How to Use Predictions? Dependence-Aware Scheduling with Runtime Distributions

Anonymous authors
Paper under double-blind review

Abstract

Non-clairvoyant schedulers choose which task to run before its runtime is known. Runtime histories can help estimate a runtime distribution for each task. Using these distributions separately misses what a completed task's runtime reveals about unfinished tasks. When runtimes are dependent, that information can change which task should run next. This raises the question: how should a scheduler use a predicted runtime distribution to choose its next action as tasks complete? We introduce a scheduler that updates a predicted joint runtime distribution using completed runtimes and the time unfinished tasks have already run. It then chooses the action with the lowest expected remaining workflow completion cost. We show that ignoring runtime dependence can increase expected total workflow completion time. Both predictions have the correct runtime distribution for each task and are passed to the same exact scheduling algorithm. We bound how prediction error affects cost relative to an optimal scheduler that knows the true distribution. From h complete histories, *Truncated Joint Fitting (TJF)* caps long runtimes while preserving task pairings. For fixed model parameters and a bounded second moment of total runtime, TJF attains $O(h^{-1/n})$ expected excess cost for $n \geq 3$ under arbitrary dependence among task runtimes. A matching worst-case lower bound shows this exponent is unavoidable, while fitting tasks separately under independence attains $O(\sqrt{\log h/h})$. In a constructed four-task offline replay using recorded Alibaba runtimes, an uncapped empirical joint prediction reduces total workflow completion time by 13.94% across 1,796 held-out batches relative to independent prediction with the same task marginals.

1 Introduction

Non-clairvoyant scheduling is a classical problem in which tasks must be scheduled without knowing their runtimes (Motwani et al., 1994). Without runtime information, a scheduler may give resources to a long task while shorter tasks wait, which can increase their completion times. To reduce these delays, the scheduler can use runtime records from previous executions to estimate how long tasks take (da Silva et al., 2015; Park et al., 2018). Learning-augmented scheduling uses these estimates to guide scheduling decisions and studies how its performance depends on prediction accuracy (Mitzenmacher, 2020; Wei & Zhang, 2020). Predictions may provide runtime information for only some jobs or specify a job order learned from historical samples (Benomar & Perchet, 2024; Lindermayr & Megow, 2022). Runtime distributions provide another way to describe uncertainty by assigning probabilities to runtimes. Recent analyses study scheduling with such distributions learned from data or supplied as imperfect predictions (Ramakrishna et al., 2026; Moseley et al., 2025). A simple way to use it is to keep each task's predicted runtime distribution but assume the runtimes are independent when scheduling. This removes the information a completed task's runtime provides about unfinished tasks. Losing that information can change the preferred next action.

To illustrate how knowledge of runtime dependence can change the next task choice and reduce scheduling cost, consider the example in Figure 1. A workflow typically consists of one or more tasks; this example uses three single-task workflows on two processors, with task runtimes x_C, x_A, x_B. *Cost* is the sum of workflow completion times, and time is measured in discrete units called *ticks*. Task C uses one processor and takes one or two ticks with equal probability. Tasks A and B each require two processors and wait for C to finish. Preempting A or B takes one tick to save

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

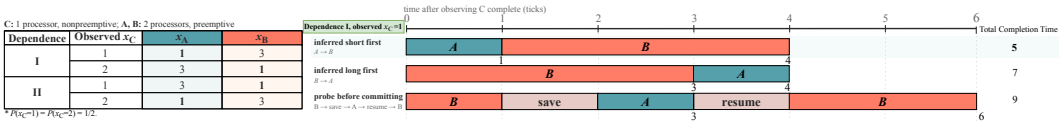

Figure 1: Toy example: three single-task workflows on two processors. The two dependence structures have identical task runtime distributions but imply opposite orders after observing x_C. Under Dependence I with $x_C = 1$, completion times measured from C's completion sum to 5 for A first, 7 for B first, and 9 for probing B before switching. Saving and resuming each take one tick, so switching costs more than continuing B.

its progress, and resuming it takes another tick. The two dependence structures have the same task runtime distributions and the same joint distribution of (x_A, x_B), but reverse how x_C determines the runtimes of A and B. An algorithm that retains the joint distribution uses the observed x_C to identify the shorter task and run it first. With zero startup delay, the sum of completion times measured from C's completion is 5 when the shorter task runs first and 7 in the reverse order. Probing the longer task for one tick before switching gives 9. Each workflow's completion time includes C's runtime, so C adds $3 \times 1.5 = 4.5$ to the expected cost. With the runtime dependence known, the expected cost is $4.5 + 5 = 9.5$. The algorithm using the same task marginals but assuming independent runtimes does not update its predictions for A and B after observing x_C. With the stated tie-breaking, its expected cost under either dependence structure is $4.5 + (5 + 7)/2 = 10.5$. The cost difference comes from using C's runtime to choose the order of A and B. This raises a broader question:

How should a scheduler use a predicted joint runtime distribution to make scheduling decisions?

Our contributions. We give a scheduling algorithm for a predicted joint runtime distribution. At each decision point, it updates the distribution using completed runtimes and unfinished-task ages, then chooses the action with the lowest expected remaining workflow completion cost under that distribution (Section 3.2). For competing terminal tasks, our algorithm has lower cost with the true joint distribution than with the same true task marginals treated as independent, provided predecessor runtimes reverse their conditional mean order (Theorem 1). We bound the algorithm's excess cost relative to the best policy with the true distribution and the same feedback in terms of prediction error (Proposition 3). With fixed model parameters and a bounded second moment of total runtime, *Truncated Joint Fitting* (TJF) achieves $O(h^{-1/n})$ expected excess cost from h independent complete histories under arbitrary dependence for fixed $n \geq 3$; a fixed-graph lower bound matches this exponent (Theorems 4 and 5). Under independent runtimes and the same moment and parameter conditions, fitting each task separately achieves $O(\sqrt{\log h/h})$ (Theorem 10). In a constructed four-task offline replay using recorded Alibaba runtimes, an uncapped empirical joint prediction reduces total workflow completion time by 13.94% across 1,796 held-out batches relative to an independent prediction with the same task marginals under the same algorithm (Section 5).

2 Related work

Non-clairvoyant scheduling algorithms make decisions without advance knowledge of task runtimes and are evaluated against an optimal clairvoyant schedule (Motwani et al., 1994). Learning-augmented scheduling adds predictions to improve performance when they are accurate while retaining worst-case guarantees (Wei & Zhang, 2020). For single-machine non-clairvoyant scheduling, Purohit et al. use predicted job sizes to design scheduling algorithms, while Im et al. study how prediction error relates to their guarantees (Purohit et al., 2018; Im et al., 2021). The information supplied to the scheduler can vary in both its representation and its availability. For representation, permutation predictions encode job orders instead of runtime values, with extensions to weighted jobs and unrelated machines and guarantees for learning the predictions (Lindermayr & Megow, 2022). For availability, partial predictions supply only some job sizes in advance, while progress bars provide information during execution (Benomar & Perchet, 2024; Benomar et al., 2025). Learning-augmented scheduling also studies online precedence constraints, where jobs and their dependencies become known only after all predecessors finish (Lassota et al., 2023). Predictions can also be updated as execution proceeds: explicit predictors combine past instances with observations from the current execution to revise their predictions (Eliáš et al., 2024).

Queueing analyses quantify the cost of using predicted job sizes (Mitzenmacher, 2020) and derive response-time guarantees under multiplicative estimation errors (Scully et al., 2022). At the distribution level, Robust Gittins gives error-dependent guarantees for independent job sizes with finite support (Moseley et al., 2025). Stochastic configuration balancing also uses known distributions for independent requests to design allocation policies for expected maximum load (Eberle et al., 2023). When runtime distributions are unknown, single-machine scheduling algorithms learn type-specific means online for independent exponential runtimes, with excess-cost bounds against a scheduler with known means (Merlis et al., 2023). With historical samples, Empirical Gittins gives finite-sample guarantees for mean response time in queues with Poisson arrivals and independent, identically distributed job sizes (Ramakrishna et al., 2026). Our scheduling policy uses a predicted joint runtime distribution to choose actions on a fixed execution graph, updating its decisions from completed runtimes and the elapsed time of unfinished tasks. We bound its expected excess workflow completion cost relative to an optimal policy with the true distribution and the same feedback in terms of distribution error. Under a bounded second moment, we also give upper and worst-case lower bounds on the number of complete histories needed to achieve a target excess cost.

3 Scheduling model and policy construction

3.1 Problem definition

A simple scheduler takes the predicted runtime distribution, keeps each task's marginal distribution, and assumes the task runtimes are independent. The scheduler can update a task's remaining-runtime prediction from its own age, but not from another task's feedback. Under dependence, the missing update can change which action minimizes expected completion cost, even when every task marginal is exact. We compare these two inputs using the same scheduling algorithm.

At time zero, $n \geq 1$ tasks in $w \geq 1$ nonempty workflows share K identical processors. A known directed acyclic graph specifies precedence, including cross-workflow edges; each workflow finishes when all its tasks complete. Task i requires exactly k_i processors, $1 \leq k_i \leq K$, with no speedup from more. Its runtime x_i is a positive integer, and age a_i counts executed ticks. Each tick, legal joint actions start ready tasks, execute, save or hold preemptible tasks, resume saved tasks, or wait under capacity and precedence. Startup has a known nonnegative integer duration; saving and resuming have known positive integer durations. Saving preserves progress and releases processors; operations occupy k_i processors without advancing age and run to completion. Nonpreemptible tasks execute continuously after startup until completion.

The runtime vector $\boldsymbol{x}$ follows an unknown joint distribution P; the scheduler receives a finite-support predicted joint distribution Q. Completion reveals x_i, while noncompletion at age a_i reveals $x_i > a_i$; operations reveal no runtime information. Policies may randomize. In the learning setting, Q is fitted from h independent histories, each recording all n runtimes as one vector; the new execution is independent of these histories. We minimize total workflow completion time, $C = \sum_{a \geq 1} w(a)$, where $w(a)$ counts unfinished workflows at the start of tick a. For fixed $\boldsymbol{x}$, $C_\pi(\boldsymbol{x})$ averages over policy randomization. Let $C_{\min}$ be the infimum expected cost among legal policies knowing P but receiving the same feedback and no advance access to $\boldsymbol{x}$. We define excess cost as follows:

$$\Delta C = \mathbb{E}C_\pi - C_{\min}. \tag{1}$$

When Q is learned, the expectation also includes histories. Write $X = \sum_i x_i$; a_{op} totals all startup durations and the save and resume durations of preemptible tasks. Theorem 1 and Proposition 3 require $\mathbb{E}_P X < \infty$; finite-history bounds assume $\mathbb{E}_P X^2 \leq M$.

