Brief Announcement: Stochastic Parallel Scheduling with Bandit Feedback

Gerdus Benadè
Boston University
Boston, Massachusetts, USA
benade@bu.edu

Rathish Das
University of Houston
Houston, Texas, USA
rathish@central.uh.edu

Thomas Lavastida
University of Texas at Dallas
Richardson, Texas, USA
thomas.lavastida@utdallas.edu

Abstract

We study the fundamental problem of scheduling jobs with stochastic sizes to minimize the total completion time for both single and parallel machines. Traditionally, the size distributions are assumed to be known to the scheduler; the challenge is accounting for the stochasticity. Departing from prior work, we explore the problem of efficiently learning to schedule when the size distributions are unknown and we are unable to directly obtain samples from it. We adopt an online bandit learning framework in which an algorithm interacts with a scheduling environment over a series of T periods. In each period, the algorithm commits to a schedule for n stochastic jobs on up to m parallel machines and then observes only the random cost of the schedule, and not individual job sizes. This so-called *bandit-feedback* provides limited information about the underlying (unknown) job size distributions. The objective is to minimize the total regret — the difference in expected total completion time of schedules chosen by the algorithm from that of the optimal schedule under complete information.

Standard approaches based on multi-armed bandits are computationally infeasible due to the large space of feasible schedules. Utilizing the structure of the scheduling problem, we give simple, combinatorial, and computationally efficient algorithms for both settings with nearly optimal regret. We guarantee $O(n^5\sqrt{T\log T})$ regret for a single machine and $O(\frac{n^5}{m}\sqrt{T\log T})$ for parallel machines. Both algorithms have total time complexity $O(T+n^4)$, or $O(1)$ amortized per-period complexity for large T. We complement our upper bounds with an almost matching lower bound of $\Omega(n\sqrt{T})$ for a single machine.

CCS Concepts

• **Theory of computation** → **Scheduling algorithms**; **Parallel algorithms**; **Online learning algorithms**.

Keywords

Online learning to stochastic schedule, Scheduling with bandit feedback

ACM Reference Format:
Gerdus Benadè, Rathish Das, and Thomas Lavastida. 2025. Brief Announcement: Stochastic Parallel Scheduling with Bandit Feedback. In *37th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '25), July 28-August 1, 2025, Portland, OR, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3694906.3743344

1 Introduction

Scheduling under stochastic job sizes is a fundamental problem in optimization and scheduling theory. Traditionally, much of the work on stochastic scheduling has focused on the case where the distribution of job sizes is known to the scheduler [12, 13, 16, 22, 24, 28]. Recently, motivated by the access to large amounts of data, the question of how to learn a good schedule (or a good scheduling policy) from data has attracted significant attention [19–21, 23, 30]. Typically, these works consider the case where the scheduler gains access to samples from individual jobs [20, 30]. In contrast, we are interested in the question of learning a schedule when we *cannot* directly sample from the distribution, i.e. from *partial feedback*. Specifically, what if the scheduler only observes the *total cost* of the schedule?

This sort of partial feedback can occur in practical settings involving parallel processing. Consider the case of a project manager assigning a set of independent tasks to her team members to complete a project in parallel. The manager may need to assign the work all at once and may only receive feedback on the entire project upon completion, yielding little information about the processing requirements of individual tasks. As another example, consider a setting where jobs are submitted by different agents to a shared cloud computing environment and there is an interest in protecting each agent's privacy. Now partial, noisy feedback can obfuscate private user information, and an algorithm which does not rely on full feedback can find cost efficient schedules without sacrificing privacy.

