Uncertainty Discourse and Applications

www.uda.reapress.com

Uncert. Disc. Appl. Vol. 3, No. 4 (2026) 272–284.

Paper Type: Original Article

Uncertainty Aware Heuristic Scheduling for Resource Constrained Distributed Data Processing

Min-Chul Park[1,*], Seo-Yeon Kim[1], Alessandro Conti[2], Samar Q. Hassan[3], Pieter J. De Wit[4]

[1] Department of Mechanical Design Engineering, Korea Advanced Institute of Engineering Design (KAIED), Daejeon, Republic of Korea; cpark216@yahoo.com.

[2] Dipartimento di Meccanica, Politecnico di Milano, Milano, Italy; alessandro.conti@uniroma1.it.

[3] Department of Industrial Engineering, Jordan University of Science and Technology, Irbid, Jordan; samar.kh@hotmail.com.

[4] Department of Mechanical Engineering, KU Leuven, Leuven, Belgium.

Citation:

Received: 11 February 2026 *Revised: 17 May 2026* *Accepted: 23 July 2026*	Park, M., Kim, S., Conti, A., Hassan, S. Q., & De Wit, P. J. (2026). Uncertainty aware heuristic scheduling for resource constrained distributed data processing. *Uncertainty Discourse and Applications, 3*(4), 272-284.

Abstract

Distributed task scheduling under variable durations requires distinguishing local risk-aware placement from a workflow-level deadline guarantee. We formulate a chance-constrained scheduling framework and examine reported results for CHARM, a restricted quantile-guided heuristic on 100 random layered Directed Acyclic Graphs (DAGs) of 30–60 tasks and an eight-worker heterogeneous cluster. Worker calibration traces inform per-task placement and a bounded local repair pass. At log-normal noise $\sigma = 0.3$, mean realized makespan is 9.80 time units versus 9.94 for a Heterogeneous Earliest Finish Time (HEFT) assignment evaluated under common dispatch semantics, a 1.4% reduction, with nominal makespan 8.07 versus 7.82. The fraction exceeding 1.1 times each method's own nominal makespan is 0.797 versus 0.891; this method-dependent threshold is a plan-relative measure rather than a common external deadline. Removing the local chance gate raises realized makespan by 0.13 and the plan-relative exceedance fraction by 0.029. Removing local repair leaves the displayed realized mean unchanged. The experiment does not certify the workflow chance constraint, evaluate runtime rescheduling, or test production traces. Reported statistical results and duration conventions require raw-outcome verification, including an inconsistency in the miss-probability signed-rank summary.

Keywords: Task scheduling, Uncertainty, Chance constraints, Heterogeneous computing, Makespan robustness, Heterogeneous earliest finish time.

1 | Introduction

Every evening a national weather service must complete an ensemble data-assimilation and forecasting pipeline before the bulletin deadline, and every night a bioinformatics centre regenerates variant-calling pipelines whose results steer the next day's experiments. These workloads are expressed as Directed Acyclic Graphs (DAGs) of tasks with data dependencies and are executed on shared clusters whose effective capacity

Corresponding Author: cpark216@yahoo.com

https://doi.org/10.48313/uda.v3i4.115

Licensee System Analytics. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0).

is far smaller than the sum of their parts. Production experience with systems such as Pegasus shows how variable scientific workloads are in practice [1], and characterisation studies report task durations spanning orders of magnitude, with stragglers common rather than exceptional [2]. When a deadline-bearing workflow lands on a contended cluster, the scheduler's treatment of that variability largely decides whether the deadline is met.

The Heterogeneous Earliest Finish Time (HEFT) heuristic ranks tasks and assigns them using point processing-time estimates [3]. Production cluster analysis demonstrates that heterogeneity and time variation affect machine performance [4]. Workflow managers and scheduling models supply the wider setting [1], [2]. The present simulation abstracts those systems into a fixed heterogeneous cluster and omits communication and contention.