3.2 Scheduling with a predicted joint distribution

Given Q, the scheduler conditions it on completed runtimes and $x_i > a_i$ for unfinished tasks. State s records task status, operation countdowns, and runtime feedback; s' is its successor, and $w(s)$ counts unfinished workflows. Let $C_{\min}(s)$ be the minimum expected remaining cost under Q, with value zero after workflows finish. The scheduler selects a legal joint action using the Bellman equation:

$$C_{\min}(s) = \min_{\text{legal joint actions}} \big\{ w(s) + \mathbb{E}_Q[C_{\min}(s') \mid s, \text{action}] \big\}.$$

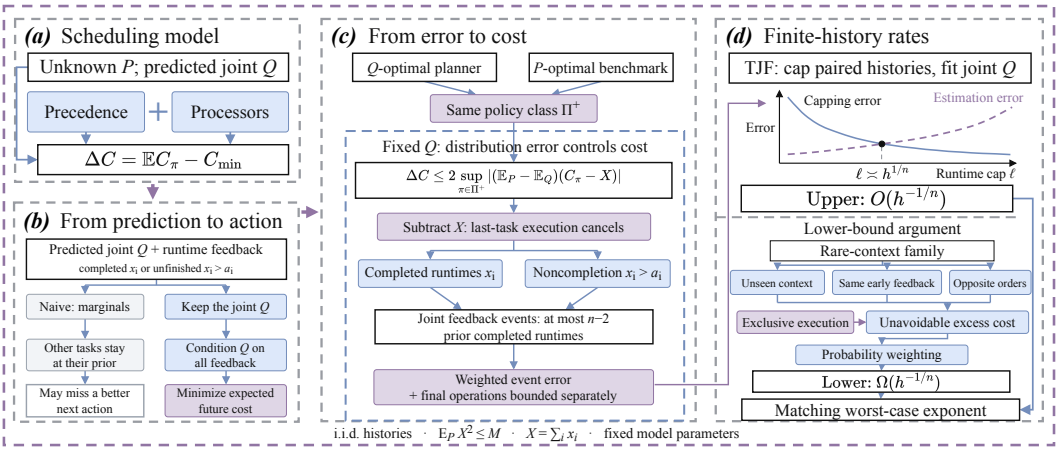

Figure 2: From predicted runtimes to scheduling guarantees. **(a)** Excess cost. **(b)** Independent runtimes leave other tasks' predictions unchanged after feedback; conditioning joint Q can change the action. **(c)** Q- and P-optimal policies share a class. Subtracting X exposes joint feedback events whose distribution error bounds excess cost. **(d)** Truncated Joint Fitting (TJF) balances capping and estimation, giving $O(h^{-1/n})$ under i.i.d. histories, $\mathbb{E}_P X^2 \le M$, fixed parameters, and $n \ge 3$; a rare-context lower bound matches the exponent.

The continuation includes later feedback and decisions through the end of the execution. Appendix B.1 specifies tie-breaking and the fallback for feedback outside Q's support. Exact planning may be exponential in n; scheduling cost excludes computation.

To obtain Q from complete histories, independent fitting uses untruncated empirical task marginals q_i and sets $Q = \prod_i q_i$. Truncated Joint Fitting (TJF) instead caps each historical runtime at $\max\{1, \lfloor h^{1/n} \rfloor\}$ and fits the empirical joint distribution of the paired vectors. The cap limits the fitted range and introduces truncation error; actual execution is uncapped. A supplied statistical graph specifies which runtimes may depend on others without changing task precedence.

Definition 1 (Statistical graph)**.** *A distribution is r-structured if a supplied directed acyclic graph on the tasks has maximum indegree r and $P(\boldsymbol{x}) = \prod_{i=1}^{n} f_i(x_i \mid$ statistical parent runtimes of task $i)$.*

Fitting these factors from complete histories gives $\Delta C = O(h^{-1/(r+4)})$ at fixed parameters under $\mathbb{E}_P X^2 \le M$ by Theorem 18. Appendix B.1 gives the fitting rule, and Appendix B.8 gives the proof. Here $r = 0$ gives independence, while $r = n - 1$ permits every joint distribution over the runtimes of all n tasks.

4 Theoretical Guarantees and Proofs

We first show when runtime feedback improves a scheduling decision, then quantify how many complete histories are needed to learn it.

4.1 The Value of Joint Runtime Information

Let f_i be task i's true marginal and let $Q = \prod_i f_i$ treat runtimes as independent. The algorithm's Q-optimal policy π thus has exact task marginals; π^* is optimal under P. Both are evaluated under P. For finite-support marginals, the algorithm of Section 3.2 supplied with Q produces π; Appendix B.2 gives the policy equivalence.

Consider $m \ge 2$ nonpreemptible terminal tasks in distinct workflows. They are pairwise incomparable, have equal startup durations, and satisfy $k_j + k_l > K$ for every pair. All other tasks form I and precede each terminal task. Let T be the last completion tick in I, or zero if $I = \varnothing$. At $T + 1$, only terminal tasks remain and $\boldsymbol{x}_I$ is known. Their remaining cost is $\sum_{a>T} w(a)$. Policy π orders them by unconditional means, while π^* can use means conditional on $\boldsymbol{x}_I$. Assume $\mathbb{E}_P X < \infty$.

Theorem 1 (Gain from joint runtime information)**.** *Under these conditions, the following inequalities hold:*

$$\begin{aligned}\mathbb{E}_P C_\pi - C_{\min} &\geq \mathbb{E}_P\left[\sum_{a>T} w(a)\right]_\pi - \mathbb{E}_P\left[\sum_{a>T} w(a)\right]_{\pi^*} \\ &\geq \frac{1}{2}\sum_{j<l}\Big(\mathbb{E}_P\big|\mathbb{E}_P[x_j - x_l \mid \boldsymbol{x}_I]\big| - \big|\mathbb{E}_P(x_j - x_l)\big|\Big).\end{aligned} \tag{2}$$

Under $\mathbb{E}_P X^2 \leq M$, TJF trained on h histories has an expected cost advantage over π of at least the displayed gap minus its finite-history excess-cost bound in Theorem 4. The gap is positive exactly when some pair's conditional mean difference takes both signs; TJF improves on π once its error bound is smaller.

Proof. Waiting at $T+1$ reveals no feedback, and equal startup durations cancel across orders. An order o has runtime-dependent cost $\sum_{q=1}^{m}(m-q+1)x_{o(q)}$. Under Q, predecessor feedback leaves terminal means unchanged. Adjacent exchange makes π sort by $\mathbb{E}_P x_j$, giving the minimum fixed-order cost:

$$\min_o \sum_{q=1}^{m}(m-q+1)\mathbb{E}_P x_{o(q)} = \sum_{j=1}^{m}\mathbb{E}_P x_j + \sum_{j<l}\min\{\mathbb{E}_P x_j, \mathbb{E}_P x_l\}. \tag{3}$$

This order is feasible under P because every P-possible runtime vector is Q-possible. After observing $\boldsymbol{x}_I$, sorting conditional means is feasible. The tower property cancels the single-task terms, and $\min(u,v) = (u+v-|u-v|)/2$ gives the second inequality. The P-optimal continuation does at least as well, while its prefix costs no more than π's: both prefixes reach the same terminal state and see the same distribution of $\boldsymbol{x}_I$. For $m=2$, no later order choice remains and the second inequality is exact. Jensen's inequality gives the sign condition; Theorem 4 gives the TJF claim. □

Independent fitting estimates the marginals from h histories. The next theorem shows how much of the joint-information gain remains when those marginals are learned.

Theorem 2 (Independent fitting under dependence)**.** *Under the conditions of Theorem 1, independent fitting from $h \geq 1$ complete histories satisfies:*

$$\begin{aligned}\Delta C \geq \frac{1}{2}\sum_{j<l}\Big(\mathbb{E}_P\big|\mathbb{E}_P[x_j - x_l \mid \boldsymbol{x}_I]\big| - \big|\mathbb{E}_P(x_j - x_l)\big|\Big) \\ - (m-1)\mathbb{E}_P\left[\left(\sum_{j=1}^{m} x_j\right)\sum_{i=1}^{n}\big(1 - f_i(x_i)\big)^h\right].\end{aligned} \tag{4}$$

The subtracted term tends to zero as $h \to \infty$. If $m = 2$ and I is a nonpreemptible chain, ΔC converges to the first term.

Proof. Condition on the histories. A comparison policy follows independent fitting through I and then orders terminal tasks by fitted means. The exchange argument of Theorem 1 gives its expected excess at least the first term. Independent fitting differs only if a realized x_i is absent from an empirical marginal; changing the terminal order can then save at most $(m-1)\sum_j x_j$. Given $\boldsymbol{x}$, the probability that x_i is unseen is $(1-f_i(x_i))^h$. A union bound gives the correction, which vanishes by dominated convergence since $f_i(x_i) > 0$ almost surely and $\mathbb{E}_P X < \infty$. For $m=2$ with a nonpreemptible chain I, the prefix is fixed; the empirical marginals converge almost surely and equation 6 gives convergence to the best fixed-order cost, making the bound exact. □

Preemption also introduces switching decisions. Consider tasks A, B in separate workflows, each preemptible and requiring all $K=2$ processors. Startup takes zero ticks; saving and resuming take one tick each. Under P, $(x_A, x_B) = (1,1)$ with probability 0.8 and $(9,9)$ otherwise. If A remains unfinished after one tick, π still assigns probability 0.8 to B being short. It saves A, completes B, and resumes A; π^* infers $B=9$ and finishes A first. In the long case, π completes B, A at ticks $11, 20$, while π^* completes A, B at ticks $9, 18$; both cost 3 in the short case. Thus $\mathbb{E}_P C_\pi = 8.6$ and

$C_{\min} = 7.8$. Here $x_A = x_B$ always, so the conditional-mean expression is zero; the 0.8 gain comes from switching, and Theorem 4 guarantees a TJF gain when its excess-cost bound is below 0.8. The right-hand side of equation 13 is also zero in this example.

4.2 From distribution error to scheduling cost

Fix the histories and let π be optimal for fitted Q. The class Π^+ rules out preemptible tasks holding processors without execution and resuming only to save again before any task executes. Appendix B.2 proves that both π and a P-optimal policy belong to this class.

Proposition 3 (Cost of distribution error). *If $\mathbb{E}_P X < \infty$, the policy π satisfies:*

$$0 \le \mathbb{E}_P C_\pi - C_{\min} \le 2 \sup_{\pi' \in \Pi^+} \left| (\mathbb{E}_P - \mathbb{E}_Q)(C_{\pi'} - X) \right|. \tag{5}$$

Subtracting policy-independent X exposes the evaluation error; the factor 2 covers both π and a P-optimal policy. The bound also holds with $C_{\pi'}$ in place of $C_{\pi'} - X$.

Proof. Choose $\pi^* \in \Pi^+$ with $\mathbb{E}_P C_{\pi^*} = C_{\min}$. Since π minimizes expected cost under Q, the middle term below is nonpositive:

$$\begin{aligned}
\mathbb{E}_P C_\pi - C_{\min} &= (\mathbb{E}_P - \mathbb{E}_Q)(C_\pi - X) + (\mathbb{E}_Q C_\pi - \mathbb{E}_Q C_{\pi^*}) + (\mathbb{E}_Q - \mathbb{E}_P)(C_{\pi^*} - X) \\
&\le |(\mathbb{E}_P - \mathbb{E}_Q)(C_\pi - X)| + |(\mathbb{E}_P - \mathbb{E}_Q)(C_{\pi^*} - X)| \\
&\le 2 \sup_{\pi' \in \Pi^+} |(\mathbb{E}_P - \mathbb{E}_Q)(C_{\pi'} - X)|.
\end{aligned}$$

The definition of $C_{\min}$ gives the lower bound. Omitting X gives the stated variant. □

For truncation, policies in Π^+ satisfy the pathwise envelope:

$$0 \le C_{\pi'}(\boldsymbol{x}) \le w(4a_{\text{op}} + 1)X(\boldsymbol{x}), \qquad \pi' \in \Pi^+. \tag{6}$$

Appendix B.7 represents $C_{\pi'} - X$ using completed runtime records and noncompletion events $x_i > a_i$, making distribution error a weighted sum of joint-event errors. The supremum precedes averaging over histories because Q and π depend on them.