With this motivation, we consider the problem of scheduling jobs to minimize the total completion time when job sizes are drawn from an unknown distribution. For simplicity, we introduce the single machine setting here; see the full paper for details on generalizing to m identical parallel machines. Let n be the number of jobs that arrive to the system and let $\mathcal{D}$ be a (potentially correlated) distribution over job size vectors $P=(P_1,P_2,\ldots,P_n)\in[0,1]^n$. One can think of each job as a "source" of broadly similar tasks. The scheduler does not have access to $\mathcal{D}$. Instead, they interact with the distribution indirectly via the following game, which takes place over T time periods. In period $t\in[T]$, the scheduler chooses an ordering π^t of the n jobs (a schedule), then an independent sample $P^t\sim\mathcal{D}$ of job sizes is drawn and the scheduler observes the total completion time:

$$\mathrm{COST}(\pi^t,P^t):=\sum_{j=1}^{n}C(j,\pi^t,P^t),$$

where $C(j, \pi, P)$ is the completion time of job j under the schedule π with job sizes P. The scheduler receives only $\textsc{cost}(\pi^t, P^t)$, the (random) sum of completion times, as feedback and does not observe individual job sizes P^t. When the distributions are known, the optimal schedule sorts the jobs in order of non-decreasing $\mu_j := \mathbb{E}[P_j]$ [26]. Denote with π^* this schedule and with $\mathrm{OPT} = \min_\pi \mathbb{E}[\textsc{cost}(\pi, P)]$ its expected total completion time. Our objective is to find an algorithm $\mathcal{A}$ which minimizes the *regret*, the expected additive increase in cost incurred by deviating from the optimal schedule:

$$\mathrm{Regret}(\mathcal{A}, T) = \mathbb{E}\left[\sum_{t=1}^{T} \textsc{cost}(\pi^t, P^t)\right] - T \cdot \mathrm{OPT}.$$

We are typically interested in the case where T grows independent of n. This reflects the realistic case where the number of jobs we must schedule at any given time is bounded, but we have ample time to refine our scheduling decisions in response to the cost feedback. Note that an algorithm with $o(T)$ regret has its per-period expected total completion time approaching OPT as $T \to \infty$.

This problem is in the form of a *Multi-armed Bandit* (MAB) problem, where we can think of each schedule as an arm. The design of algorithms for bandit problems is a well-studied area, for example, see Lattimore and Szepesvári [18] and Slivkins [29] for in-depth introductions to the area. We can view our completion time scheduling problem under the *linear bandits* and *combinatorial bandits* frameworks [1, 3, 5, 7, 15, 17, 27, 33]. While solutions derived from these frameworks give a nearly optimal $\tilde{\Theta}(\sqrt{T})$ dependence on T for the regret, they suffer from the drawback that having exponentially many schedules (arms) can lead to high computational complexity. In many practical scenarios, it is critical that minimal time is spent on computing scheduling decisions since the scheduler may be utilizing the same computing resources needed for the jobs, making prior algorithms derived from these frameworks undesirable. This leads us to ask the following:

> *Can we develop highly efficient algorithms with nearly optimal regret for completion time scheduling under bandit feedback? In particular, can we do so while using only $O(1)$ amortized work per period?*

1.1 Our Contributions

We answer these questions positively by giving the first algorithms for stochastic completion time scheduling under bandit feedback with nearly optimal (up to a factor of $\sqrt{\log T}$) dependence on T in the regret and requiring only $O(1)$ amortized work per-period. In contrast to many papers on stochastic scheduling, our results do not require independence between job sizes — jobs may have correlated size distributions. Importantly, our algorithms are efficient in terms of time complexity. We use *total work* to refer to the computational effort (time complexity) needed by an algorithm to determine the schedules that should be used across all T periods.

Single Machine: Even in the single machine setting, standard approaches based on linear bandits (e.g., LinUCB) will incur high time complexity due to (1) the need to repeatedly estimate underlying parameters (the mean sizes μ_j, in our case), and (2) utilizing the current parameter estimates to determine the next schedule via some exploration policy. The first step is generally accomplished by solving an $n \times n$ linear system, costing $\Omega(\mathrm{poly}(n))$ time per period, while the latter involves finding the schedule with the most optimistic expected cost under the current parameter estimates, costing at least $\Omega(n)$ time. Thus approaches based on linear bandits must incur $\Omega(nT)$ total work. Our first result shows that we can do significantly better than standard approaches in the single machine setting, which we state in the following theorem.