Precedence-constrained scheduling is computationally difficult even for deterministic processing times [5], and stochastic models introduce decisions about distributions as well as assignment [6]. Chance constraints state a probability requirement, while robust optimization protects against a specified uncertainty set. The present study examines a tractable risk-aware heuristic; it does not establish that existing scheduling research lacks explicit uncertainty treatment.

The contribution is a framework that separates distribution estimation, schedule construction, and risk assessment, together with reported evidence for a simpler worker-quantile implementation. This separation makes clear which design features are proposed and which are evaluated.

The formal model expresses workflow deadline risk as a chance constraint. A local 90% task-placement gate is examined experimentally but is not equivalent to that joint workflow requirement.

Worker calibration traces inform the implemented placement rule. Per-class Bayesian shrinkage and transfer-distribution estimation are general extensions, not tested components.

A bounded local repair pass is evaluated during schedule construction. Runtime monitoring and posterior-based rescheduling remain conceptual extensions.

The analysis states the conditions required for probability certification and the cost of estimating risk; no unconditional reliability certificate is claimed.

Reported experiments compare CHARM, random assignment, EFT list scheduling, HEFT-based assignment, and static greedy on 100 synthetic DAGs under three noise levels.

The rest of the paper is arranged in the following order. Section 2 reviews deterministic and stochastic scheduling research together with the workflow systems that motivate the problem, and Section 3 introduces the model, the research questions, and the chance-constrained formulation. Section 4 presents the CHARM framework and *Algorithm 1*. Section 5 reports the executed experimental campaign and ends with an epistemic status note. Section 6 examines limitations and alternative positions, and Section 7 concludes with empirical answers and a roadmap.

2 | Related Work

2.1 | Deterministic Directed Acyclic Graph and List Scheduling

List scheduling assigns task priorities and places ready tasks greedily; Kwok and Ahmad [7] survey deterministic graph scheduling. HEFT uses upward ranks based on average durations and insertion-based earliest finish [3]. Stochastic priority-policy analysis addresses related random-duration scheduling problems [6], while textbook treatments establish the broader theory [5]. The problems and guarantees of those sources are not transferred to the heterogeneous DAG heuristic tested here.

The deterministic methods considered here consume point processing-time estimates and produce schedules quickly. CHARM retains a list-scheduling structure and uses empirical worker multipliers to guide placement. Broader stochastic scheduling literature already studies uncertain durations; the proposed comparison

concerns this specific computational construction rather than a universal distinction between deterministic and uncertainty-aware methods.

Insertion, duplication, and lookahead alter how a list scheduler allocates tasks [7]. These design choices can interact with duration models. The reported comparison evaluates static assignment and worker priority under a shared dispatcher, so it does not directly compare all insertion schedules as constructed or establish that uncertainty treatment is independent of dispatch semantics.

2.2 | Scheduling under Uncertainty

Stochastic scheduling analyses priority policies under random processing times [5], [6]. Chance-constrained programming expresses a requirement that a constraint hold with a specified probability [8]. Robust optimization protects against a specified uncertainty set [9], [10]. Concentration inequalities support sampling error bounds under their assumptions [11], and Bayesian modelling offers methods for estimating predictive distributions [12]. These tools provide a formal context for the general framework.

Closer to systems practice, cloud scheduling research has examined deadline-constrained provisioning, including partial critical-path algorithms for infrastructure-as-a-service clouds [13] and auto-scaling approaches that trade cost against deadlines [14]. These works largely treat the deadline as deterministic given provisioned resources, and uncertainty enters through provisioning margins rather than through the schedule itself. Our position is complementary: we take the cluster as given and resource-constrained, and move the uncertainty treatment inside the scheduling computation, where trace-informed distributions can be exploited rather than padded away.

The reported experiment releases one DAG on a fixed cluster. Online arrivals, queue stability across jobs, and runtime adaptation require separate models and are outside its scope. The static local repair pass should not be interpreted as an evaluated online scheduler.