4.3 Finite-history guarantee

TJF fits the empirical joint distribution of capped history vectors, preserving the task pairings. The following result quantifies its excess cost as history grows.

Theorem 4 (Arbitrary dependence). *Suppose $\mathbb{E}_P X^2 \le M$. For every $h \ge 1$, the policy learned by TJF satisfies the following bounds for $n \ge 3$ and $n = 2$, respectively:*

$$\Delta C \le \left[8\{w(4a_{\text{op}} + 1) + 1\}M + 2c\sqrt{M} \right] h^{-1/n}, \tag{7}$$

$$\Delta C \le \left[8\{w(4a_{\text{op}} + 1) + 1\}M + 2c\sqrt{M\{1 + \log(2\sqrt{h} + 1)\}} \right] h^{-1/2}. \tag{8}$$

For $n \ge 3$, c is specified in Lemma 17. For $n = 2$, $c = (2a_{\text{op}} + 1)(w + n)(c_2 + 2\sqrt{2}c_1)$, with c_1, c_2 from Lemma 16 in Appendix B.7. For $n = 1$, $\Delta C = 0$.

Proof. For an integer cap $\ell \ge 1$, capping preserves the second-moment bound. The feedback-event representation leaves at most $n - 2$ completed runtimes before the final-operation term; weighted Cauchy–Schwarz controls that term as well. Appendix B.7 bounds the uniform empirical error over Π^+ by $c\sqrt{M\ell^{n-2}/h}$ for $n \ge 3$. The vectors differ after capping only if $X > \ell$. The envelope equation 6 and $\mathbb{E}_P[X \mathbb{1}_{\{X > \ell\}}] \le M/\ell$ bound the corresponding error in $C_\pi - X$ by $2\{w(4a_{\text{op}} + 1) + 1\}M/\ell$. Proposition 3 combines both errors:

$$\Delta C \le \frac{4\{w(4a_{\text{op}} + 1) + 1\}M}{\ell} + 2c\sqrt{\frac{M\ell^{n-2}}{h}}. \tag{9}$$

Choose $\ell = \max\{1, \lfloor h^{1/n} \rfloor\}$ and use $h^{1/n}/2 \le \ell \le h^{1/n}$ to obtain equation 7. For $n = 2$, the inverse-age sum is at most $1 + \log(2\ell + 1)$; taking $\ell = \max\{1, \lfloor \sqrt{h} \rfloor\}$ gives equation 8. A single task is scheduled optimally pathwise. □

When runtimes are independent and $\mathbb{E}_P X^2 \le M$, fitting empirical task marginals gives $\Delta C = O(\sqrt{\log h/h})$ at fixed parameters by Theorem 10 in Appendix B.4.

4.4 A MATCHING WORST-CASE EXPONENT

For $n \ge 3$, the $h^{-1/n}$ exponent is unavoidable under arbitrary runtime dependence, even with a fixed workflow graph.

Theorem 5 (Fixed-graph lower bound)**.** *For every $n \ge 3$, there is an n-task graph with three workflows on two processors whose structure and operation durations are independent of h. For every $h \ge 1$ and every learner given h complete histories, some runtime distribution on this graph satisfies:*

$$\Delta C \ge \frac{1}{16h^{1/n}}, \qquad \mathbb{E}_P X^2 \le n^2 + 2(n+1)^2. \tag{10}$$

The distribution may depend on h and the learner. The learner may randomize, probe, and preempt.

Proof. Set $x = \lceil h^{1/n} \rceil + 1$. The graph has an $(n-2)$-task chain before two competing successors. Give each of $(x-1)^{n-2}$ rare chain contexts probability $p = 1/[2(x-1)^n]$; otherwise all runtimes are one. Independently for each rare context, choose successor lengths $(x, 2x)$ or $(2x, x)$ with equal probability, then fix these choices across executions. Fix the learner's randomization. A context is absent from the histories with probability at least $1 - hp \ge 1/2$. At such a context, the successor assignments remain equally likely. If the first successor to receive x execution ticks is long, their completion times sum to at least $5x$, against $4x$ for the comparator; averaging the assignments gives excess at least $x/2$, even with probing or preemption. Averaging over contexts yields:

$$(x-1)^{n-2} p \, \frac{x}{4} = \frac{x}{8(x-1)^2} \ge \frac{1}{16h^{1/n}},$$

where $x - 1 \le 2h^{1/n}$. Ordinary runs have $X = n$; rare runs have $X \le (2n+2)(x-1)$ and total probability $1/[2(x-1)^2]$, giving the stated second-moment bound. Some fixed assignment attains the average. Appendix B.6 gives the full construction. □

For fixed parameters and $n \ge 3$, Theorems 4 and 5 match the worst-case $h^{-1/n}$ exponent. Under the stated moment bounds, $O(\varepsilon^{-n})$ histories suffice for excess $\varepsilon > 0$, and the hard family requires at least $(16\varepsilon)^{-n}$.

5 EXPERIMENTAL STUDY

Section 4.1 shows that completed predecessor runtimes can change which remaining task should run first. Whether a prediction fitted from recorded histories preserves enough of this information to improve scheduling is an empirical question. We characterize three audited workflow cohorts and build four-task replay batches from a separate Alibaba cohort. In these batches, two terminal tasks compete for one processor after their predecessors finish, making the ordering choice after feedback explicit. We compare an uncapped joint prediction with an independent prediction sharing its task marginals and with a fixed order chosen before feedback. We then weaken the dependence retained in the prediction and evaluate prescribed TJF across history sizes, measuring forecast error, completion time, and online decision time (Figure 3 and Table 1).

5.1 SETUP

Recorded workflows. Figure 3(a) reports duration ranges, batch counts, and chronological history, development, and test splits for three audited workflow cohorts. Shell Chronos[1] provides 3,402 complete three-task chains. Alibaba 2018 (Alibaba, 2018) provides 8,738 three-task chains and 1,860 three-task joins after matching tasks to completed instances. Alibaba runtime labels are instance start-to-end intervals, which need not equal the corresponding CPU service time.

[1] https://doi.org/10.5281/zenodo.3254576

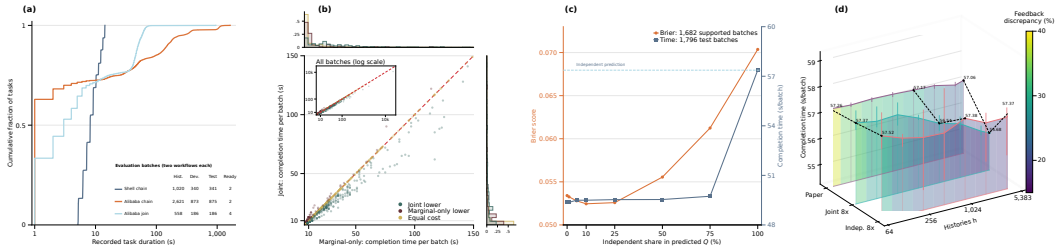

Figure 3: Real-trace evaluation. **(a)** Task-duration distributions and batch counts in three audited workflow cohorts. **(b)** Per-batch completion time under joint and independent predictions with shared task marginals; colors mark the lower-cost prediction. **(c)** Forecast error and completion time as joint information is replaced by independent marginals. **(d)** History-size results for the prescribed TJF and exploratory eightfold caps; color encodes completed-predecessor feedback discrepancy, and bars span five draws. Panels (b–d) replay constructed four-task Alibaba batches.

Shared scheduling choice. Airflow supports cross-workflow task dependencies and limits concurrency through shared pools (Apache Software Foundation, 2025a;b). The comparison uses a separate Alibaba cohort of two-task chains. Of 39,977 candidates, 22,007 lack a valid unique instance match and 23 have inconsistent timing. Pairing adjacent eligible chains of the same task template by verified first-task instance start time yields 8,973 four-task batches, with one chain unpaired. In this replay, we keep each recorded within-chain edge and add two cross-chain edges, so both first tasks precede both terminal tasks. The terminal tasks then compete for one processor. Both predecessor runtimes are therefore observed before the terminal tasks are ordered. With one shared processor, that order affects the sum of the two workflow completion times. The added edges and processor limit are replay assumptions. Replay assumes simultaneous release, nonpreemptible tasks, zero startup time, and recorded durations unchanged under rescheduling. The chronological split gives 5,383 history, 1,794 development, and 1,796 test batches. We measure the sum of the two workflow completion times in seconds on the same test batches for every method.

5.2 Scheduling comparisons

Scheduling policies. Four-task batches let predecessor runtimes affect which terminal task runs first. We fit an uncapped empirical joint distribution Q from 5,383 histories. Independent prediction uses four marginals of Q but treats runtimes as independent. Both use the exact Bellman scheduling algorithm, tie breaking, and sequential fallback after unsupported feedback. A fixed-order policy chooses an expected-cost-optimal order under Q before observing runtimes. Joint versus independent prediction tests the value of predicted dependence; joint versus fixed order tests the value of updating after feedback. Independent prediction and fixed order choose the same terminal order.

Prediction sensitivity. We mix Q with its independent counterpart, keeping marginals and the scheduler fixed. The Brier score measures forecasts of terminal-task completion by ages 1, 2, 4, 8, 16, 32, and 64 seconds. It uses 1,682 batches with supported predecessor feedback for every mixture; completion time uses 1,796 batches. We fit ten joint predictions from disjoint 512-history subsets and compare each with the independent prediction from its marginals on the same test batches. Together, the Brier score and completion time separate forecast quality from the effect of scheduling decisions. The disjoint subsets then test how the comparison varies with the histories used to fit Q.

History size. TJF caps recorded runtimes at $\max\{1, \lfloor h^{1/4} \rfloor\}$ seconds before fitting the joint distribution. Five nested history draws cover eight sizes from 64 to 5,383; the full-history point repeats one fitted distribution. Each independent comparator uses the marginals of the corresponding capped joint prediction and the same scheduler. Feedback discrepancy is total variation between predicted and observed completed-predecessor pair frequencies across all test batches. We examine whether a wider cap lets more observed predecessor runtimes inform the terminal order. Figure 3(d) and Table 1 report an exploratory eightfold cap selected after inspecting the original test outcomes.

Table 1: History coverage and online overhead under the exploratory eightfold runtime cap. Values average five draws on 1,796 test batches. Fallback is feedback outside capped support; unseen feedback is absent from history.

Histories h	64	128	256	512	1,024	2,048	4,096	5,383
Runtime cap (s)	16	24	32	32	40	48	64	64
All-task coverage (%)	88.4	91.6	93.4	93.4	94.5	95.4	96.5	96.5
Fallback (%)	28.3	25.3	20.4	15.4	12.6	10.0	7.6	6.8
Unseen feedback (%)	28.4	25.2	20.3	15.2	12.2	9.7	7.2	6.3
Decision p95 (μs)	1.8	2.5	4.3	6.9	11.5	19.5	34.8	42.9
Peak RSS (MiB)	21.6	21.5	21.8	22.1	22.1	22.6	23.2	23.4

5.3 Results and discussion

Scheduling benefit. With the uncapped Q, joint prediction costs 88,666 seconds versus 103,032 seconds for independent prediction and fixed order, a 13.94% saving (Figure 3(b)). With Q fixed, the paired-batch 95% bootstrap interval for the signed joint-minus-independent difference is $[-26.79\%, -2.45\%]$. The predictions choose different terminal orders after feedback in 1,106 batches. A post-hoc check excluding one batch responsible for 11,188 of 14,366 seconds saved still finds a 3.95% saving against independent prediction on the remaining batches.