THEOREM 1.1. *There is an algorithm for single machine completion time scheduling under bandit feedback with regret bounded by $O\left(n^5\sqrt{T \log T}\right)$. Moreover, this algorithm can be implemented using $O\left(T + n^4\right)$ total work over all T periods.*

Compared to existing approaches, our algorithm achieves similar regret in terms of T with significantly better time complexity overall, requiring only amortized $O(1)$ work per period for large T. To achieve this, our algorithm estimates the mean difference between pairs of job sizes by iteratively swapping pairs of jobs within some reference schedule. The reference schedule is periodically updated to remain consistent with the updated estimates. To bound the regret, we use an observation due to Lindermayr and Megow [20] that the regret of a schedule can be written as the sum of differences in mean sizes of pairs of jobs that are inverted relative to the optimal schedule. With a careful analysis we can upper bound the total number of rounds that our algorithm inverts a given pair of jobs. To bound the total work, our algorithm lazily updates the reference schedule allowing us to amortize the costs over all T time periods.

Parallel Setting: The parallel setting introduces the complication that some schedules have the same expected total completion time. For example, swapping a pair of jobs which are both scheduled first on two different machines yields no change in expected total completion time. Our algorithm for the parallel setting needs to carefully take this into account when estimating pairwise differences in job sizes. This makes the algorithm design for the parallel setting more intricate, but it also implies that there is "less to learn", since it is enough (in expectation) to schedule each job in the correct position (i.e. first, second, etc.) regardless of machine. This allows us show that the regret decreases in the number of machines.

THEOREM 1.2. *There exists an algorithm for completion time scheduling on parallel machines under bandit feedback with regret $O\left(\frac{n^5}{m}\sqrt{T \log T}\right)$. Moreover, this can be implemented using $O(T + n^4)$ total work over all T periods.*

Despite the regret decreasing in the number of machines, the total work is independent of m, making additional machines a strict benefit. This differs from other scheduling problems where the time complexity of computing schedules generally grows with *both* n and m (see e.g., [25]).

Lower Bound: Finally, we complement our upper bounds with a lower bound for the single machine setting. Naively extending the standard lower bound for multi-armed bandits to our setting yields a $\Omega(\sqrt{nT})$ bound. With a careful construction specific to completion time scheduling we are able to improve on this and show for the single machine setting that the dependence on T in our upper bound is optimal up to logarithmic terms.

THEOREM 1.3. *For any algorithm $\mathcal{A}$ for single machine completion time scheduling under bandit feedback, there exists an instance $\mathcal{I}$, such that the regret of $\mathcal{A}$ on instance $\mathcal{I}$ is $\Omega\left(n\sqrt{T}\right)$.*

This lower bound is information theoretic — it holds for all algorithms regardless of their time complexity. Thus our upper bounds show that we can get close to this lower bound in terms of regret while performing only $O(1)$ amortized work per period for large T. We leave extending the lower bound to the parallel setting for future work.

1.2 Related Work

Offline stochastic machine scheduling problems have been studied since the 1980s [8]. Weiss [31, 32] examines the effectiveness of the Weighted Shorted Expected Processing Time (WSEPT) rule in stochastic machine scheduling and establishes asymptotically optimal performance bounds for specific classes of processing time distributions. More recently, Möhring et al. [24] and Skutella and Uetz [28] devised approximation algorithms for several variants of stochastic scheduling problems by combining WSEPT with linear programming based approaches.

There is a rich literature for multi-armed bandits and online learning, Lattimore and Szepesvári [18], Cesa-Bianchi and Lugosi [4], and Hazan [14] provide comprehensive introductions to the area. Our problem can be framed as an instance of a *linear* or *combinatorial* bandit [1, 3, 5, 7, 15, 17, 27, 33]. This results in a number of arms exponential in n, which makes these approaches intractable both computationally and statistically and leads to a per-period complexity of $\Omega(n)$. Our algorithms require only $O(T + n^4)$ total work across all periods and have $O(1)$ amortized per-period complexity.

