Sojourn Time Minimization of Successful Jobs

Yuan Yao
Wuhan Institute of Technology
yaoyuan.usc@gmail.com

Marco Paolieri
Univ. of Southern California
paolieri@usc.edu

Leana Golubchik
Univ. of Southern California
leana@usc.com*

1. INTRODUCTION

Due to a growing interest in deep learning applications [5], compute-intensive and long-running (hours to days) training jobs have become a significant component of datacenter workloads. A large fraction of these jobs is often exploratory, with the goal of determining the best model structure (e.g., the number of layers and channels in a convolutional neural network), hyperparameters (e.g., the learning rate), and data augmentation strategies for the target application. Notably, *training jobs are often terminated early* if their learning metrics (e.g., training and validation accuracy) are not converging, with only a few completing successfully.

For this motivating application, we consider the problem of scheduling a set of jobs that can be terminated at predetermined checkpoints with known probabilities estimated from historical data. We prove that, in order to minimize the time to complete the first K *successful* jobs on a single server, optimal scheduling does not require preemption (even when preemption overhead is negligible) and provide an optimal policy; advantages of this policy are quantified through simulation.

Related Work. While job scheduling has been investigated extensively in many scenarios (see [6] and [2] for a survey of recent result), most policies require that the cost of waiting times of each job be known at scheduling time; in contrast, in our setting the scheduler does not know which job will be the K-th successful job, and sojourn times of subsequent jobs do not contribute to the target metric.

For example, [4, 3] minimize makespan (i.e., the time to complete *all* jobs) for known execution times and waiting time costs; similarly, Gittins index [1] and SR rank [7] minimize expected sojourn time of *all* jobs, i.e., both successfully completed jobs and jobs terminated early. Unfortunately, scheduling policies not distinguishing between these two types of jobs may favor jobs where the next stage is short and leads to early termination with high probability, which is an undesirable outcome in our applications of interest.

2. OPTIMAL SCHEDULING POLICY

We consider the problem of scheduling N jobs on a single server, where the exact size of each job $i = 1, \ldots, N$ is unknown, but it is modeled by a discrete random variable taking one of the M_i values $0 < x_{i,1} < x_{i,2} < \cdots < x_{i,M_i}$; we denote by $p_{i,j} \in (0, 1)$ the probability that job i terminates when service time reaches size $x_{i,j}$, with $\sum_{j=1}^{M_i} p_{i,j} = 1$. In our applications of interest, the largest possible size x_{i,M_i} corresponds to the successful completion of job i, while other sizes $x_{i,j}$ for $1 \leq j < M_i$ represent the early termination of job i after its j-th checkpoint. Note that this is equivalent to setting checkpoints at deterministic fractions of each job.

Jobs are scheduled on the server using a preemptive, non-anticipating policy which selects a job and runs it until the next checkpoint. For example, if job i is initially selected, its first stage with size $x_{i,1}$ is completed: with probability $p_{i,1}$, the job terminates after this stage; with probability $1-p_{i,1}$, the job continues. If job i continues, its remaining size values are $x_{i,2} < \cdots < x_{i,M_i}$ with probabilities $p_{i,j}/(1 - p_{i,1})$ for $j = 2, \ldots, M_i$; when it is selected to run again by the scheduler, the second stage is completed in $x_{i,2}$ time units. This process continues until all jobs terminate early (at some checkpoint) or complete their last stage successfully. Our goal is to *minimize the expected time to complete K successful jobs*, for $1 \leq K \leq N$.

First, we show that preemption is not required to minimize the expected time to complete K successful jobs.