Prediction sensitivity. At 75% independent mixing, the Brier score rises from 0.05341 to 0.06124, while completion time rises by only 0.74% to 89,325 seconds (Figure 3(c)). Only 251 of 978 changed orders alter realized cost; the mixture still saves 13.30% against fully independent prediction. The Brier score improves at some small mixing weights and varies nonmonotonically. Across ten 512-history predictions, joint prediction costs less than its independent comparator in seven cases and more in three.

Finite histories and overhead. Under prescribed TJF, feedback discrepancy falls overall from 38.22% at $h = 64$ to 22.91% at $h = 5,383$, and fallback from 35.80% to 17.98% (Figure 3(d)). Savings against independent prediction with the same cap change from -0.04% to 0.54% between these endpoints, without a monotone trend. At full history, 323 batches account for 83% of comparator cost; their predecessor runtimes are absent from the fitted distribution, so fallback yields no saving. Selected after inspecting the original test outcomes, the exploratory eightfold cap raises runtime coverage from 78.95% to 96.55% and lowers fallback to 6.85% (Table 1). It costs 100,002 seconds versus 103,032 seconds for its independent comparator, saving 2.94% at full history; at $h = 256$, its mean cost is 0.43% higher than that comparator. At full history on macOS arm64, its 95th-percentile online decision time is 42.9 μs versus 21.9 μs for prescribed TJF. Online time excludes 8.63 ms of fitting and 0.32 ms of value precomputation. Without the true P, the trace cannot measure the policy-weighted error in Proposition 3 or the excess cost in Theorem 4.

6 Conclusion

In learning-augmented scheduling, our algorithm conditions a predicted joint runtime distribution on execution feedback to minimize expected remaining workflow completion cost. For competing nonpreemptible terminal tasks, we prove a strict gain over independence with the same true marginals when completed runtimes reverse their conditional mean order. We bound the additional scheduling cost due to distribution error. From h independent complete histories, TJF attains $O(h^{-1/n})$ expected excess cost for $n \geq 3$ under fixed model parameters and a bounded second moment of total runtime; a lower bound on a fixed graph matches this worst-case exponent. In a constructed four-task offline replay using recorded Alibaba runtimes, uncapped empirical joint prediction reduced total workflow completion time by 13.94% relative to independent prediction with the same marginals. A post-hoc check excluding the batch with the largest gain still finds a 3.95% saving against independent prediction on the remaining batches. With the prescribed TJF cap, the gain was 0.54% at 5,383 histories against independent prediction with the same capped marginals. The guarantee assumes exact optimization, whose computation is excluded from scheduling cost; obtaining comparable guarantees with faster algorithms remains open.

AI USE STATEMENT

Generative AI was used only to check the manuscript's structure and grammar.

REPRODUCIBILITY STATEMENT

Section 3.1 states the scheduling model and sampling assumptions, while Section 3.2 specifies the planning and fitting procedures. Appendix B.1 gives the tie-breaking rules, estimator details, and fallback behavior for feedback outside the fitted distribution's support. Appendix B contains the proofs and lower-bound constructions. Section 5 identifies the public traces and reports cohort construction, chronological splits, replay assumptions, comparison policies, and evaluation settings. It also identifies the exploratory cap selected after inspection of the original test outcomes.

REFERENCES

Alibaba. Alibaba Cluster Trace 2018. https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2018/trace_2018.md, 2018.

Apache Software Foundation. Cross-DAG dependencies. https://airflow.apache.org/docs/apache-airflow/2.10.5/howto/operator/external_task_sensor.html, 2025a. Apache Airflow 2.10.5 documentation.

Apache Software Foundation. Pools. https://airflow.apache.org/docs/apache-airflow/2.10.5/administration-and-deployment/pools.html, 2025b. Apache Airflow 2.10.5 documentation.

Ziyad Benomar and Vianney Perchet. Non-clairvoyant scheduling with partial predictions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 3506–3538. PMLR, 2024. URL https://proceedings.mlr.press/v235/benomar24a.html.

Ziyad Benomar, Romain Cosson, Alexander Lindermayr, and Jens Schlöter. Non-clairvoyant scheduling with progress bars. In *Advances in Neural Information Processing Systems*, volume 38, pp. 93520–93561. Curran Associates, Inc., 2025. doi: 10.52202/085713-3127. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/868f2266086530b2c71006ea1908b14a-Paper-Conference.pdf.

Dimitri P. Bertsekas and Steven E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Academic Press, 1978. URL https://faculty.engineering.asu.edu/bertsekas/books/stochastic-optimal-control-the-discrete-time-case/.

D. Bitouzé, B. Laurent, and P. Massart. A Dvoretzky–Kiefer–Wolfowitz type inequality for the Kaplan–Meier estimator. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 35(6): 735–763, 1999. doi: 10.1016/S0246-0203(99)00112-0. URL https://www.numdam.org/item/AIHPB_1999__35_6_735_0/.

Rafael Ferreira da Silva, Gideon Juve, Mats Rynge, Ewa Deelman, and Miron Livny. Online task resource consumption prediction for scientific workflows. *Parallel Processing Letters*, 25(3): 1541003, 2015. doi: 10.1142/S0129626415410030.

Franziska Eberle, Anupam Gupta, Nicole Megow, Benjamin Moseley, and Rudy Zhou. Configuration balancing for stochastic requests. In *Integer Programming and Combinatorial Optimization*, volume 13904 of *Lecture Notes in Computer Science*, pp. 127–141. Springer, 2023. doi: 10.1007/978-3-031-32726-1_10.

Marek Eliáš, Haim Kaplan, Yishay Mansour, and Shay Moran. Learning-augmented algorithms with explicit predictors. In *Advances in Neural Information Processing Systems*, volume 37, pp. 97972–98008. Curran Associates, Inc., 2024. doi: 10.52202/079017-3109. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b1a72c79ac7512df8d7b573f38143ac4-Paper-Conference.pdf.

David Haussler. Sphere packing numbers for subsets of the Boolean n-cube with bounded Vapnik–Chervonenkis dimension. *Journal of Combinatorial Theory, Series A*, 69(2):217–232, 1995. doi: 10.1016/0097-3165(95)90052-7. URL https://doi.org/10.1016/0097-3165(95)90052-7.

Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 285–294, 2021. doi: 10.1145/3409964.3461790.

Alexandra Anna Lassota, Alexander Lindermayr, Nicole Megow, and Jens Schlöter. Minimalistic predictions to schedule jobs with online precedence constraints. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 18563–18583. PMLR, 2023. URL https://proceedings.mlr.press/v202/lassota23a.html.

Alexander Lindermayr and Nicole Megow. Permutation predictions for non-clairvoyant scheduling. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 357–368. Association for Computing Machinery, 2022. doi: 10.1145/3490148.3538579. URL https://arxiv.org/abs/2202.10199.

Nadav Merlis, Hugo Richard, Flore Sentenac, Corentin Odic, Mathieu Molina, and Vianney Perchet. On preemption and learning in stochastic scheduling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24478–24516. PMLR, 2023. URL https://proceedings.mlr.press/v202/merlis23a.html.

Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. In *11th Innovations in Theoretical Computer Science Conference*, volume 151 of *Leibniz International Proceedings in Informatics*, pp. 14:1–14:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi: 10.4230/LIPIcs.ITCS.2020.14. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2020.14.

Benjamin Moseley, Heather Newman, Kirk Pruhs, and Rudy Zhou. Robust Gittins for stochastic scheduling. arXiv preprint arXiv:2504.10743, 2025. URL https://arxiv.org/abs/2504.10743.

Rajeev Motwani, Steven Phillips, and Eric Torng. Nonclairvoyant scheduling. *Theoretical Computer Science*, 130(1):17–47, 1994. doi: 10.1016/0304-3975(94)90151-1. URL https://doi.org/10.1016/0304-3975(94)90151-1.

Jun Woo Park, Alexey Tumanov, Angela Jiang, Michael A. Kozuch, and Gregory R. Ganger. 3Sigma: Distribution-based cluster scheduling for runtime uncertainty. In *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–17, 2018. doi: 10.1145/3190508.3190515. URL https://www.pdl.cmu.edu/PDL-FTP/CloudComputing/eurosys18-final49.pdf.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.neurips.cc/paper_files/paper/2018/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html.

Shefali Ramakrishna, Amit Harlev, and Ziv Scully. Empirical Gittins for data-driven M/G/1 scheduling with arbitrary job size distributions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 10(1):9:1–9:38, 2026. doi: 10.1145/3788091. URL https://amitharlev.com/assets/pdf/sigmetrics2026-ramakrishna.pdf.

Ziv Scully, Isaac Grosof, and Michael Mitzenmacher. Uniform bounds for scheduling with job size estimates. In *13th Innovations in Theoretical Computer Science Conference*, volume 215 of *Leibniz International Proceedings in Informatics*, pp. 114:1–114:30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi: 10.4230/LIPIcs.ITCS.2022.114. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2022.114.

Alexander Wei and Fred Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Advances in Neural Information Processing Systems*, volume 33, pp. 8042–8053. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/5bd844f11fa520d54fa5edec06ea2507-Abstract.html.

Jon A. Wellner. On an exponential bound for the Kaplan–Meier estimator. *Lifetime Data Analysis*, 13(4):481–496, 2007. doi: 10.1007/s10985-007-9055-z. URL https://doi.org/10.1007/s10985-007-9055-z.

A NOTATION

Table 2: Summary of notation.

Symbol	Meaning
Scheduling model and objective	
w	Number of workflows
n, m	Total number of tasks and number of competing terminal tasks in Section 4.1, respectively ($m \leq n$)
K	Number of identical processors
k_i	processors simultaneously required by task i
x_i	Runtime of task i
$\boldsymbol{x}$	Runtime vector
$X = \sum_i x_i$	Total runtime
a_i	Executed age of task i
a_{op}	Aggregate startup, save, and resume duration parameter
$C_\pi(\boldsymbol{x})$	Completion-time cost of policy π on runtime vector $\boldsymbol{x}$
$C_{\min}$	Minimum expected cost of a policy that knows P and receives the same execution feedback
ΔC	Expected excess cost, $\mathbb{E}C_\pi - C_{\min}$
Runtime distributions and histories	
P	True joint distribution of task runtimes
Q	Fitted finite-support joint distribution
f_i, F_i	True probability mass and cumulative distribution of task i
q_i, Q_i	Fitted probability mass and cumulative distribution of task i
h	Number of complete runtime histories
M	Second-moment bound, $\mathbb{E}_P X^2 \leq M$
Policies and proof classes	
π	Scheduling policy
Π	Normalized policy class used in the appendix
Π^+	Compressed policy subclass used for the learning comparison
Structured and censored extensions	
r	Maximum indegree of the supplied statistical graph
ℓ	Runtime truncation level
$x_i^{\max}$	Known runtime endpoint for task i in the censored-log setting
h_i	Number of censored logs for task i
$u(a), v(a)$	Completion and at-risk counts at age a in the Kaplan–Meier estimator

B THEORY DETAILS AND PROOFS

The policy reductions give the fitted policy and a P-optimal policy a common cost bound. We use this bound in the learning proofs; the scope of Theorem 1 is stated separately.

B.1 ADDITIONAL ALGORITHM DETAILS

Execution conventions. Capacity is checked at tick starts, counting resident tasks and operations; completed tasks release processors for the next tick. Zero-duration startup permits execution in the same tick. Readiness follows actual completion, including execution beyond fitted support. State s records physical task states, operation countdowns, unfinished ages, and completed runtimes. Resident preemptible tasks may execute, save, or hold their processors; nonpreemptible tasks execute continuously after startup, and operations run to completion.