More broadly, classical problems in stochastic optimization have been studied in the online learning setting [6, 9, 11]. Most relevant to us, Gatmiry et al. [10] present near-optimal regret bounds of $O(\text{poly}(n)\sqrt{T}\log T)$ for the prophet inequality and Pandora's box problems in the bandit-feedback framework, along $\Omega(\sqrt{T})$ and $\Omega(\sqrt{nT})$ lower bounds, respectively. We study the same form of bandit-feedback tailored to scheduling, where the algorithm observes the total completion time but not individual job sizes, and establish similar near-optimal regret bounds of $O(\text{poly}(n)\sqrt{T\log T})$ and $O(\text{poly}(n)\sqrt{T\log T}/m)$ for the problem of minimizing total completion time in the single and parallel machine settings, respectively, along with a $\Omega(n\sqrt{T})$ lower bound. Agarwal, Ghuge, and Nagarajan [2] work on the less-restricted semi-bandit model, where in each round the algorithm additionally observes the samples of random variables that are "probed" in that round, in addition to the objective value. They give results for the Pandora's box, prophet inequality, and stochastic knapsack problems.

A recent stream of work introduces learning to scheduling problems. Merlis et al. [23] and Lee and Vojnovic [19] both study static scheduling settings and show that preemption can be used to learn about job characteristics. While we consider non-preemptive schedules and never observe direct information about individual jobs, our problem repeats over T periods, which creates similar opportunity for learning. te Rietmole and Uetz [30] consider the problem of minimizing total weighted completion time and ask what happens if you see only a single sample from each of the job size distributions. We similarly limit the sampling of job sizes and observe only the objective function as proxy for the underlying job size samples. Lindermayr and Megow [20] study the same objective with additional (potentially untrustworthy) predictions of the shortest remaining processing time order. Rather than having the predicted order externally provided, we estimate this ordering with increasing accuracy over time by exploring carefully selected schedules.

2 Overview of Results

2.1 Single Machine Setting

We now give a brief overview of our results, starting with the upper bound for completion time scheduling on a single machine with bandit feedback. In order to motivate our algorithm, we use the following characterization of an algorithm's regret. To state this properly, we let $N(j,j') := |\{t \in [T] \mid \pi^t(j) < \pi^t(j')\}|$ be the number of times algorithm $\mathcal{A}$ puts job j before job j'.

LEMMA 2.1. *For any algorithm $\mathcal{A}$ for completion time scheduling with bandit feedback, we have*

$$\text{Regret}(\mathcal{A},T) = \sum_{j,j':\mu_j>\mu_{j'}} (\mu_j - \mu_{j'})\mathbb{E}[N(j,j')].$$

This lemma follows from an observation due to Lindermayr and Megow [20].

Let $\Delta(j,j') := \mu_j - \mu_{j'}$ denote the difference in the mean job sizes of j,j'. We can interpret Lemma 2.1 as expressing the regret as the sum over a collection of $\binom{n}{2}$ two-armed bandit problems, one for each pair of jobs $j,j' \in [n]$, each with an arm for each relative ordering of j,j'. For each pair of jobs j,j' in the summation, the total regret incurred is $\Delta(j,j')$, the regret of playing the bad arm (inverting jobs j,j'), times the expected number of times the bad arm is played, $N(j,j')$.

One challenge is that we have to decide on a schedule collectively, so we cannot necessarily solve a sequence of two-arm bandit problems. Instead our approach is based on updating estimates of $\Delta(j,j')$ for each pair j,j' and then fixing the relative order of j and j' once we are confident in our estimate of $\Delta(j,j')$. To this end we maintain a confidence interval $C(j,j')$ around our estimate of $\Delta(j,j')$. Once the relative order of j and j' is fixed our algorithm aims to select schedules that are consistent with this relative ordering. Inconsistencies may be introduced only when evaluating another pair of jobs j,j'', and we show that this can only happen for a limited number of periods after the relationship between j and j' is established. We maintain a directed acyclic graph (DAG) G to store precedence relations and use topological orderings of G to choose schedules. More formally, we define consistency of a schedule in the following way.