THEOREM 2.1. *There exists a schedule minimizing the expected sojourn time of the K-th successful job where jobs are not preempted.*

PROOF. Assume that there is a schedule S minimizing the expected sojourn time of the K-th successful job, where job i is the last job being preempted; let j be the last job preempting i, and let s_j and s_i indicate the last interleaving stages of j and i, respectively. Then, the modified schedule where all stages of i before s_i are moved between s_j and s_i (so that i is not preempted) has no higher expected sojourn time of the K successful job: (1) if i terminates early after any of its stages, its sojourn time is increased, but it does not contribute to the mean sojourn time of the K-th *successful* jobs; (2) if i completes successfully, its sojourn time does not change in the modified schedule; (3) sojourn time does not change for other jobs completing before the first stage or after the last stage of i in the original schedule; (4) for other jobs completing or terminating between the first and last stage of i in the original schedule, sojourn time is reduced. Since, by hypothesis, i is the last job being preempted, at least one other job k has nonzero probability of terminating successfully between the first and last stage of i in the original schedule; the expected sojourn time of the

* Work supported in part by NSF Grants CCF-1763747 and CNS-1816887.

K-th successful job does not increase in the modified schedule. As a result we can repeat the move performed above to create a schedule without interleaved jobs. And this schedule has expected sojourn time of the K-th successful job less than or equal to that of S. $\square$

Theorem 2.1 significantly reduces the problem of finding an optimal policy, since it allows us to schedule entire jobs until completion or early termination, instead of individual job stages. We represent each scheduling policy as a permutation $S = \langle J_1, J_2, \ldots, J_N \rangle$ of $\langle 1, 2, \ldots, N \rangle$ and we denote the expected sojourn time of the K-th successful job under S by $T_K^{(S)}$. Then, our goal is to find an optimal schedule $S^* = \arg\min_{S \in \mathbb{G}_\mathbb{N}} T_K^{(S)}$ where $\mathbb{G}_\mathbb{N}$ is the set of all permutations of $\langle 1, 2, \ldots, N \rangle$. For each policy S, $T_K^{(S)}$ is given by:

$$T_K^{(S)} = \sum_{\boldsymbol{l} \in \boldsymbol{L}} Pr[\boldsymbol{l}] \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) = \sum_{\boldsymbol{l} \in \boldsymbol{L}} \left(\prod_{i=1}^{N} p_{J_i, l_{J_i}} \right) \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) \qquad (1)$$

where $L_{J_i} = \{1, 2, \ldots, M_{J_i}\}, i = 1, \ldots, N$ is the set of possible stages for job J_i, $\boldsymbol{L} = L_{J_1} \times L_{J_2} \times \ldots \times L_{J_N}$, and each vector $\boldsymbol{l} = [l_{J_1}, l_{J_2}, \ldots, l_{J_N}]^T \in \boldsymbol{L}$ is a different combination of final execution stages. We define $f_K(\boldsymbol{l}) : \boldsymbol{L} \to \{0, K, K+1, K+2, \ldots, N\}$ as the index of the K-th completed job given the final execution stages $\boldsymbol{l}$; we let $f_K(\boldsymbol{l}) = 0$ when there are fewer than K successful jobs in $\boldsymbol{l}$.

The following lemma shows that the order of first K jobs does not affect $T_K^{(S)}$.

Lemma 2.2. *Given two schedules S and S' where the order of the first K jobs is permuted, $T_K^{(S)} = T_K^{(S')}$.*

Proof. Irrespective of the order of the first K jobs, their contributions to $T_K^{(\cdot)}$ remains the same since $T_K^{(\cdot)}$ includes the sum of execution times of all these jobs. $\square$

Then, the next lemma provides an optimal ordering of the following $N - K$ jobs.

