 2,$$

concluding the proof of the theorem. ◀

To prove the lemmas, we first make an observation on the structure of optimal δ-delay policies. When we execute a set of Bernoulli jobs on a machine, we immediately observe whether one of the jobs was a long job (i.e., $p_j = 1$), and also the number of vanishing jobs (i.e., $p_j = 0$) that have already been executed. This indicates that optimal δ-delay policies do not insert deliberate idle time in the schedule (since waiting does not provide any information on running jobs), and for the case $\frac{1}{\delta} \in \mathbb{N}$, they may only take reassignment decisions at times of the form $k\delta$ for $k \in \mathbb{N}$. We call policies with this property *δ-active.*

Proof of Lemma 3.2. The starting time of each job in $\mathrm{OPT}^{\text{DELAY}}_{\delta}$ is an integer multiple of δ, because $1/\delta \in \mathbb{N}$ and $\mathrm{OPT}^{\text{DELAY}}_{\delta}$ is δ-active. Let $J_{ki}(\boldsymbol{p})$ denote the set of jobs started on machine i at time $k\delta$ by $\mathrm{OPT}^{\text{DELAY}}_{\delta}$, for a realization $\boldsymbol{p} \in \{0,1\}^n$ of the processing times. $J_{ki}(\boldsymbol{p})$ may contain many vanishing jobs executed at time $t = k\delta$, and at most one long job executed during the time interval $[k\delta, k\delta + 1)$. It is easy to construct a 1-delay policy (call it Π_1) that executes the same set of jobs $J_{ki}(\boldsymbol{p})$ during the interval $[k, k+1)$ on machine i, by taking at time $k-1$ the same reassignment decisions as $\mathrm{OPT}^{\text{DELAY}}_{\delta}$ takes at time $(k-1)\delta$, and by waiting until time $t = k$ to execute the reassigned jobs. In both schedules, the makespan is caused by the same long job (if there is at least one long job). Its starting time is $\mathrm{OPT}^{\text{DELAY}}_{\delta}(\boldsymbol{p}) - 1$ in the optimal δ-delay policy and $\frac{1}{\delta}(\mathrm{OPT}^{\text{DELAY}}_{\delta}(\boldsymbol{p}) - 1)$ in the policy Π_1. Hence the policy Π_1 has makespan $\Pi_1(\boldsymbol{p}) = \frac{1}{\delta}\cdot(\mathrm{OPT}^{\text{DELAY}}_{\delta}(\boldsymbol{p}) - 1) + 1$ for any realization $\boldsymbol{p} \neq \boldsymbol{0}$, and $\Pi_1(\boldsymbol{p}) = \mathrm{OPT}^{\text{DELAY}}_{\delta}(\boldsymbol{p}) = 0$ if $\boldsymbol{p} = \boldsymbol{0}$. Taking expectations yields

$$\mathrm{OPT}_1 \leq \mathbb{E}[\Pi_1(\boldsymbol{p})] = \frac{1}{\delta}\cdot \mathrm{OPT}^{\text{DELAY}}_{\delta} + \mathbb{P}(\boldsymbol{p} \neq \boldsymbol{0})\cdot\left(1 - \frac{1}{\delta}\right) \leq \frac{1}{\delta}\cdot \mathrm{OPT}^{\text{DELAY}}_{\delta},$$

where we have used the fact that $\delta \leq 1$. This implies $\mathrm{OPT}^{\text{DELAY}}_{\delta} \geq \delta \cdot \mathrm{OPT}_1$. ◀

This lemma allows us to work with 1-delay policies, which are easier to handle: At all times $t \in \mathbb{N}$, a 1-active policy observes the set of jobs non-started yet at time $t - 1 + \epsilon$ (for an infinitesimal small $\epsilon > 0$) and reassigns them to *any* machine, on which they will start at time t at the earliest: we call it an *iteration.*

We denote by R_t the random variable describing the number of remaining jobs at time $t \in \mathbb{N}_0$, before OPT_1 runs the jobs, and by $\Lambda_t = \frac{R_t}{Nm}$ the fraction of remaining jobs at time t. For the initial state we have $\Lambda_0 = 1$ (a.s.). Not surprisingly, the optimal policy balances the remaining jobs as evenly as possible on the m machines.