Definition 2.2. Given a DAG $G = ([n],E)$ and a permutation $\pi : [n] \to [n]$, we say that π is consistent with G if and only if for all pairs of jobs j,j' we have $j \to j' \in E \implies \pi(j) < \pi(j')$.

See Algorithm 1 and Algorithm 2 for a formal description of our algorithm, which can be implemented efficiently, as we state next.

LEMMA 2.3. *Algorithm 1 can be implemented in at most $O(T + n^4)$ total computational work over all T periods.*

The key observation for bounding the time complexity is that the potentially expensive operation of updating the schedule π to be

Algorithm 1: $\tilde{O}(\text{poly}(n) \cdot \sqrt{T})$ Regret Alg.

```
Data: Job count $n$, Time horizon $T$
Result: Sequence of schedules with $\tilde{O}(n^5\sqrt{T})$ regret.
1  Procedure LearnToSchedule($n$):
       //Initialize graph keeping track of learned information.
2      $G \leftarrow ([n], \emptyset)$
3      $U \leftarrow \{\{j, j'\} \mid j, j' \in [n], j \neq j'\}$
       //Initialize estimates.
4      $\forall \{j, j'\} \in U, \widehat{\Delta}(j, j') = 0$
5      $\pi \leftarrow$ any schedule
6      for $\ell = 1, 2, \ldots$ do
7          for $\{j, j'\} \in U$ do
8              $\widehat{\Delta}(j, j'), C(j, j') \leftarrow$ Update($\pi$, $j$, $j'$, $\ell$)
9              if $0 \notin C(j, j')$ then
10                 Add edge $j \to j'$ to $G$ if $\widehat{\Delta}(j, j') < 0$
11                 Otherwise add the edge $j' \to j$ to $G$
12                 $U \leftarrow U \setminus \{j, j'\}$
13         if $G$ is acyclic then
14             $\pi \leftarrow$ any schedule consistent with $G$
15         else
16             $\pi \leftarrow$ any schedule
```

Algorithm 2: Update Procedure

```
Data: Schedule $\pi$, Jobs $j$, $j'$, and phase $\ell$
Result: Updated estimate $\widehat{\Delta}(j, j')$ and CI $C(j, j')$
1  Procedure Update($\pi$, $j$, $j'$, $\ell$):
2      WLOG say that $\pi(j) < \pi(j')$
3      Let $\pi'$ be $\pi$ but with $j$ and $j'$ swapped
4      Use schedule $\pi$ and let $a$ be its cost
5      Use schedule $\pi'$ and let $b$ be its cost
6      $k \leftarrow \pi(j') - \pi(j)$
7      $\widehat{\Delta}(j, j') \leftarrow \left((\ell - 1)\widehat{\Delta}(j, j') + (a - b)/k\right)/\ell$
8      $C(j, j') \leftarrow \left[\widehat{\Delta}(j, j') \pm 2n^2\sqrt{\log T/\ell}\right]$
9      return $\widehat{\Delta}(j, j'), C(j, j')$
```

consistent with the graph G only needs to be done when a new edge is added to G, which happens at most $O(n^2)$ times.

Moving to the regret analysis, it suffices to bound the number of times the algorithm incorrectly orders each pair of jobs by Lemma 2.1. This is accomplished by the following lemma.

LEMMA 2.4. *For each pair $j, j' \in [n]$ with $\mu_j > \mu_{j'}$, we have $\mathbb{E}[N(j, j')] = O\left(n^6 \log(T)/(\mu_j - \mu_{j'})^2\right)$.*

This essentially comes from the fact that we add edges to the graph G when the corresponding confidence interval no longer contains 0. However, we have to be more careful as it is possible for our algorithm to invert a pair of jobs j and j' even when the edge $j \to j'$ is in G. Thus to bound $\mathbb{E}[N(j, j')]$ we have to account for these later time steps where inversions may incur which involves carefully considering triples of jobs j, j', j'' in a case analysis.