Lemma 2.3. *In an optimal schedule $S^* = \langle J_1^*, J_2^*, \ldots, J_N^* \rangle$, we have that $R(J_q^*) \leq R(J_{q+1}^*)$ for all $K \leq q < N$, where $R(i) = \left(\sum_{k=1}^{M_i} p_{i,k} x_{i,k} \right) / p_{i,M_i}$ is the rank of job i.*

Proof Sketch. Consider a schedule $S = \langle J_1, J_2, \ldots, J_N \rangle$ and look at two adjacent jobs J_q and J_{q+1}, where $q \in \{K, K+1, \ldots, N-1\}$. Swapping the positions of these two jobs gives a new schedule $S' = \langle J_1, J_2, \ldots, J_{q+1}, J_q, \ldots, J_N \rangle$. Considering whether the K-th successful job happens before job J_q, at job J_q, at job J_{q+1} or after job J_{q+1}, for schedule S, we have

$$\begin{aligned} T_K^{(S)} &= \sum_{\boldsymbol{l} \in \boldsymbol{L}} Pr[\boldsymbol{l}] \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) \\ &= \sum_{\boldsymbol{l} \in \boldsymbol{L}, f_K(\boldsymbol{l}) < q} Pr[\boldsymbol{l}] \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) \\ &+ \sum_{\boldsymbol{l} \in \boldsymbol{L}, f_K(\boldsymbol{l}) = q} Pr[\boldsymbol{l}] \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) \\ &+ \sum_{\boldsymbol{l} \in \boldsymbol{L}, f_K(\boldsymbol{l}) = q+1} Pr[\boldsymbol{l}] \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) \\ &+ \sum_{\boldsymbol{l} \in \boldsymbol{L}, f_K(\boldsymbol{l}) > q+1} Pr[\boldsymbol{l}] \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) \end{aligned} \qquad (2)$$

A similar expression can be derived for $T_K^{(S')}$ with four terms on the right hand side (RHS), with the following observations: 1) the first term on the RHS of Eq. (2) and first term on the RHS of $T_K^{(S')}$ are equal. Intuitively, S and S' share the same set of first $q-1$ jobs. Since there are already K successful jobs before jobs J_q and J_{q+1}, these two jobs have no contribution to $T_K^{(S)}$ or $T_K^{(S')}$. 2) similarly, the fourth term on the RHS of Eq. (2) and fourth term on the RHS of $T_K^{(S')}$ are equal.

Let $\hat{\boldsymbol{L}} \subset \boldsymbol{L}$, $\bar{\boldsymbol{L}} \subset \boldsymbol{L}$ be the sets of execution results of schedule S where there are exactly $K-1$ and $K-2$ successful jobs in the first $q-1$ jobs, respectively. We may express $Pr[\boldsymbol{l}]$ with $p_{i,j}$ values in the second and third term on the RHS of Eq. (2). In particular the second term of Eq. (2) only depends on jobs 1 through q and thus can be written as:

$$\sum_{\boldsymbol{l} \in \boldsymbol{L}, f_K(\boldsymbol{l}) = q} Pr[\boldsymbol{l}] \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) = \sum_{\boldsymbol{l} \in \hat{\boldsymbol{L}}} \prod_{i=1}^{q-1} p_{J_i, l_{J_i}} \cdot p_{J_q, M_{J_q}} \left(\sum_{i=1}^{q-1} x_{J_i, l_{J_i}} + x_{J_q, M_{J_q}} \right) \qquad (3)$$

And the third term can be expanded as follows:

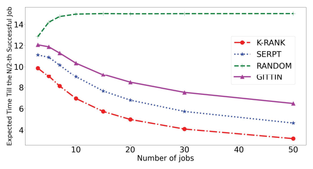
$$\begin{aligned} &\sum_{\boldsymbol{l} \in \boldsymbol{L}, f_K(\boldsymbol{l}) = q+1} Pr[\boldsymbol{l}] \cdot \left(\sum_{i=1}^{f_K(\boldsymbol{l})} x_{J_i, l_{J_i}} \right) = \\ &\sum_{\boldsymbol{l} \in \hat{\boldsymbol{L}}} \prod_{i=1}^{q-1} p_{J_i, l_{j_i}} \sum_{k=1}^{M_{J_q}-1} p_{J_q, k} p_{J_{q+1}, M_{J_{q+1}}} \cdot \\ &\left(\sum_{i=1}^{q-1} x_{J_i, l_{J_i}} + x_{J_q, k} + x_{J_{q+1}, M_{J_{q+1}}} \right) + \\ &\sum_{\boldsymbol{l} \in \bar{\boldsymbol{L}}} \prod_{i=1}^{q-1} p_{J_i, l_{j_i}} \cdot p_{J_q, M_{J_q}} p_{J_{q+1}, M_{J_{q+1}}} \cdot \\ &\left(\sum_{i=1}^{q-1} x_{J_i, l_{J_i}} + x_{J_q, M_{J_q}} + x_{J_{q+1}, M_{J_{q+1}}} \right) \end{aligned} \qquad (4)$$