▶ **Proposition 3.4.** *In iteration t, OPT_1 assigns the remaining $\Lambda_t Nm$ jobs by balancing the load as evenly as possible, i.e., each machine receives $\lceil \Lambda_t N \rceil$ or $\lfloor \Lambda_t N \rfloor$ jobs.*

Proof sketch. Consider a realization of the jobs started before time $t - 1$ for some $t \in \mathbb{N}$, and in which r jobs remain at time $t - 1$. A 1-active policy must reassign the r jobs to the m machines. By moving jobs between two machines, one can show that the balancing policy, which assigns $\lfloor \frac{r}{m} \rfloor$ or $\lceil \frac{r}{m} \rceil$ jobs to each machine, minimizes the (random) number of remaining jobs at time t for the order of stochastic dominance, in the class of 1-active policies. Then, the optimality of the balancing policy follows from the fact that the expected cost-to-go from iteration t, $r \mapsto \mathbb{E}[\mathrm{OPT}_1 - t | R_t = r]$ is monotone decreasing with respect to the number of remaining jobs. ◀

For notational convenience let $\lfloor \Lambda_t N \rceil_i$ denote the number of jobs assigned to machine i by OPT_1. By independence of the processing times, the number of jobs that must be drawn before picking a long job is geometrically distributed with parameter $\frac{1}{N}$. Consequently, we obtain the following observation.

▶ **Observation 3.5.** *For $i \in [m]$ let $G_i \sim \mathrm{Geom}(\frac{1}{N})$ be i.i.d. random variables. Then, we have*

$$\Lambda_{t+1} \overset{d}{=} \frac{1}{Nm} \sum_{i=1}^{m} (\lfloor \Lambda_t N \rceil_i - G_i)_+.$$

We can now prove that OPT_1 is of order $\Omega(\log\log(m))$. To do this, we first need a lemma showing that Λ_t converges quadratically to 0.

▶ **Lemma 3.6.** *For $N = \Omega(\sqrt{m})$ and $t \in \left\{1, \dots, \lfloor \log_2\left(\frac{1}{4}\log_{2e}(m)\right)\rfloor\right\}$ we have*

$$\mathbb{P}\left(\Lambda_t \geq (2e)^{1-2^t}\right) \geq \left(1 - e^{-2\sqrt{m}}\right)^t.$$

A rigorous proof of this lemma is proved in the appendix of the full version [30]. For now, we just explain the intuition behind the quadratic convergence of Λ_t to 0 in expectation, by taking a (hand-wavy look) at the conditional expectation $\mathbb{E}[\Lambda_{t+1}|\Lambda_t = \lambda]$ for large values of N. Using that $\frac{\lfloor \lambda N \rceil}{N} \to \lambda$ and the well-known fact that $\frac{G_i}{N}$ converges in distribution to an exponential random variable $X \sim \mathrm{Exp}(1)$, we see that when $N \to \infty$, $\mathbb{E}[\Lambda_{t+1}|\Lambda_t = \lambda]$ should approach $\mathbb{E}[(\lambda - X)_+] = \int_{x=0}^{\lambda} (\lambda - x)e^{-x}dx = \lambda + e^{-\lambda} - 1$. Then, the quadratic convergence of Λ_t is suggested by the inequalities $\frac{\lambda^2}{e} \leq \lambda + e^{-\lambda} - 1 \leq \frac{\lambda^2}{2}$, which hold for all $\lambda \in [0, 1]$.

With this lemma, we obtain a short proof for Lemma 3.3.