From here, Theorem 1.1 follows directly from Lemma 2.3 and Lemma 2.4 via standard calculations.

2.2 Parallel Setting

Next, we sketch the challenges that arise in moving from the single machine setting to the parallel setting and how we handle them. First, our results for the single machine setting were heavily motivated by the regret decomposition given in Lemma 2.1, which no longer holds in the parallel setting. As a result, it is no longer possible to bound the regret by bounding the number of times our algorithm inverts some pair of jobs compared to the SEPT order π^*. Instead, we carefully bound the regret of each schedule used in terms of the confidence interval width, leading to the stated regret bound.

Second, our algorithm can no longer explore by simply swapping the positions of any two jobs, as we did in Algorithm 1. Swapping some pairs of jobs does not change the expected total completion time of the resulting schedule, giving us no information about the corresponding pairwise difference. To account for this, we modify the update step of the algorithm to ensure that it is still possible to learn all the necessary pairwise differences between different jobs. With some care, we are additionally able to do this in a computationally efficient manner, leading to Theorem 1.2.

2.3 Lower Bound

To establish the lower bound, we define two instances such that it is challenging for any algorithm to distinguish between them, leading to high regret. At a high level, the optimal schedule for the first instance follows the order $\pi^*(j) = j$ for $j \in [n]$ but in the second instance each job pair of the form $j, j+1$ for odd $j \in [n]$ has their order swapped in the optimal schedule. Each jobs' size is Bernoulli distributed with different parameters for the mean value such that it is difficult for any algorithm to distinguish between these two instances without incurring significant regret. This is quantified by bounding the KL-divergence between the distributions induced by the algorithm on each instance.

3 Conclusion and Future Work

In this paper we study bandit algorithms for scheduling to minimize the total completion time under stochastic job sizes from an unknown distribution in the parallel setting. Due to the combinatorial nature of this problem, standard approaches from multi-armed bandits are computationally infeasible, so we propose a highly computationally efficient algorithm with an almost optimal regret bound.

There are several directions of study that are opened up by this work. An immediate open problem is tightening the dependence of the regret bounds on n. For completion time scheduling under bandit feedback, it would be interesting to extend our results to the case where each job j has a weight w_j measuring its relative importance, i.e., minimizing weighted total completion time. Finding bandit algorithms with low regret for stochastic makespan minimization [12, 16] may be particularly interesting, since the non-linear nature of the load balancing objective makes the problem challenging when you can't sample job sizes directly.

References

[1] Jacob D. Abernethy, Elad Hazan, and Alexander Rakhlin. 2008. Competing in the Dark: An Efficient Algorithm for Bandit Linear Optimization. In *21st Annual Conference on Learning Theory - COLT 2008, Helsinki, Finland, July 9-12, 2008*. Omnipress, 263–274. http://colt2008.cs.helsinki.fi/papers/127-Abernethy.pdf

[2] Arpit Agarwal, Rohan Ghuge, and Viswanath Nagarajan. 2023. Semi-Bandit Learning for Monotone Stochastic Optimization. *CoRR* abs/2312.15427 (2023). https://doi.org/10.48550/ARXIV.2312.15427 arXiv:2312.15427

[3] Sébastien Bubeck, Nicolò Cesa-Bianchi, and Sham M. Kakade. 2012. Towards Minimax Policies for Online Linear Optimization with Bandit Feedback. In *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland (JMLR Proceedings, Vol. 23)*, Shie Mannor, Nathan Srebro, and Robert C. Williamson (Eds.). JMLR.org, 41.1–41.14. http://proceedings.mlr.press/v23/bubeck12a/bubeck12a.pdf

[4] Nicolò Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, learning, and games*. Cambridge University Press. https://doi.org/10.1017/CBO9780511546921