2.3 | Workflow Systems Traces and Simulation

Pegasus and workflow profiling studies supply operational context [1], [2], and production cluster analysis illustrates heterogeneity [4]. The campaign uses synthetic layered DAGs and a custom dispatcher rather than production replay. This allows controlled pairing but leaves operational workload diversity and communication costs untested.

Trace-based evaluation can connect scheduling models to observed task variability. Controlled paired simulation also supports comparisons on identical generated instances. The reported experiment uses common random draws across methods, but only ten realized draws per DAG and noise level, so its per-DAG exceedance estimates are coarse. Production-trace validation and calibration uncertainty remain necessary extensions.

2.4 | Synthesis and Positioning

CHARM combines a list-scheduling backbone with distribution-informed placement and repair. The general framework targets a workflow chance constraint; the implemented heuristic uses empirical worker quantiles and a local placement gate. Its reported advantage must be assessed separately from whether it meets an externally specified deadline risk.

3 | Problem Formulation and Research Questions

3.1 | Research Questions and Hypotheses

The research questions concern uncertainty representation, scheduling benefit, and distribution misspecification. The reported experiment addresses restricted versions of them.

RQ1 asks how predictive duration distributions and their estimation uncertainty can inform scheduling. The evaluated implementation uses worker multiplier quantiles rather than a fitted per-class posterior.

RQ2 asks whether risk-aware construction improves realized performance and workflow deadline reliability at an acceptable cost. A common external deadline is required to evaluate the latter directly.

RQ3 asks how performance responds to distribution error, heavy tails, and correlated contention. The reported three-level noise sweep tests only a limited form of distribution shift.

H1 predicts an improvement from risk-aware construction relative to mean-based HEFT. The experiment measures realized makespan and an own-plan exceedance fraction, rather than joint deadline compliance.

H2 predicts a benefit from runtime repair. The reported ablation concerns a static local repair pass, so it cannot test runtime rescheduling in its stated form.

H3 predicts controlled degradation as the estimated distribution departs from the true distribution. Three noise levels do not establish linearity in an estimation-error metric, heavy-tail behaviour, or robustness to correlated contention.

These hypotheses distinguish design aims from findings. No verified preregistration record was supplied. Sections 3 and 4 describe the general framework; Section 5 identifies the narrower implementation and evidence.

3.2 | System Model and Notation

A job is a DAG G = (T, E) of tasks i in T connected by data edges (i, j) in E. The platform consists of heterogeneous resources r in R, each with a speed normalisation and a resource-time price. The duration of task i on resource r is a random variable D_{i, r} whose distribution is estimated from traces, and the transfer volume along edge (i, j) is likewise random when its endpoints are placed on different resources. The job carries a deadline delta and, optionally, a resource budget B. A schedule s assigns each task to a resource and a start order, and its makespan C_max(s) is the completion time of the final task, itself a random variable induced by the durations. *Table 1* lists the principal symbols.

Table 1. Principal notation for the chance-constrained scheduling problem.

Symbol	Meaning	Unit
G = (T, E)	Workflow DAG of tasks and data edges	-
D_{i,r}	Random duration of task i on resource r	Time units
W_{ij}	Random transfer volume on edge (i, j)	MB
R	Set of heterogeneous resources	-
delta	Job deadline	Time units
epsilon	Tolerated violation probability	-
C_max(s)	Makespan of schedule s	Time units
rho_r	Resource-time price of resource r	Cost/time unit
alpha	Rank quantile level	-
z_{1-alpha}	Standard normal quantile at level 1-alpha	-
K	Independent draw count for conceptual risk assessment	-
L_i	Latest allowable start of task i	Time units

3.3 | Formal Problem Statement

A schedule assigns each task to a resource and a nonpreemptive worker order, respecting precedence and at most one executing task per worker. With resource-time price ϱ_r, the general problem minimizes expected execution cost subject to a fixed deadline δ and risk tolerance ε:

$$\begin{aligned} &\text{minimize_s } E[\Sigma_{i\in T}\, \rho_{s(i)}\, D_{i,s(i)}] \\ &\text{s.t} \\ &\Pr\{C_{max}(s) > \delta\} \le \varepsilon, \end{aligned} \tag{1}$$

precedence and worker nonoverlap constraints.