Algorithm. Normal mode retains and renormalizes support vectors in Q that match completed runtimes and exceed unfinished ages. With $w(s)$ unfinished workflows and zero terminal cost, the

algorithm solves the Bellman equation:

$$C_{\min}(s) = \min_{\text{legal joint actions}} \{w(s) + \mathbb{E}[C_{\min}(s') \mid s, \text{action}]\}, \tag{11}$$

where the expectation is under Q. States are processed in decreasing total execution; deterministic operations within each layer form a positive-cost shortest-path problem. Ties are broken lexicographically by task index: execute before save before hold, wait before resume, and start before wait.

Incompatible feedback triggers a permanent switch to Processor release: finish ongoing operations, save preemptible tasks, complete active nonpreemptible tasks, and start none. Sequential completion then finishes ready tasks in index order. From any mode, exactly one remaining task triggers Single remaining task, which finishes required operations and executes continuously.

Held-out selection. For $h \geq 2$, train independent fitting and TJF on $\lfloor h/2 \rfloor$ histories and replay both policies on the rest using only execution feedback. Select the lower mean cost, break ties toward TJF, and freeze the policy without retraining.

Censored logs. Under Theorem 20, Kaplan–Meier fitting estimates each marginal and moves residual mass to its known endpoint $x_i^{\max}$. Let $u(a)$ count completions at age a and $v(a)$ count records observed through that age. The fitted CDF takes the form:

$$Q_i(a) = 1 - \prod_{a'=1}^{a} \left(1 - \frac{u(a')}{v(a')}\right), \quad 0 \leq a < x_i^{\max}, \qquad Q_i(x_i^{\max}) = 1, \tag{12}$$

with the ratio zero when the denominator is zero. The algorithm uses $Q = \prod_i q_i$, where $q_i(x) = Q_i(x) - Q_i(x-1)$.

Structured fitting. A context lists statistical parent runtimes in task-index order; true conditionals at zero-probability contexts place all mass at one. Truncate runtimes at $\ell = \max\{1, \lfloor h^{1/(r+4)} \rfloor\}$. Fit each conditional from histories with the same truncated context, using the pooled truncated marginal for unseen contexts. Multiplying the fitted conditional probabilities gives Q. Independent fitting and TJF give stronger guarantees at $r = 0$ and $r = n - 1$, respectively.

Computational cost. There are at most $\prod_i d_i$ physical states, with d_i defined below. Accounting for ages and completed runtimes multiplies this count by one factor per task, each equal to one plus that task's maximum fitted runtime. Exact rational computation may be exponential in n; scheduling cost excludes computation.

B.2 Common policy reductions

The physical state count first bounds idle duration; two pathwise modifications then give the sharper bound for Π^+. A task is unstarted, resident, complete, or, if preemptible, saved. Starting an operation consumes one tick, so its countdown states run from one to duration minus one. We count the physical states of task i as follows:

$$d_i = \begin{cases} 4 + \max\{\text{startup} - 1, 0\} + (\text{save} - 1) + (\text{resume} - 1), & \text{preemptible}, \\ 3 + \max\{\text{startup} - 1, 0\}, & \text{nonpreemptible}. \end{cases}$$

Thus there are at most $\prod_i d_i$ physical state combinations, excluding ages and completed runtimes. An idle tick contains no effective execution.

Definition 2 (Normalized policies)**.** *A policy in* Π *is a deterministic function of state and mode, defined at every state. Modes progress from Normal to Processor release to Sequential completion to Single remaining task. The last mode is entered directly from any mode when exactly one task remains. Modes after Normal follow Appendix B.1. No physical state repeats at tick starts during consecutive idle ticks in Normal mode.*

Lemma 6 (Cost envelope and optimal policies)**.** *Every* $\pi \in \Pi$ *finishes every runtime vector and satisfies the following cost bound:*

$$0 \leq C_\pi(\boldsymbol{x}) \leq w\left(8 \prod_{i=1}^{n} d_i - 1\right) X(\boldsymbol{x}).$$

For every P with $\mathbb{E}_P X < \infty$, some $\pi^* \in \Pi$ attains $C_{\min}$ among all legal policies, including randomized policies using the full history. Each fitting method in Section 3.2, together with the algorithm, returns a policy in Π optimal under Q. The Single remaining task rule completes the final task as early as any legal policy on every path.

Proof. The state determines the conditional runtime distribution, legal actions, transitions, and cost. It therefore defines a countable-state MDP with finitely many actions per state and positive cost on unfinished ticks (Bertsekas & Shreve, 1978).

Normal mode forbids idle state repetitions; later modes only complete required operations before the next execution. Modes move forward, giving at most $4\prod_i d_i - 1$ consecutive idle ticks. At most X ticks contain execution, so the following bound holds for total duration:

$$X + (X+1)\left(4\prod_i d_i - 1\right) \le \left(8\prod_i d_i - 1\right) X.$$

Multiplying by w gives the cost bound. The final-task rule performs only required operations and continuous execution, attaining earliest completion.

For $\mathbb{E}_P X < \infty$, Processor release followed by Sequential completion provides a finite-cost continuation at every positive-probability state. The Bellman equation equation 11, now under P, admits minimizing actions with the specified tie-breaking order. Positive per-tick cost gives almost-sure termination, and monotone convergence gives optimality. During idle transitions the optimal remaining value strictly decreases, so no state repeats. Use the fixed fallback at zero-probability observations and the earliest-completion rule for the final task. This yields an optimum in Π. For finite-support Q, the algorithm solves the same equations on finitely many compatible Normal states. □

Definition 3 (Compressed policies)**.** *The class $\Pi^+ \subseteq \Pi$ adds three rules in Normal mode. A resident preemptible task executes or starts saving. No task is resumed and saved again without any task executing between the starts of those operations. Normal mode is exited only after an execution tick, except that a single-task instance starts in Single remaining task mode.*

Lemma 7 (Two pathwise modifications)**.** *Fix a state and a deterministic legal policy π, possibly using the full history. If π keeps a resident preemptible task i idle for the current tick, a policy π_1 can execute i instead with $C_{\pi_1}(\boldsymbol{x}) \le C_\pi(\boldsymbol{x})$ for every runtime vector. If π resumes a saved task i and no task executes before it starts saving i again, a policy π_2 can keep i saved until that save would finish, copy all other actions, and satisfy $C_{\pi_2}(\boldsymbol{x}) = C_\pi(\boldsymbol{x})$ on every vector.*

Proof. Execution uses the same processors as idle residence. Simulate π with task i one execution unit ahead: noncompletion implies noncompletion at the simulated age, and completion reveals x_i. All observations can therefore be reconstructed, and successors become ready no later. No workflow finishes later.

For the second modification, no execution feedback occurs before saving starts. Keeping i saved frees processors and preserves all other actions and feedback. When the simulated save ends, both policies have the same state and pathwise cost. □

Lemma 8 (Idle duration and cost)**.** *For every $\pi \in \Pi^+$, consecutive idle ticks on any real or virtual path number at most $2a_{\mathrm{op}}$, including paths from arbitrary states. Every runtime vector finishes within $(4a_{\mathrm{op}} + 1)X$ ticks, and the corresponding cost satisfies the following bound:*

$$0 \le C_\pi(\boldsymbol{x}) \le w(4a_{\mathrm{op}} + 1)X(\boldsymbol{x}).$$

From any state, the remaining duration is at most $(2a_{\mathrm{op}} + 1)$ times the remaining effective execution volume plus $2a_{\mathrm{op}}$.

Proof. Every idle Normal tick starts or advances an operation; otherwise the physical state repeats. During an idle sequence, each task contributes at most one startup, one save, and two resume durations. The compression rules force execution after a completed resume unless it was already active when the sequence began. This gives $2a_{\mathrm{op}}$ ticks. Later modes perform only operations required before the next execution, with the same bound. The argument also holds for finite prefixes of virtual paths.

At most X ticks contain execution and at most $X + 1$ idle sequences occur. Hence, the following duration bound holds:

$$X + 2a_{\mathrm{op}}(X + 1) \le (4a_{\mathrm{op}} + 1)X.$$

Multiply by w for cost. Replacing X by remaining execution volume gives the continuation bound. □

Proposition 9 (Optimal and fitted policies are compressed)**.** *Use the specified tie-breaking order: execute, save, hold for a resident preemptible task; wait, resume for a saved task; and start, wait for an unstarted task. For every P with $\mathbb{E}_P X < \infty$, some $\pi^* \in \Pi^+$ attains $C_{\min}$. Each fitting method in Section 3.2, together with the algorithm, yields a policy in Π^+. Held-out selection chooses between the policies returned by independent fitting and TJF, which both belong to this class.*

Proof. Lemma 7 and the tie-breaking order remove idle residence and resume-then-save without increasing the optimal value. The mode construction in Lemma 6 gives the remaining compression rule. The same argument applies under Q, and selection preserves the chosen policy's rules. □

B.3 Scope of Theorem 1

Fixed orders and waiting. A fixed order may be chosen from training histories and an independent random seed. Conditioning on that choice gives the fixed-order cost in equation 3 or a larger one; averaging preserves the lower bound in equation 2. At a stage decision with no active task or operation, waiting changes neither the physical state nor the runtime information. A policy that waits can be coupled to one that skips these ticks while simulating the original policy's clock. The observation sequence is unchanged, and pathwise cost falls by the sum of $w(a)$ over skipped ticks. For $m = 2$, a nonwaiting policy that uses no pre-stage feedback selects its order before any terminal-task feedback, so it is a fixed-order policy.

Upper bound on stage gain. Sorting terminal tasks by their realized runtimes gives a pathwise lower bound on every legal continuation. Comparing this order with equation 3 yields the following stage-gain bound:

$$\begin{aligned} &\mathbb{E}_P\left[\sum_{a>T} w(a)\right]_{\pi} - \mathbb{E}_P\left[\sum_{a>T} w(a)\right]_{\pi^*} \\ &\quad\le \frac{1}{2}\sum_{j<l}\left(\mathbb{E}_P|x_j - x_l| - |\mathbb{E}_P(x_j - x_l)|\right). \end{aligned} \tag{13}$$

If $\boldsymbol{x}_I$ determines all terminal runtimes, the pathwise order is feasible, so equality holds.

When the stage upper bound covers total cost. If I is a precedence chain, waiting or saving delays all subsequent completions. Both policies therefore execute the chain continuously and incur the same prefix cost. If $\boldsymbol{x}_I$ has distribution $\prod_{i\in I} f_i$ under P, π minimizes expected prefix cost under the true prefix distribution because its stage value is independent of $\boldsymbol{x}_I$. In either case, equation 13 also bounds $\mathbb{E}_P C_\pi - C_{\min}$.

Startup times and later tasks. If startup durations differ, replace each x_j in the ordering argument by x_j plus its known startup duration. In particular, the conditional mean difference includes the difference between their known startup durations. The same comparison applies if every other task either precedes all m terminal tasks or follows all of them, and every later task has all m tasks as predecessors. Later workflows add the same charge for the terminal tasks' total duration under every order. All nonwaiting orders reach the same state at the same tick after the terminal tasks finish, so their later continuation value is independent of the order.

Adaptation beyond the pairwise gap. Let $I = \varnothing$, $m = 3$, $K = 1$, and all startup durations be zero. Let x_1 be 1 or 2 with equal probability, and set $(x_2, x_3) = (1, 5)$ when $x_1 = 1$ and $(5, 1)$ when $x_1 = 2$. The lower bound in equation 2 is zero. Policy π runs $(1, 2, 3)$ and costs 13.5, whereas running task 1 first and then the shorter remaining task costs 11.5. Thus a positive benefit need not require a positive pairwise gap when $m \ge 3$.