Proof of Lemma 3.3. By Lemma 3.6 we obtain for $t = \Omega(\log\log(m))$ and $N = \Omega(\sqrt{m})$

$$\mathrm{OPT}_1 \geq \mathbb{P}\left(C_{\max} \geq t\right) \cdot t \geq \mathbb{P}\left(\Lambda_t \geq (2e)^{1-2^t}\right) \cdot t \geq \underbrace{\left(1 - e^{-2\sqrt{m}}\right)^t}_{\xrightarrow{m\to\infty} 1} \cdot t = \Omega(\log\log(m)). ◀$$

A similar result can be shown for τ-shift policies, for the same instance I_N.

▶ **Theorem 3.7.** *Let $\tau \leq 1$, such that $\frac{1}{\tau} \in \mathbb{N}$. For instance I_N let OPT_τ^{SHIFT} and OPT denote the value of an optimal τ-shift policy and of an optimal non-anticipatory policy, respectively. Then, for $N = \Omega(\sqrt{m})$ we have*

$$\frac{OPT_\tau^{\text{SHIFT}}}{OPT} = \Omega(\tau \cdot \log\log(m)).$$

Proof. Similarly as for the case of δ-delay policies, for $1/\tau \in \mathbb{N}$ it is clear that an optimal τ-shift policy for instance I_N must be τ-active. Therefore, the optimal τ-active policy coincides with both the optimal τ-shift and the optimal τ-delay policy. This shows that $\mathrm{OPT}_\tau^{\text{SHIFT}} = \mathrm{OPT}_\tau^{\text{DELAY}}$, and the result follows from Theorem 3.1. ◀

4 Conclusion

We considered the stochastic optimization problem of minimizing the expected makespan on parallel identical machines. While any list scheduling policy is a constant factor approximation, the performance guarantee of all fixed assignment policies is at least $\Omega\left(\frac{\log m}{\log\log m}\right)$. We introduced two classes of policies to establish a happy medium between the two extremes of adaptive and non-adaptive policies. The policy $\mathrm{LEPT}_{\delta,\alpha}$, which is both a δ-delay and a τ-shift policy, was shown to have performance guarantee of $\mathcal{O}(\log\log m)$ if δ and τ are in the scale of the instance. Moreover, we provided a matching lower bound for $\delta, \tau = \Theta(\mathrm{OPT})$. Therefore, $\mathrm{LEPT}_{\delta,\alpha}$ improves upon the performance of an optimal fixed assignment policy using a small amount of adaptivity. Moreover, there exists no δ-delay or τ-shift policy beating its performance guarantee by more than a constant.

For the case of $\delta, \tau = \mathcal{O}(\frac{1}{\log\log m})$, Theorem 3.1 gives a constant lower bound, while Theorem 2.1 only gives a doubly logarithmic upper bound. An open question is whether a constant approximation guarantee is possible in this case.

A possible future line of research is the analysis of δ-delay and τ-shift policies for stochastic scheduling problems with other numerous objectives, different machine environments as well as various job characteristics. Moreover, it would be interesting to design other non-anticipatory policies whose adaptivity can be controlled.

References

1 Gagan Aggarwal, Rajeev Motwani, and An Zhu. The load rebalancing problem. *Journal of Algorithms*, 60(1):42–59, 2006.

2 Susanne Albers and Matthias Hellwig. On the value of job migration in online makespan minimization. *Algorithmica*, 79(2):598–623, 2017.

3 Noga Alon, Yossi Azar, Gerhard J. Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

4 B.P. Berg and B.T. Denton. Fast approximation methods for online scheduling of outpatient procedure centers. *INFORMS Journal on Computing*, 29(4):631–644, 2017.

5 Lin Chen, Klaus Jansen, and Guochuan Zhang. On the optimality of approximation schemes for the classical scheduling problem. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 657–668, 2013.

6 Anindya De, Sanjeev Khanna, Huan Li, and Hesam Nikpey. An efficient PTAS for stochastic load balancing with poisson jobs. In *47th International Colloquium on Automata, Languages, and Programming*, volume 168 of *LIPIcs*, pages 37:1–37:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

7 Brian T. Denton, Andrew J. Miller, Hari J. Balasubramanian, and Todd R. Huschka. Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research*, 58(4-1):802–816, 2010.