[5] Nicolò Cesa-Bianchi and Gábor Lugosi. 2012. Combinatorial bandits. *J. Comput. Syst. Sci.* 78, 5 (2012), 1404–1422. https://doi.org/10.1016/J.JCSS.2012.01.001

[6] Shuchi Chawla, Evangelia Gergatsouli, Yifeng Teng, Christos Tzamos, and Ruimin Zhang. 2020. Pandora's Box with Correlations: Learning and Approximation. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, Sandy Irani (Ed.). IEEE, 1214–1225. https://doi.org/10.1109/FOCS46700.2020.00116

[7] Varsha Dani, Sham M Kakade, and Thomas Hayes. 2007. The Price of Bandit Information for Online Optimization. In *Advances in Neural Information Processing Systems*, Vol. 20. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2007/file/bf62768ca46b6c3b5bea9515d1a1fc45-Paper.pdf

[8] Michael Alan Howarth Dempster. 1982. A stochastic approach to hierarchical planning and scheduling. In *Deterministic and Stochastic Scheduling: Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems held in Durham, England, July 6–17, 1981*. Springer, 271–296.

[9] Hossein Esfandiari, Mohammad Taghi Hajiaghayi, Brendan Lucier, and Michael Mitzenmacher. 2019. Online Pandora's Boxes and Bandits. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 1885–1892. https://doi.org/10.1609/AAAI.V33I01.33011885

[10] Khashayar Gatmiry, Thomas Kesselheim, Sahil Singla, and Yifan Wang. 2024. Bandit Algorithms for Prophet Inequality and Pandora's Box. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, David P. Woodruff (Ed.). SIAM, 462–500. https://doi.org/10.1137/1.9781611977912.18

[11] Evangelia Gergatsouli and Christos Tzamos. 2022. Online Learning for Min Sum Set Cover and Pandora's Box. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, 7382–7403. https://proceedings.mlr.press/v162/gergatsouli22a.html

[12] Anupam Gupta, Amit Kumar, Viswanath Nagarajan, and Xiangkun Shen. 2018. Stochastic Load Balancing on Unrelated Machines. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, Artur Czumaj (Ed.). SIAM, 1274–1285. https://doi.org/10.1137/1.9781611975031.83

[13] Varun Gupta, Benjamin Moseley, Marc Uetz, and Qiaomin Xie. 2020. Greed Works - Online Algorithms for Unrelated Machine Stochastic Scheduling. *Math. Oper. Res.* 45, 2 (2020), 497–516. https://doi.org/10.1287/MOOR.2019.0999

[14] Elad Hazan. 2016. Introduction to Online Convex Optimization. *Found. Trends Optim.* 2, 3-4 (2016), 157–325. https://doi.org/10.1561/2400000013

[15] Shinji Ito, Daisuke Hatano, Hanna Sumita, Kei Takemura, Takuro Fukunaga, Naonori Kakimura, and Ken-Ichi Kawarabayashi. 2019. Oracle-Efficient Algorithms for Online Linear Optimization with Bandit Feedback. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/e6385d39ec9394f2f3a354d9d2b88eec-Paper.pdf

[16] Jon M. Kleinberg, Yuval Rabani, and Éva Tardos. 1997. Allocating Bandwidth for Bursty Connections. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, Frank Thomson Leighton and Peter W. Shor (Eds.). ACM, 664–673. https://doi.org/10.1145/258533.258661

[17] Tor Lattimore, Csaba Szepesvári, and Gellért Weisz. 2020. Learning with Good Feature Representations in Bandits and in RL with a Generative Model. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 5662–5670. http://proceedings.mlr.press/v119/lattimore20a.html

[18] Tor Lattimore and Csaba Szepesvári. 2020. *Bandit Algorithms*. Cambridge University Press.