The first term on the RHS of Eq. (4) corresponds to the cases where J_q failed but J_{q+1} succeeded. And the second term corresponds to the cases where both J_q and J_{q+1} succeeded.

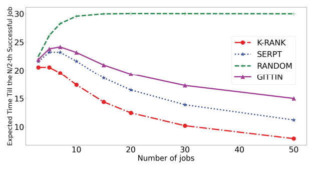
A similar expression can be derived for the third term of $T_K^{(S')}$. The only difference between this expression and Eq. (4) is the order of jobs J_q and J_{q+1}. Thus its second term equals the second term on the RHS of Eq. (4).

Given above, we can compare $T_K^{(S)}$ and $T_K^{(S')}$:

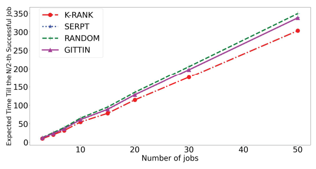
$$T_K^{(S)} - T_K^{(S')} = \cdots = \sum_{\boldsymbol{l} \in \hat{\boldsymbol{L}}} \prod_{i=1}^{q-1} p_{J_i, l_{J_i}} p_{J_q, M_{j_q}} p_{J_{q+1}, M_{j_{q+1}}} \cdot \left(\frac{\sum_{k=1}^{M_{J_q}} p_{J_q, k} x_{J_q, k}}{p_{J_q, M_{J_q}}} - \frac{\sum_{k=1}^{M_{J_{q+1}}} p_{J_{q+1}, k} x_{J_{q+1}, k}}{p_{J_{q+1}, M_{J_{q+1}}}} \right)$$



(a) First



(b) Second



(c) N/2-th

Figure 1: Expected time till the K-th successful job

$$= \sum_{\iota \in \hat{\boldsymbol{L}}} \prod_{i=1}^{q-1} p_{J_i, l_{J_i}} p_{J_q, M_{j_q}} p_{J_{q+1}, M_{j_{q+1}}} [R(J_q) - R(J_{q+1})] \quad (5)$$

We can see from Eq. (5) that if $R(J_q) > R(J_{q+1})$, then swapping them results in a lower $T_K^{(\cdot)}$ value. Therefore, in an optimal schedule S^* we must have $R(J_q^*) \leq R(J_{q+1}^*)$ for all $q \geq K$. $\square$

Based on Lemmas 2.2 and 2.3, we can state our main result.

THEOREM 2.4. *The schedule* $S^* = \langle J_1^*, J_2^*, \ldots, J_N^* \rangle$ *where* $R(J_1^*) \leq R(J_2^*) \leq \ldots R(J_N^*)$ *minimizes* $T_K^{(\cdot)}$.