Eq. (1) is a general chance-constrained model [8]. A simulated exceedance fraction is an estimator of its risk, not automatically a certificate. For a fixed schedule evaluated using K independent draws from a specified model, let $\hat{p}$_viol be the empirical fraction:

$$\hat{p}_viol(s) = K^{-1} \Sigma_{k=1}^{K} 1\{C_max^{(k)}(s) > \delta\};\ \hat{p}_viol(s) + \sqrt{[\log(1/\beta)/(2K)]} \le \varepsilon. \tag{2}$$

The margin in *Eq. (2)* is a one-sided Hoeffding bound at confidence $1-\beta$, where $0 < \beta < 1$, for a fixed schedule under the sampling model [11]. It does not account for misspecified duration laws, estimated model parameters, correlated draws, or repeated adaptive schedule selection. Independent final certification or a simultaneous bound is needed when many candidates are searched. The reported implementation neither uses this workflow certification rule nor demonstrates the deadline guarantee in *Eq. (1)*.

4 | Proposed Methodology

4.1 | Framework Overview

The general framework has estimation, construction, and monitoring layers. Estimated duration and transfer models supply schedule candidates; risk evaluation determines whether they meet a fixed deadline; monitoring can initiate rescheduling with updated information. These are conceptual interfaces. The reported implementation uses empirical worker multiplier traces, a local deadline gate, and a static bounded repair pass.

Offline estimation and event-triggered updates can reduce runtime overhead, but their cost and calibration must be measured. No production monitoring or runtime-event experiment is reported here.

4.2 | Distributional Estimation from Traces

A per-class log-normal model with parameter uncertainty and shrinkage is a possible estimation layer [12]. Such a model would require an estimation procedure, priors, calibration checks, and a predictive distribution before it could support a chance constraint. Empirical quantiles are another option, with uncertainty depending on trace size and relevance. Finite moment diagnostics cannot prove that a distribution has infinite variance. The reported campaign uses 500 multiplier samples per worker at $\sigma_cal = 0.3$, without evaluating per-class shrinkage or transfer estimation.

4.3 | Chance Aware Construction

A list scheduler can prioritize tasks using a duration score that combines a predictive mean μ with variability σ. *Eq. (3)* is a normal-quantile surrogate, not a log-normal quantile or a bound on a path maximum:

$$\tilde{d}_\{i,r\} = \mu_\{i,r\} + z_\{1-\alpha\}\, \sigma_\{i,r\},\ 0 < \alpha < 1. \tag{3}$$

For $D \sim \text{LogNormal}(a,b^2)$, the exact marginal quantile is $\exp(a + b\, z_\{1-\alpha\})$. Summing marginal quantiles does not in general give the same quantile of a sum or workflow makespan. Joint deadline assessment must include precedence, worker queueing, dependence, and communication assumptions. The tested heuristic derives task deadlines from downstream slack and uses empirical worker multiplier quantiles; it does not propagate exact workflow distributions.

If no candidate placement meets a risk requirement, the framework must report infeasibility or use a specified escalation whose revised risk and resource cost are evaluated. Replication or relaxing ε changes the decision problem and cannot silently preserve the original guarantee. A local per-task gate with probability 0.9 does not imply 90% workflow completion probability; a joint certificate or valid allocation of total risk is required.