[19] Dabeen Lee and Milan Vojnovic. 2021. Scheduling jobs with stochastic holding costs. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 19375–19384. https://proceedings.neurips.cc/paper/2021/hash/a19744e268754fb0148b017647355b7b-Abstract.html

[20] Alexander Lindermayr and Nicole Megow. 2022. Permutation Predictions for Non-Clairvoyant Scheduling. In *SPAA '22: 34th ACM Symposium on Parallelism in Algorithms and Architectures, Philadelphia, PA, USA, July 11 - 14, 2022*, Kunal Agrawal and I-Ting Angelina Lee (Eds.). ACM, 357–368. https://doi.org/10.1145/3490148.3538579

[21] Sebastián Marbán, Cyriel Rutten, and Tjark Vredeveld. 2011. Learning in Stochastic Machine Scheduling. In *Approximation and Online Algorithms - 9th International Workshop, WAOA 2011, Saarbrücken, Germany, September 8-9, 2011, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 7164)*, Roberto Solis-Oba and Giuseppe Persiano (Eds.). Springer, 21–34. https://doi.org/10.1007/978-3-642-29116-6_3

[22] Nicole Megow, Marc Uetz, and Tjark Vredeveld. 2006. Models and Algorithms for Stochastic Online Scheduling. *Math. Oper. Res.* 31, 3 (2006), 513–525. https://doi.org/10.1287/MOOR.1060.0201

[23] Nadav Merlis, Hugo Richard, Flore Sentenac, Corentin Odic, Mathieu Molina, and Vianney Perchet. 2023. On Preemption and Learning in Stochastic Scheduling. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 24478–24516. https://proceedings.mlr.press/v202/merlis23a.html

[24] Rolf H. Möhring, Andreas S. Schulz, and Marc Uetz. 1999. Approximation in stochastic scheduling: the power of LP-based priority policies. *J. ACM* 46, 6 (1999), 924–942. https://doi.org/10.1145/331524.331530

[25] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. 1995. Fast Approximation Algorithms for Fractional Packing and Covering Problems. *Math. Oper. Res.* 20, 2 (1995), 257–301. https://doi.org/10.1287/MOOR.20.2.257

[26] Michael H. Rothkopf. 1966. Scheduling with Random Service Times. *Management Science* 12, 9 (1966), 707–713. http://www.jstor.org/stable/2627947

[27] Paat Rusmevichientong and John N. Tsitsiklis. 2010. Linearly Parameterized Bandits. *Math. Oper. Res.* 35, 2 (2010), 395–411. https://doi.org/10.1287/MOOR.1100.0446

[28] Martin Skutella and Marc Uetz. 2005. Stochastic Machine Scheduling with Precedence Constraints. *SIAM J. Comput.* 34, 4 (2005), 788–802. https://doi.org/10.1137/S0097539702415007

[29] Aleksandrs Slivkins. 2019. Introduction to Multi-Armed Bandits. *Found. Trends Mach. Learn.* 12, 1-2 (2019), 1–286. https://doi.org/10.1561/2200000068

[30] Puck te Rietmole and Marc Uetz. 2024. Sequencing Stochastic Jobs with a Single Sample. In *Combinatorial Optimization - 8th International Symposium, ISCO 2024, La Laguna, Tenerife, Spain, May 22-24, 2024, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 14594)*, Amitabh Basu, Ali Ridha Mahjoub, and Juan José Salazar González (Eds.). Springer, 235–247. https://doi.org/10.1007/978-3-031-60924-4_18

[31] Gideon Weiss. 1990. Approximation results in parallel machnies stochastic scheduling. *Annals of Operations Research* 26, 1 (1990), 195–242.

[32] Gideon Weiss. 1992. Turnpike optimality of Smith's rule in parallel machines stochastic scheduling. *Mathematics of Operations Research* 17, 2 (1992), 255–270.

[33] Shuo Yang, Tongzheng Ren, Sanjay Shakkottai, Eric Price, Inderjit S. Dhillon, and Sujay Sanghavi. 2022. Linear Bandit Algorithms with Sublinear Time Complexity. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, 25241–25260. https://proceedings.mlr.press/v162/yang22m.html