For the fixed-order qualification, take one earlier task C with edges to three terminal tasks and $K = 1$, zero startup, and nonpreemptible tasks. With probability $1/2$, $(x_C, x_1, x_2, x_3) = (1, 1, 3, 5)$; otherwise it is $(2, 2, 5, 3)$. The best fixed-order stage cost is $(14 + 19)/2 = 16.5$, and the lower bound in equation 2 is 1. A policy that ignores x_C can start task 1 and then use its completion time to choose between tasks 2 and 3. Its stage cost is $(14 + 17)/2 = 15.5$, equal to the optimal stage cost. The pairwise lower bound therefore applies to orders fixed before this feedback, not to such adaptive orders.

Preemption. The conditional-mean expression in equation 2 need not bound the gain from below, and equation 13 need not bound it from above, when terminal tasks are preemptible. In the instance below, tasks A and B belong to distinct single-task workflows, each occupies all K processors, startup is zero, and saving and resuming take one tick each.

For the lower bound, take $K = 1$ and an earlier one-processor, nonpreemptible task C in a third single-task workflow, with edges $C \to A$ and $C \to B$. The vectors (x_C, x_A, x_B) are $(1, 1, 3)$ with probability 0.9 and $(2, 9, 3)$ with probability 0.1. The true conditional policy runs the shorter task first and costs $0.9(5) + 0.1(15) = 6$. Policy π executes one tick of A and, if it has not finished, saves A, completes B, and resumes A. Its stage cost under P is $0.9(5) + 0.1(19) = 6.4$, giving an excess of 0.4, below the 0.6 lower bound in equation 2.

The example in Section 4.1 also has zero right-hand side in equation 13, while $\mathbb{E}_P C_\pi - C_{\min} = 0.8$.

B.4 Independent runtimes

Theorem 10 (Independent runtimes)**.** *Suppose task runtimes are mutually independent and $\mathbb{E}_P X^2 \le M$. For every $h \ge 1$, the policy learned by independent fitting satisfies the following bound:*

$$\Delta C \le \frac{2w(4a_{\mathrm{op}} + 1)}{\sqrt{h}} \left[\sqrt{nM\{1 + \log(\sqrt{h} + 1)\}} + \left(4 + \tfrac{3}{2}\sqrt{2n}\right) M \right]. \tag{14}$$

At fixed model parameters, $\Delta C = O(\sqrt{\log h / h})$. For $n = 1$, excess cost is zero.

We first bound the cost change from replacing one marginal. Fix $\pi \in \Pi^+$ and the other runtimes; within Lemmas 11 and 12, X denotes their sum and x the target runtime. Follow a virtual path on which the target never completes. Each finite prefix is realizable, and the real path agrees through the target's xth execution unit.

Lemma 11 (Common-path decomposition)**.** *Segment a runs from target age a through its next execution tick. Every segment is finite. The following pathwise bounds hold:*

$$0 \le C(a) \le w(4a_{\mathrm{op}} + 1)\big(1 + \textit{other execution in segment } a\big),$$

$$\sum_a \textit{other execution in segment } a \le X, \qquad 0 \le C_\pi(x) - \sum_{a=0}^{x-1} C(a) \le w(4a_{\mathrm{op}} + 1)X.$$

Parallel execution is counted separately for each task.

Proof. Other tasks supply at most X execution units, and Lemma 8 bounds every idle interval. Each segment therefore ends in target execution. Charging preceding ticks to its execution units gives $C(a) \le w(2a_{\mathrm{op}} + 1)(1 + \text{other execution in segment } a)$. The segments partition all execution through target completion. Afterwards, remaining volume is at most X; if positive, the continuation takes at most $(2a_{\mathrm{op}}+1)X + 2a_{\mathrm{op}} \le (4a_{\mathrm{op}}+1)X$ ticks. Multiplying by w bounds its cost; otherwise the remainder is zero. □

Lemma 12 (Replacing one runtime distribution)**.** *Let f, q be positive-integer runtime distributions with finite means and CDFs F, Q. For fixed other runtimes and $\pi \in \Pi^+$, the following bound holds:*

$$|\mathbb{E}_f C_\pi - \mathbb{E}_q C_\pi| \le w(4a_{\mathrm{op}} + 1) \left[\sum_{a \ge 0} |F(a) - Q(a)| + \frac{3}{2} X \sum_x |f(x) - q(x)| \right].$$

Proof. Summation by parts separates cost before and after target completion:

$$\mathbb{E}_f C_\pi = \sum_a C(a)[1 - F(a)] + \sum_x f(x) \left[C_\pi(x) - \sum_{a<x} C(a) \right].$$

Lemma 11 ensures integrability and gives the following bound on the difference of the first terms:

$$w(4a_{\mathrm{op}} + 1) \left[\sum_a |F(a) - Q(a)| + X \sup_a |F(a) - Q(a)| \right].$$

The remainder contributes at most $w(4a_{\mathrm{op}} + 1)X \sum_x |f(x) - q(x)|$. Using $\sup_a |F(a) - Q(a)| \le \frac{1}{2} \sum_x |f(x) - q(x)|$ gives the factor $3/2$. □

Lemma 13 (Empirical distribution error)**.** *Let q_i be the empirical distribution of h samples from f_i, with CDFs Q, F, and put $M_i = \mathbb{E}_{f_i} x_i^2$. For every positive integer $\bar{a}$, the following error bounds hold:*

$$\mathbb{E} \sum_{a \ge 0} |Q(a) - F(a)| \le \sqrt{\frac{M_i\{1 + \log(\bar{a} + 1)\}}{h}} + \frac{2M_i}{\bar{a}},$$

$$\mathbb{E} \sum_x |q_i(x) - f_i(x)| \le \sqrt{\frac{2M_i}{h}}.$$

Proof. Empirical variance gives $\mathbb{E}|Q(a) - F(a)| \le \sqrt{[1 - F(a)]/h}$, while the triangle inequality gives $2[1 - F(a)]$. Use the first bound up to $\bar{a}$ and the second above it. Weighted Cauchy–Schwarz yields the bound:

$$\sum_{a \le \bar{a}} \sqrt{1 - F(a)} \le \left[\sum_{a \ge 0} (a + 1)[1 - F(a)] \sum_{a \le \bar{a}} (a + 1)^{-1} \right]^{1/2} \le \sqrt{M_i\{1 + \log(\bar{a} + 1)\}},$$

since the weighted survival sum is $\mathbb{E}[x_i(x_i+1)/2] \le M_i$. The tail is at most $2\mathbb{E} \max\{x_i - \bar{a} - 1, 0\} \le 2M_i/(\bar{a} + 1)$. Similarly, the following bound holds:

$$\mathbb{E} \sum_x |q_i(x) - f_i(x)| \le \frac{1}{\sqrt{h}} \sum_x \sqrt{f_i(x)} \le \sqrt{\frac{M_i}{h}} \left(\sum_{x \ge 1} x^{-2} \right)^{1/2} \le \sqrt{\frac{2M_i}{h}}.$$

□

Proof of Theorem 10. Replace f_i by q_i in task order. At step i, earlier coordinates use q_j and later coordinates use f_j. Averaging Lemma 12 over these other runtimes gives the following bound uniformly over $\pi \in \Pi^+$:

$$|\mathbb{E}_P C_\pi - \mathbb{E}_Q C_\pi| \le w(4a_{\mathrm{op}} + 1) \sum_i \left[\sum_a |Q_i(a) - F_i(a)| \right.$$
$$\left. + \frac{3}{2} \sum_x |q_i(x) - f_i(x)| \left(\sum_{j<i} \mathbb{E}_{q_j} x_j + \sum_{j>i} \mathbb{E}_{f_j} x_j \right) \right].$$

Independence makes the empirical error for task i independent of other empirical means, whose expectations equal their true means. Apply Lemma 13 and use $\sum_i M_i \le M$, $\sum_i \sqrt{M_i} \le \sqrt{nM}$, and $\mathbb{E}X \le \sqrt{M}$ to obtain the bound:

$$\mathbb{E} \sup_{\pi \in \Pi^+} |\mathbb{E}_P C_\pi - \mathbb{E}_Q C_\pi| \le w(4a_{\mathrm{op}} + 1) \left[\sqrt{\frac{nM\{1 + \log(\bar{a} + 1)\}}{h}} + \frac{2M}{\bar{a}} + \frac{3\sqrt{2n}M}{2\sqrt{h}} \right].$$

The fitted policy and an optimum under P both lie in Π^+ by Proposition 9. Apply Proposition 3 to the preceding uniform bound and take $\bar{a} = \max\{1, \lfloor \sqrt{h} \rfloor\}$. Since $\sqrt{h}/2 \le \bar{a} \le \sqrt{h}$, this gives equation 14. For $n = 1$, the final-task rule gives zero excess cost. □

B.5 INDEPENDENT LOWER BOUND

Proposition 14 (Independent lower bound)**.** *There is a family of independent two-task instances with* $\mathbb{E}_P X^2 \le 49$ *such that, for every* $h \ge 1$*, every learner has, on some family member,* $\Delta C \ge \frac{1-1/\sqrt{6}}{4\sqrt{h}}$.

Proof. Use two single-task workflows A, B, each requiring both processors. Startup is instantaneous; saving and resuming take one tick. Set $x_B = 2$ and let x_A equal 1 with probability p and 5 otherwise. These independent runtimes satisfy $\mathbb{E}X^2 \le 49$.

Starting B yields a minimum expected cost $9 - 4p$. Starting A gives cost 4 if it completes after one tick and 12 otherwise; saving it first would cost 13. Thus starting A yields a minimum expected cost $12 - 8p$. The actions tie at $p = 3/4$, and the wrong choice costs $4|p - 3/4|$. Waiting first adds at least 2.

Set $p_\pm = 3/4 \pm 1/(8\sqrt{h})$. For each fixed history, let q be the probability of starting B and $\psi_\pm$ the conditional excess costs. Then $\psi_+ \ge q/(2\sqrt{h})$ and $\psi_- \ge (1-q)/(2\sqrt{h})$. Only the A histories distinguish the distributions. KL additivity and Pinsker's inequality bound the distance between the full history distributions:

$$h\,\mathrm{KL}(p_+\|p_-) \le \frac{h\{1/(4\sqrt{h})\}^2}{p_-(1-p_-)} \le \frac{1}{3}, \qquad \mathrm{TV} \le \sqrt{\frac{h\,\mathrm{KL}(p_+\|p_-)}{2}} \le \frac{1}{\sqrt{6}}.$$

Integrating $\psi_+ + \psi_- \ge 1/(2\sqrt{h})$ against their common part gives the lower bound:

$$\mathbb{E}_+\psi_+ + \mathbb{E}_-\psi_- \ge \frac{1-\mathrm{TV}}{2\sqrt{h}} \ge \frac{1-1/\sqrt{6}}{2\sqrt{h}}.$$

At least one distribution therefore has excess cost at least $(1-1/\sqrt{6})/(4\sqrt{h})$. □

For target excess cost $\varepsilon \in (0,1]$, $O(\varepsilon^{-2}\log(e/\varepsilon))$ histories suffice under independence. Under its stated moment bound, the hard family requires at least $((1-1/\sqrt{6})/(4\varepsilon))^2$ histories.