4.4 | Runtime Repair

Runtime repair would condition remaining durations on observed progress and reschedule only unexecuted tasks while respecting completed and running work. Its effectiveness depends on engine support, repair cost, trigger definitions, and re-evaluation of joint risk. This is a proposed extension. The reported repair pass modifies a static schedule using a quantile-inflated makespan estimate, with at most five moves, and does not demonstrate event-driven runtime adaptation.

4.5 | The CHARM Algorithm

Algorithm 1 states the general construction-and-monitoring interfaces, not the procedure executed in Section 5. Probability checks refer to independent or simultaneous risk assessment under a specified joint model. The narrower worker-quantile heuristic is described in the experimental setup.

Algorithm 1. Conceptual workflow risk assessment and rescheduling.

Input: DAG G, resources R, fixed deadline δ, risk ε, predictive model, certification confidence
1: fit or load specified predictive models and calibration evidence
2: generate a precedence-feasible schedule using a list heuristic
3: assess workflow risk using independent or simultaneous certification
4: while no admissible schedule has been found do
5: generate a new candidate within a stated search budget
6: verify worker nonoverlap and precedence constraints
7: evaluate expected cost and joint deadline risk
8: if the risk condition is met, retain the candidate
9: if the search budget is exhausted, report failure or an explicit escalation
10: end construction loop; if no admissible schedule exists, report failure and stop
11: otherwise dispatch the retained schedule; optionally monitor execution progress
12: while unfinished work remains do
13: update the conditional duration model without changing completed work
14: if a specified repair trigger is met, generate a feasible remaining-work schedule
15: reassess its risk using the stated certification procedure
16: apply it only within the repair and resource budgets
17: report estimated risk, confidence scope, and any escalation
Output: schedule and risk assessment conditional on the supplied model

4.6 | Complexity Assumptions and Failure Conditions

A HEFT-style construction has $O(|T|^2|R|)$ scheduling cost under its standard implementation. Dense task–resource costs require $O(|T||R|)$ storage, in addition to $O(|T|+|E|)$ graph and schedule state. Simulating K workflow realizations adds the cost of K full dispatch runs; it cannot generally be bounded by only the number of nominal critical paths, because resource contention can change the realized path. Candidate search, model estimation, and repeated repairs multiply these costs. The reported experiment supplies total harness timing, without a separate scheduler-cost benchmark.

The simulation assumes the stated task multipliers and fixed worker factors. Correlated contention requires an explicit joint model and is untested.

Calibration traces must remain relevant to future durations. The outer noise levels intentionally change the multiplier distribution.

Task deadlines derived from slack are heuristic allocations; near-tied paths and queueing can invalidate a local interpretation as a joint guarantee.

Repair requires an explicit move and computation budget. The tested pass allows at most five moves.

These conditions define the proposed risk assessment. Monitoring does not by itself establish that every failure regime is detectable or that a cold-start fallback is safe.

Heavy-tailed durations may require empirical or model-based quantiles rather than moment scores; their adequacy must be validated.

Shared contention can couple task durations and defeat an independence-based assessment.

A cold start without defensible duration information leaves reliability unquantified. A conservative schedule cannot be called safe without a justified uncertainty set.

5 | Experimental Results and Analysis

5.1 | Setup

The manuscript reports 100 random layered DAGs with one entry and one exit, 30–60 tasks (mean 45.3), 3–6 middle layers, and inter-layer edge probabilities in [0.2, 0.4]. Eight workers have speed factors in [0.53, 1.91] and fixed noise factors f_w in [0.66, 1.48]. A nominal task duration is base duration divided by speed; realized duration multiplies it by LogNormal(0,σ f_w), with $\sigma \in \{0.2,0.3,0.4\}$. Under this reported parameterization the multiplier mean is $\exp((\sigma f_w)^2/2)$, greater than one, so the nominal duration is not the stochastic mean. The table label "expected" denotes nominal simulation values here. Whether the code instead centers multipliers requires verification.