PROOF. Let $S = \langle J_1, J_2, \ldots, J_N \rangle$ be an optimal schedule. Then we can reorder the first K jobs of S by sorting them according to their rank in ascending order. Suppose the resulting schedule is $S' = \langle J_1', J_2', \ldots, J_K', J_{K+1}, \ldots, J_N \rangle$, where $J_1', J_2', \ldots, J_K'$ is a permutation of $J_1, J_2, \ldots, J_K$. From Lemma 2.2, we know that S' is also optimal, i.e., $T_K^{(S)} = T_K^{(S')}$. For S', it has to be the case that $R(J_1') \leq R(J_2') \leq \ldots R(J_K') \leq R(J_{K+1}) \leq \ldots \leq R(J_N)$. Otherwise, according to Theorem 2.3, we can swap two adjacent jobs to reduce $T_K^{(S')}$. Then, S' is the schedule defined in Theorem 2.4. $\square$

Note that with Theorem 2.4 we can compute an optimal schedule in $O(M_{max}N + N\log N)$ time, where $M_{max} = \max_{i=1}^{N} M_i$ is the maximum number of stages of all jobs. We call this algorithm K-RANK.

3. SIMULATION RESULTS

In this section we provide numerical results showings that our algorithm performs well as compared to other algorithms with respect to different N and K values. The experiments are carried out as follows. (i) Job sizes and probabilities $x_{i,k}$, and $p_{i,k}, i = 1, \ldots, N, k = 1, \ldots, M_i$ are generated for all N jobs according to a uniform distribution [1]. (ii) Using the stage probabilities we determine at which stage each job completes (or terminates). (iii) We compute job schedules using that algorithm and then compute the resulting sojourn time of the K-th successful job under (ii); the results are taken as output of one trial. (iv) We repeat (i) through (iii) at least 1000000 times for N. Then we use the results of these trials to compute the average sojourn time of the K-th successful job under each schedule.

To better demonstrate the performance of our approach and to demonstrate the need to tailor a scheduling algorithm for successful jobs (as compared to all jobs), we also include the following three baseline algorithms in our experiments: 1) **RANDOM**: Execute jobs in random order; 2) **SERPT**: Sort and execute jobs based on expected remaining processing time, in ascending order, and 3) **SR**: Sort jobs based on SR rank [7] (equivalent to Gittins' index) and schedule them in ascending rank order. Note that, by design, RANDOM will execute each job until success or failure (i.e., without preemption) before selecting a different job for execution. This is reasonable since we know from Theorem 2.1 that preemption of jobs only increases expected sojourn time of successful jobs. In addition, including SR in the experiments allows us to demonstrate whether such an approach can deliver good performance for our objective.

Experiments for $K = 1$ and 2 can be found in Fig. 1(a) and (b), respectively, where we observe that our approach outperforms all baselines in both cases, e.g., for $N = 50$, it out performs the second best (SERPT) by about 30%.

Intuitively, larger K values mean less room for optimization. To evaluate the performance of our approach when K is large, we also experimented with $K = N/2$ with results presented in Fig. 1(c), where we observe that K-RANK reduces mean sojourn time of the K-th successful job by about 11% for all N values, as compared to the second best.

[1]We obtained similar results with other distributions

4. CONCLUSIONS

In this paper we studied the problem of scheduling multi-stage jobs with early termination, optimizing expected sojourn time of the K-th successful job and showed that when there is only one server, preemption is not needed. We presented an optimal scheduling algorithm and quantified the improvements obtained through simulations. Our future directions in this early termination setting include extending results to 1) a multi-server setting and 2) a broader set of simulations with more realistic workloads.

5. REFERENCES

[1] S. Aalto, U. Ayesta, and R. Righter. Properties of the Gittins index with application to optimal scheduling. *Probability in the Engineering and Informational Sciences*, 25(3):269–288, 2011.

[2] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, M. Sterna, and J. Weglarz. *Handbook on scheduling: From theory to practice*. Springer, 2019.

[3] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, 7(1):1–17, 1978.

[4] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

[5] Y. LeCun, Y. Bengio, and G. E. Hinton. Deep learning. *Nat.*, 521(7553):436–444, 2015.

[6] M. Pinedo. *Scheduling*, volume 29. Springer, 2012.

[7] K. C. Sevcik. Scheduling for minimum total loss using service time distributions. *Journal of ACM*, 21(1):66–75, 1974.