8 Matthias Englert, Deniz Ozmen, and Matthias Westermann. The power of reordering for online minimum makespan scheduling. *SIAM Journal on Computing*, 43(3):1220–1237, 2014.

9 Michael R. Garey and David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness, 1979.

10 Ashish Goel and Piotr Indyk. Stochastic load balancing and related problems. In *40th Annual Symposium on Foundations of Computer Science*, pages 579–586. IEEE, 1999.

11 Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

12 Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

13 Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and Alexander H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics*, volume 5, pages 287–326. Elsevier, 1979.

14 Anupam Gupta, Amit Kumar, Viswanath Nagarajan, and Xiangkun Shen. Stochastic load balancing on unrelated machines. *Mathematics of Operations Research*, 46(1):115–133, 2021.

15 Varun Gupta, Benjamin Moseley, Marc Uetz, and Qiaomin Xie. Greed works—online algorithms for unrelated machine stochastic scheduling. *Mathematics of Operations Research*, 45(2):497–516, 2020.

16 Willy Herroelen and Roel Leus. Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 42(8):1599–1620, 2004.

17 Dorit S. Hochbaum. Various notions of approximations: Good, better, best and more. *Approximation algorithms for NP-hard problems*, 1997.

18 Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM*, 34(1):144–162, 1987.

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

20 David Isern, David Sánchez, and Antonio Moreno. Agents applied in health care: A review. *International journal of medical informatics*, 79(3):145–166, 2010.

21 Klaus Jansen. An EPTAS for scheduling jobs on uniform processors: using an MILP relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, 24(2):457–485, 2010.

22 Klaus Jansen, Kim-Manuel Klein, and José Verschae. Closing the gap for makespan scheduling via sparsification techniques. *Mathematics of Operations Research*, 45(4):1371–1392, 2020.

23 Jon Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. *SIAM Journal on Computing*, 30(1):191–217, 2000.

24 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1):259–271, 1990.

25 Nicole Megow, Marc Uetz, and Tjark Vredeveld. Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research*, 31(3):513–525, 2006.

26 Rolf H. Möhring, Franz Josef Radermacher, and Gideon Weiss. Stochastic scheduling problems I—general strategies. *Zeitschrift für Operations Research*, 28(7):193–260, 1984.

27 Rolf H. Möhring, Andreas S. Schulz, and Marc Uetz. Approximation in stochastic scheduling: the power of lp-based priority policies. *Journal of the ACM*, 46(6):924–942, 1999.

28 Marco Molinaro. Stochastic lp load balancing and moment problems via the l-function method. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 343–354. SIAM, 2019.

29 Guillaume Sagnol, Christoph Barner, Ralf Borndörfer, Mickaël Grima, Mathees Seeling, Claudia Spies, and Klaus Wernecke. Robust allocation of operating rooms: A cutting plane approach to handle lognormal case durations. *European Journal of Operational Research*, 271(2):420–435, 2018.

30 Guillaume Sagnol and Daniel Schmidt genannt Waldschmidt. Restricted adaptivity in stochastic scheduling, 2021. `arXiv:2106.15393`.

31 Guillaume Sagnol, Daniel Schmidt genannt Waldschmidt, and Alexander Tesch. The price of fixed assignments in stochastic extensible bin packing. In *International Workshop on Approximation and Online Algorithms*, pages 327–347. Springer, 2018.

32 Sartaj K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23(1):116–127, 1976.

33 Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009.

34 Andreas S. Schulz. Stochastic online scheduling revisited. In *International Conference on Combinatorial Optimization and Applications*, pages 448–457. Springer, 2008.

35 Martin Skutella, Maxim Sviridenko, and Marc Uetz. Unrelated machine scheduling with stochastic processing times. *Mathematics of Operations Research*, 41(3):851–864, 2016.

36 Guanlian Xiao, Willem van Jaarsveld, Ming Dong, and Joris van de Klundert. Models, algorithms and performance analysis for adaptive operating room scheduling. *International Journal of Production Research*, 56(4):1389–1413, 2018.