Every method is reported as receiving nominal durations and a 500-task multiplier calibration trace per worker at σ_cal = 0.3. CHARM sets T* to 1.05 times the HEFT nominal construction makespan, derives task deadlines from downstream slack, and chooses a worker passing an empirical 90% local finish gate, preferring the earliest median-scaled finish. A static repair pass responds to a quantile-inflated makespan estimate above T*. The exact feasibility fallback, task-deadline formula, and repair moves require the implementation record. The 0.9 task gate is an estimated local condition, not a certified workflow chance constraint.

All methods are evaluated without communication cost by a shared event-driven dispatcher. A static assignment and per-worker priority order are converted into execution by starting the highest-priority ready task on each idle worker. This may differ from HEFT's constructed insertion schedule, so the comparison is between assignments under common dispatch rather than faithful insertion-schedule execution. Ten common random draws per DAG and noise level pair methods. The reported miss fraction uses a different threshold for each method, 1.1 times its own nominal makespan; robustness is exactly one minus that fraction and supplies no independent outcome. No common external deadline or T* compliance is reported. Software references identify NumPy, SciPy, and statsmodels [15–17], and the stated tests use Wilcoxon [18], Demšar [19], and Holm [20].

Table 2. Executed experimental setup: workload, cluster, information model, methods, evaluation, and software, as run on 2026-10-11.

Component	Setting (as Executed)
Workloads	100 random layered DAGs, 30-60 tasks (mean 45.3), 3-6 layers, inter-layer edge probability 0.2-0.4, log-normal base durations, one entry and one exit task
Cluster	8 heterogeneous workers, speed factors in [0.53, 1.91], realized duration = expected x LogNormal(0, sigma x f_w) with per-worker noise factor f_w in [0.66, 1.48]
Information model	Schedulers see expected durations only, plus a 500-task calibration trace per worker (empirical multiplier quantiles) taken at sigma_cal = 0.3
Noise levels	Sigma in {0.2, 0.3, 0.4}, 10 common-random-number duration draws per DAG and level
Methods	Random assignment, EFT list scheduling (insertion-based), HEFT (upward rank + insertion-based EFT), static greedy on expected durations, CHARM (deadline from downstream slack, P(finish <= deadline) >= 0.9 via worker quantiles, local repair)
Evaluation	One shared event-driven simulator (static assignment, dynamic ready dispatch per worker in priority order) scores all methods on expected and realized durations
Metrics	Expected makespan, realized makespan (mean +/- std), deadline-miss probability P(realized > 1.1 x expected), robustness index P(realized <= 1.1 x expected)

Table 2. Continued.

Component	Setting (as Executed)
Statistics	Paired Wilcoxon over the 100 DAGs (primary: realized makespan at sigma = 0.3, CHARM vs HEFT); Friedman over five methods with Holm-corrected pairwise Wilcoxon
Software/runtime	Python 3.12, numpy, scipy, statsmodels, matplotlib; CPU only; full campaign 8.4 s

5.2 | Main Comparison RQ1 RQ2

At $\sigma = 0.3$, *Table 3* gives mean realized makespan 9.80 for CHARM and 9.94 for HEFT-based assignment under the shared dispatcher, a reduction of 1.4%. Reported within-DAG standard deviations are 0.94 and 1.02. The source gives paired Wilcoxon W = 1413, p = 0.000132 across 100 DAG means, with CHARM lower on 66 DAGs. The medians are nearly equal, 9.68 and 9.67. Random, EFT, and static greedy have larger realized means. CHARM's nominal makespan is 8.07 versus 7.82 for HEFT, a 3.2% increase. This comparison describes an assignment trade-off under the simulator, rather than an increase in expected monetary or energy cost.

Table 3. Main comparison at sigma = 0.3 over 100 DAGs: expected makespan, realized makespan (mean over 10 draws per DAG), deadline-miss probability, and robustness index. Makespans in cluster time units.