B.6 FIXED-GRAPH LOWER BOUND DETAILS

Proof of Theorem 5. Let $x = \lceil h^{1/n} \rceil + 1$. A chain of $n-2$ one-processor tasks forms one workflow and precedes two successors, each forming its own workflow and requiring both processors. Startup takes zero ticks; saving and resuming take one tick each. Give each of the $(x-1)^{n-2}$ chain contexts in $\{x, \dots, 2x-2\}^{n-2}$ probability $p = 1/[2(x-1)^n]$; otherwise all runtimes are one. For each rare context, choose $(x, 2x)$ or $(2x, x)$ as the successor lengths independently and equally likely, then fix these choices across executions. The chain is the only available work before the successors, so the comparator runs it without delay and then takes the shorter successor. Ordinary runs have $X = n$; rare runs have $X \le (2n+2)(x-1)$ and total probability $1/[2(x-1)^2]$. Hence $\mathbb{E}_P X^2 \le n^2 + 2(n+1)^2$ for every fixed assignment of successor lengths.

Fix the learner's randomization. At a context unseen in its histories, the two successor assignments remain equally likely. Neither successor can finish before receiving x execution ticks, so the first to receive x ticks is the same under either assignment. If it is long, the two completion times sum to at least $5x$, against $4x$ for the comparator. The other assignment costs at least $4x$, so their average excess is at least $x/2$, even with probing or preemption. Each context is unseen with probability at least $1 - hp \ge 1/2$. Averaging over the assignments gives the following lower bound:

$$(x-1)^{n-2} p\,\frac{x}{4} = \frac{x}{8(x-1)^2} \ge \frac{1}{16h^{1/n}},$$

where the last step uses $x - 1 \le 2h^{1/n}$. Some fixed assignment attains this bound. □

Graphs with a runtime-independent optimal policy need no histories.

B.7 FEEDBACK EVENTS AND EMPIRICAL BOUNDS

Proposition 3 reduces excess cost to differences in $C_\pi - X$. Before the final task, each tick contributes the number of unfinished workflows minus the number of executing tasks, with absolute value at most $w + n$. The final task's execution cancels, leaving only its operations. We index the earlier contributions by completed runtimes and unfinished-task ages. A virtual path fixes completed runtimes and lets other tasks report noncompletion; each segment ends at the next execution tick.

Lemma 15 (Representation of policy cost)**.** *Fix $\pi \in \Pi^+$ and write $C = C_\pi$. For each completed set I with $|I| \le n-2$, completed record $\boldsymbol{x}'$, and nonnegative integer total unfinished age a, there is at most one virtual segment. Its initial ages sum to a, and its increment in $C - X$ has absolute value at most $(2a_{\mathrm{op}}+1)(w+n)$. For each final task i and completed record of all other tasks, there is at most one remaining-operation term, lying in $[0, (2a_{\mathrm{op}}+1)(w+n)]$. With coefficients and thresholds fixed in each summand, $C - X$ has the following representation:*

$$C(\boldsymbol{x}) - X(\boldsymbol{x}) = \sum_{I:\,|I|\le n-2} \sum_{\boldsymbol{x}',a} \textit{segment increment} \cdot \mathbb{1}_{\{\boldsymbol{x}_I = \boldsymbol{x}',\ \boldsymbol{x}_{\bar{I}} > \boldsymbol{a}\}} + \sum_i \sum_{\boldsymbol{x}'} \textit{remaining operation cost} \cdot \mathbb{1}_{\{\boldsymbol{x}_I = \boldsymbol{x}',\ x_i > a_i\}}.$$

Here $\bar{I}$ is the unfinished set; in the second sum, I contains all tasks except i. Missing segments contribute zero, and finitely many terms are nonzero on each runtime vector.

Proof. Fix runtimes $\boldsymbol{x}'$ for I. The resulting virtual path has realizable finite prefixes. While the completed set equals I, every execution tick increases total unfinished age. Thus each total age a identifies at most one segment. Lemma 8 bounds its length by $2a_{\mathrm{op}} + 1$, giving the stated increment bound.

Let $\boldsymbol{a}$ be its initial unfinished ages. A real path reaches this segment exactly when $\boldsymbol{x}_I = \boldsymbol{x}'$ and $\boldsymbol{x}_{\bar{I}} > \boldsymbol{a}$: before a first divergence, both paths would receive identical feedback. The final tick's feedback determines which segment follows.

Once task i alone remains, only required operations contribute to $C - X$; they take at most a_{op} ticks. The same reachability argument gives the final threshold event. Completed sets only grow, so the segments and final-operation term partition $C - X$. □

Lemma 16 (Uniform threshold error)**.** *Fix an event and let n denote the number of task coordinates in this lemma. Intersect the event with $\{x_i > a_i,\ 1 \le i \le n\}$ over nonnegative integer thresholds. For this class or any subclass, let Q be the empirical distribution of h independent samples. With empirical error defined as event frequency minus event probability, the following bound holds with the displayed c_n:*

$$\mathbb{E} \sup |\textit{empirical error}| \le c_n \sqrt{\frac{\Pr(\textit{fixed event})}{h}},$$

$$c_n = 6 \sum_{m \ge 1} 2^{-m} \sqrt{2\log 2 + 4n\log(n+1) + 4nm\log 4}.$$

Coordinates may be dependent and distributions may have atoms.

Proof. On the sample, use the square root of the indicator-disagreement fraction as the distance. In one coordinate, nested thresholds with consecutive disagreement fractions greater than $4^{-m}/n$ number at most $1 + n4^m$, because those differences are disjoint. A maximal such family covers all thresholds within a disagreement fraction $4^{-m}/n$. Combining the threshold covers for each coordinate gives a distance-2^{-m} cover with at most $(1 + n4^m)^n$ representatives.

Assign independent Rademacher signs to the samples and telescope signed event averages through these covers, starting from the zero indicator. Consecutive representatives have distance at most $3 \cdot 2^{-m}$, so they differ on at most $9h4^{-m}$ points. Using $\cosh z \le e^{z^2/2}$ gives the exponential-moment bound:

$$\mathbb{E} \exp\big(\lambda \,\text{signed-average difference}\big) \le \exp\left(\frac{9\lambda^2 4^{-m}}{2h}\right).$$

Each difference pairs one representative from each of two consecutive covers, so their sizes multiply to bound the number of differences. Since $\log(1 + n4^m) \le \log(n+1) + m\log 4$, optimizing the exponential-moment bound yields the bound:

$$\mathbb{E}\max|\text{signed-average difference at level } m| \le \frac{3\cdot 2^{-m}}{\sqrt{h}}\sqrt{2\log 2 + 4n\log(n+1) + 4nm\log 4}.$$

The chain becomes exact because only finitely many indicators occur on the sample. Summing over levels gives $c_n/(2\sqrt{h})$, and symmetrization doubles this bound. For the fixed event, apply the signed-average bound to its $hQ(\text{fixed event})$ sample points while retaining normalization by h. Then the following empirical-error bound holds:

$$\mathbb{E}\sup|\text{empirical error}| \le \frac{c_n}{h}\mathbb{E}\sqrt{hQ(\text{fixed event})} \le c_n\sqrt{\frac{\Pr(\text{fixed event})}{h}}.$$

This covering argument follows the empirical-process approach of Haussler (1995) and permits dependent coordinates. □

Write $\mathbb{E}_{\mathrm{tr}}$ and $\Pr_{\mathrm{tr}}$ for the distribution truncated at a positive integer ℓ. A second-moment weight controls the sum over completed records; its inverse-weight count determines the power of ℓ.

Lemma 17 (Weighted empirical error)**.** *Suppose $n \ge 3$, the truncated distribution is supported on $\{1,\dots,\ell\}^n$, and $\mathbb{E}_{\mathrm{tr}}X^2 \le M$. For its empirical distribution Q from h samples, the following bound holds with the displayed constant c:*

$$\mathbb{E}\sup_{\pi\in\Pi^+}|(\mathbb{E}_Q - \mathbb{E}_{\mathrm{tr}})(C_\pi - X)| \le c\sqrt{\frac{M\ell^{n-2}}{h}},$$

$$c = (2a_{\mathrm{op}}+1)(w+n)\left[\sqrt{n+1}\sum_{n'=2}^{n}\binom{n}{n'}c_{n'} + \sqrt{2}nc_1\right].$$

Proof. Lemma 15 and the triangle inequality give the following uniform bound over policies:

$$(2a_{\mathrm{op}}+1)(w+n)\sum_{I:\,|I|\le n-2}\sum_{\boldsymbol{x}',a}\max_{\boldsymbol{a}:\,\sum_{i\notin I}a_i=a}|(\mathbb{E}_Q-\mathbb{E}_{\mathrm{tr}})\mathbb{1}_{\{\boldsymbol{x}_I=\boldsymbol{x}',\boldsymbol{x}_{\bar{I}}>\boldsymbol{a}\}}|$$
$$+(2a_{\mathrm{op}}+1)(w+n)\sum_i\sum_{\boldsymbol{x}'}\max_{a_i\le\ell}|(\mathbb{E}_Q-\mathbb{E}_{\mathrm{tr}})\mathbb{1}_{\{\boldsymbol{x}_I=\boldsymbol{x}',x_i>a_i\}}|.$$

In the second sum, I contains all tasks except i. Completed runtimes range from one to ℓ, and total age ranges from zero to $n\ell$; nonempty events have thresholds at most $\ell-1$.

First fix I with $|I| \le n-2$. For each record $\boldsymbol{x}'$ and total age a, the threshold events lie inside $\{\boldsymbol{x}_I = \boldsymbol{x}', \sum_{i\notin I}x_i > a\}$. Lemma 16 therefore bounds their expected maximum error by $c_{n-|I|}\sqrt{\Pr_{\mathrm{tr}}(\boldsymbol{x}_I=\boldsymbol{x}',\sum_{i\notin I}x_i>a)/h}$.

Apply Cauchy–Schwarz with weight $a+1+\sum_{i\in I}x'_i$ to separate a second-moment term from an inverse-weight count. On the corresponding event, this weight is at most X, and at most $\sum_{i\notin I}x_i$ ages contribute. Hence

$$\sum_{\boldsymbol{x}',a}\left(a+1+\sum_{i\in I}x'_i\right)\Pr_{\mathrm{tr}}\left(\boldsymbol{x}_I=\boldsymbol{x}',\sum_{i\notin I}x_i>a\right)$$
$$=\mathbb{E}_{\mathrm{tr}}\sum_{a=0}^{\sum_{i\notin I}x_i-1}\left(a+1+\sum_{i\in I}x_i\right)\le\mathbb{E}_{\mathrm{tr}}\left[X\sum_{i\notin I}x_i\right]\le M.$$

The inverse-weight factor counts completed records and ages. If $I \ne \varnothing$, keep one completed coordinate x in the denominator; the others contribute $\ell^{|I|-1}$ choices. Group pairs (x,a) by their

common denominator $x + a + 1$. The number of pairs in a group is at most its denominator, so each group contributes at most one. There are $(n+1)\ell$ possible denominators, so

$$\sum_{\boldsymbol{x}',a} \frac{1}{a+1+\sum_{i\in I} x'_i} \le \ell^{|I|-1} \sum_{x=1}^{\ell} \sum_{a=0}^{n\ell} \frac{1}{x+a+1}$$
$$\le (n+1)\ell^{|I|} \le (n+1)\ell^{n-2}.$$

For $I = \varnothing$, the factor is at most $n\ell + 1 \le (n+1)\ell^{n-2}$. Combining the two factors bounds the contribution of each completed set by $(2a_{\mathrm{op}}+1)(w+n)c_{n-|I|}\sqrt{(n+1)M\ell^{n-2}/h}$.

The final-operation term has $n-1$ completed runtimes. Weight one completed runtime by its square; the convergent sum of x^{-2} removes one coordinate and retains ℓ^{n-2}. For fixed final task i, Lemma 16 gives error $c_1\sqrt{\Pr_{\mathrm{tr}}(\boldsymbol{x}_I = \boldsymbol{x}')/h}$. Choose $j \in I$ to obtain

$$\left(\sum_{\boldsymbol{x}'} \sqrt{\Pr_{\mathrm{tr}}(\boldsymbol{x}_I = \boldsymbol{x}')}\right)^2 \le \left[\sum_{\boldsymbol{x}'} (x'_j)^2 \Pr_{\mathrm{tr}}(\boldsymbol{x}_I = \boldsymbol{x}')\right] \left[\sum_{\boldsymbol{x}'} (x'_j)^{-2}\right]$$
$$\le M\ell^{n-2} \sum_{x=1}^{\ell} x^{-2} \le 2M\ell^{n-2}.$$

Summing over n final tasks and $\binom{n}{n'}$ completed sets with n' unfinished tasks gives the stated c. □

The constant c in Lemma 17 may grow exponentially in n.

The two-task case. Before the final task, $I = \varnothing$ and the inverse-weight sum is $\sum_{a=0}^{2\ell}(a+1)^{-1} \le 1 + \log(2\ell+1)$. The final-operation calculation above still applies, giving the following empirical-error bound:

$$(2a_{\mathrm{op}}+1)(w+n)\left[c_2\sqrt{1+\log(2\ell+1)} + 2\sqrt{2}c_1\right]\sqrt{M/h}.$$

Take $\ell = \max\{1, \lfloor\sqrt{h}\rfloor\}$. Since $\sqrt{1+\log(2\ell+1)} \ge 1$, this is at most $c\sqrt{M\{1+\log(2\ell+1)\}/h}$ with $c = (2a_{\mathrm{op}}+1)(w+n)(c_2 + 2\sqrt{2}c_1)$. The same truncation comparison gives equation 8.

B.8 Structured fitting

Theorem 18 (Structured upper bound)**.** *For an r-structured distribution with $\mathbb{E}_P X^2 \le M$, the policy learned by structured fitting satisfies the following bound for every $h \ge 1$:*

$$\Delta C \le \{w(4a_{\mathrm{op}}+1)+1\}\left[4(n+3)M + 16n^2M + 4n\sqrt{nM} + 2n^2\right] h^{-1/(r+4)}. \tag{15}$$

Proof. Coordinatewise truncation can change the supplied factorization. Truncate P at ℓ, then multiply the truncated conditionals along the statistical graph to obtain the recombined distribution. Write $\mathbb{E}_{\mathrm{trunc}}$ and $\mathbb{E}_{\mathrm{rec}}$ for expectations under these distributions; zero-probability contexts use a point mass at one. Comparing P with Q through the recombined distribution separates truncation and recombination from estimation:

$$\Delta C \le 2 \sup_{\pi\in\Pi^+} |(\mathbb{E}_P - \mathbb{E}_{\mathrm{rec}})(C_\pi - X)| + 2\mathbb{E} \sup_{\pi\in\Pi^+} |(\mathbb{E}_{\mathrm{rec}} - \mathbb{E}_Q)(C_\pi - X)|.$$

First bound the difference between P and the recombined distribution. As in Theorem 4, truncation contributes at most $2\{w(4a_{\mathrm{op}}+1)+1\}M/\ell$. On $\{\max_j x_j < \ell\}$, the original, truncated, and recombined point probabilities agree because parent contexts and conditionals agree. Thus only the tail contributes:

$$\Pr_{\mathrm{rec}}(\max_j x_j \ge \ell) = \Pr_{\mathrm{trunc}}(\max_j x_j \ge \ell) \le M/\ell^2,$$
$$|(\mathbb{E}_{\mathrm{trunc}} - \mathbb{E}_{\mathrm{rec}})(C_\pi - X)| \le \{w(4a_{\mathrm{op}}+1)+1\}\left[\mathbb{E}_P(X\mathbb{1}_{\{X\ge\ell\}}) + \frac{nM}{\ell}\right]$$
$$\le \{w(4a_{\mathrm{op}}+1)+1\}(n+1)M/\ell.$$

Adding truncation bounds the first supremum by $\{w(4a_{\mathrm{op}}+1)+1\}(n+3)M/\ell$.

For estimation, replace conditionals by q_i in reverse topological order, leaving parent contexts under the recombined distribution. Couple runtimes using independent uniforms in topological order and fix all but task i's uniform. Its context and non-descendants remain fixed, while descendants are functions of x_i. $|C_\pi - X| \le \{w(4a_{\mathrm{op}}+1)+1\}n\ell$ then bounds the replacement error by

$$\{w(4a_{\mathrm{op}}+1)+1\}n\ell \sum_{\text{contexts}} \Pr_{\mathrm{rec}}(\text{context}) \sum_x |\Pr_{\mathrm{trunc}}(x \mid \text{context}) - q_i(x \mid \text{context})|.$$

The context weights above are recombined marginals, whereas estimation uses truncated histories. The two joint distributions agree outside tails of mass at most M/ℓ^2 each, so the summed absolute differences between context probabilities are at most $2M/\ell^2$. Since the inner mass error is at most two, changing to truncated context weights adds at most $4\{w(4a_{\mathrm{op}}+1)+1\}nM/\ell$ per task.

For a context of truncated probability $p > 0$, the number of matching histories is $N \sim \mathrm{Bin}(h,p)$. When this count is positive, Lemma 13 bounds the expected mass error by $\sqrt{2\mathbb{E}_{\mathrm{trunc}}[x_i^2 \mid \text{context}]/N}$. For $N = 0$, use the bound two. Averaging over N uses $N^{-1/2} \le \sqrt{2/(N+1)}$, $\mathbb{E}[1/(N+1)] \le 1/((h+1)p)$, and $\Pr(N=0) \le e^{-hp}$. Summing over contexts gives

$$\begin{aligned}
&\sum_{\text{contexts}} p\,\mathbb{E} \sum_x |\Pr_{\mathrm{trunc}}(x \mid \text{context}) - q_i(x \mid \text{context})| \\
&\quad\le \frac{2}{\sqrt{h+1}} \sum_{\text{contexts}} \sqrt{p\,\mathbb{E}_{\mathrm{trunc}}[x_i^2 \mid \text{context}]} + 2\sum_{\text{contexts}} pe^{-hp} \\
&\quad\le 2\sqrt{\frac{\ell^r M_i}{h+1}} + \frac{2\ell^r}{eh},
\end{aligned}$$

where $M_i = \mathbb{E}_P x_i^2$. The last step uses at most ℓ^r contexts, Cauchy–Schwarz, and $pe^{-hp} \le 1/(eh)$.

Summing the replacement bounds over tasks, using $\sum_i \sqrt{M_i} \le \sqrt{nM}$, and including the context-weight corrections yields the bound:

$$\begin{aligned}
&\mathbb{E} \sup_{\pi\in\Pi^+} |(\mathbb{E}_{\mathrm{rec}} - \mathbb{E}_Q)(C_\pi - X)| \\
&\quad\le \{w(4a_{\mathrm{op}}+1)+1\}n\ell \left[2\sqrt{\frac{\ell^r nM}{h+1}} + \frac{2n\ell^r}{eh}\right] + \frac{4\{w(4a_{\mathrm{op}}+1)+1\}n^2 M}{\ell}.
\end{aligned}$$

Set $\ell = \max\{1, \lfloor h^{1/(r+4)} \rfloor\}$. Then $1/\ell \le 2h^{-1/(r+4)}$, $\ell\sqrt{\ell^r/h} \le h^{-1/(r+4)}$, and $\ell^{r+1}/h \le h^{-3/(r+4)} \le h^{-1/(r+4)}$. Substituting both error bounds into the initial comparison and using $4/e \le 2$ gives equation 15. □

Theorem 19 (Structured lower bound)**.** *For every $r \ge 1$, the family in Theorem 5 with $n = r+2$ is r-structured. Hence every learner has excess cost at least $1/(16h^{1/(r+2)})$ on some member of this family, with the second-moment bound in equation 10 unchanged.*

Proof. In the family of Theorem 5, the first chain runtime has no parent. Each later runtime is one after an ordinary predecessor and uniform on the rare range after a rare predecessor; both successors depend on the r chain runtimes. Thus the distribution is r-structured; the cost and moment bounds are unchanged. □

B.9 Censored runtime logs

Theorem 20 (Censored logs)**.** *Suppose task runtimes are independent and bounded by known positive integers $x_i^{\max}$. For task i, its $h_i \ge 1$ logs are i.i.d. with runtime distribution f_i and are independent of the new execution. Censoring age is independent of runtime, and logs of different tasks are independent. Write F_i, Q_i for the true and fitted task CDFs. The policy learned by Kaplan–Meier fitting satisfies the following bound:*

$$\Delta C \le 2w(4a_{\mathrm{op}}+1)\left(1 + 3\sum_j x_j^{\max}\right) \sum_i \mathbb{E} \sum_{a=0}^{x_i^{\max}-1} |Q_i(a) - F_i(a)|. \tag{16}$$

If $\Pr(\textit{censoring age} > x_i^{\max}) > 0$*, the expected sum of CDF errors for task* i *is bounded by* $\frac{x_i^{\max} c_0}{\Pr(\textit{censoring age} > x_i^{\max})\sqrt{h_i}}$*, where* c_0 *is an absolute constant.*

Proof. Fix task i, with masses f_i, q_i and CDFs F, Q on $\{1, \dots, x_i^{\max}\}$. The identity $q_i(x) - f_i(x) = \{Q(x) - F(x)\} - \{Q(x-1) - F(x-1)\}$ and matching endpoint values give $\sum_x |q_i(x) - f_i(x)| \le 2G$, where $G = \sum_{a < x_i^{\max}} |Q(a) - F(a)|$. Since other runtimes sum to at most $\sum_j x_j^{\max}$, Lemma 12 gives the following bound for one replacement:

$$w(4a_{\mathrm{op}} + 1) \left[G + \frac{3}{2} \left(\sum_j x_j^{\max} \right) 2G \right] = w(4a_{\mathrm{op}} + 1) \left(1 + 3 \sum_j x_j^{\max} \right) G.$$

Replace marginals successively and apply Section 4.2 twice to obtain equation 16.

Let Q' be the Kaplan–Meier estimator before endpoint mass is moved, so $Q' = Q$ below $x_i^{\max}$. Theorem 2 of Bitouzé et al. (1999) yields the following tail bound for some absolute C:

$$\Pr\left(\sqrt{h_i} \sup_a |(Q'(a) - F(a)) \Pr(\text{censoring age} \ge a)| > \lambda \right) \le 2.5 e^{-2\lambda^2 + C\lambda}.$$

The theorem permits discrete distributions and independent i.i.d. runtime and censoring sequences. Completion takes priority on ties, giving the left-limit survival weight; uncensored bounded runs may use censoring age $x_i^{\max} + 1$.

Integrating the tail bound gives the following estimate:

$$\mathbb{E} \sup_a |(Q'(a) - F(a)) \Pr(\text{censoring age} \ge a)| \le \frac{c_0}{\sqrt{h_i}}, \quad c_0 = \int_0^\infty \min\{1, 2.5 e^{-2\lambda^2 + C\lambda}\} \, d\lambda < \infty.$$

Wellner (2007) bounds C. The survival weight is at least $\Pr(\text{censoring age} > x_i^{\max})$, giving the CDF-error bound in the theorem. Substitution into equation 16 completes the proof. □