Method	Expected Makespan	Realized Makespan (Mean ± within-DAG std)	Miss Prob. (Mean ± std)	Robustness
Random	19.66 ± 5.54	21.82 (2.64)	0.501 ± 0.165	0.499
EFT list	9.40 ± 1.94	10.95 (1.14)	0.663 ± 0.189	0.337
Greedy (exp.)	9.63 ± 2.07	11.21 (1.19)	0.652 ± 0.213	0.348
HEFT	7.82 ± 1.69	9.94 (1.02)	0.891 ± 0.136	0.109
CHARM	8.07 ± 1.71	9.80 (0.94)	0.797 ± 0.179	0.203

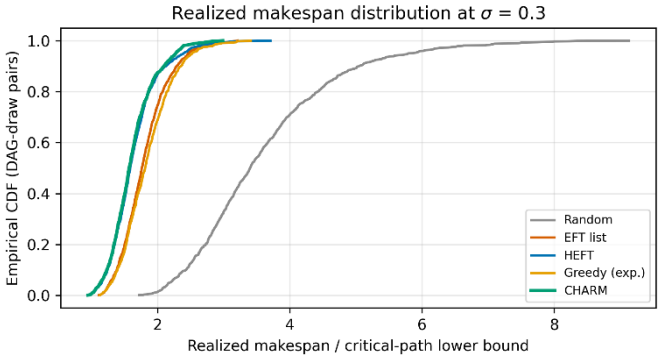


Fig. 1. Reported empirical CDF of realized makespan at σ = 0.3, normalized by the stated critical-path lower bound, pooling 1,000 DAG–draw observations per method. The close CHARM and HEFT curves do not establish stochastic dominance; draws within a DAG are related observations.

Plan-relative exceedance fractions are 0.797 for CHARM and 0.891 for HEFT; their complementary robustness values are 0.203 and 0.109. CHARM's threshold is larger because its nominal makespan is larger. Random assignment has a smaller exceedance fraction despite much slower execution because its own plan is substantially longer. These values measure conformity to each method's own nominal plan, not the probability of meeting one shared application deadline. The source's signed-rank summary for this comparison is internally inconsistent and remains a verification dependency in *Table 7*.

5.3 | Ablation Which Component Carries the Gain

Removing the local chance gate raises displayed realized makespan from 9.80 to 9.93, with reported W = 1348 and p = 5.19 × 10^{-5}; its own-plan exceedance fraction rises by 0.029, from 0.797 to 0.826. The ablated nominal makespan (8.04) differs from HEFT's (7.82), so this variant cannot be described as identical to HEFT. Removing static repair leaves both displayed realized means at 9.80 (reported p = 0.69) and changes exceedance fraction from 0.797 to 0.842 while also lowering the nominal plan from 8.07 to 7.93. This change does not by itself establish tail protection at a common threshold. The source reports repair on 46% of DAGs, 0.91 moves per DAG, capped at five. The experiment supports a benefit of the local gate on realized makespan within this design; runtime-repair H2 remains untested.

Table 4. Ablation at sigma = 0.3: CHARM without the chance constraint and without the repair loop, against the full method.

Variant	Expected Makespan	Realized Makespan	Miss prob.	Robustness	Wilcoxon vs CHARM (Realized)
CHARM (full)	8.07 ± 1.71	9.80	0.797	0.203	-
w/o chance constraint	8.04 ± 1.80	9.93	0.826	0.174	W = 1348, p = 0.000 (69/100)
w/o repair	7.93 ± 1.64	9.80	0.842	0.158	W = 504, p = 0.690 (22/100)

*Note: Wilcoxon tests are paired over the 100 DAGs on the realized makespan.

Table 5 varies target factor γ on the first 30 DAGs. Nominal means range from 8.32 to 8.36, realized means from 10.01 to 10.11, and own-plan exceedance fractions from 0.797 to 0.810. The small variation suggests limited sensitivity in this particular subset and parameter range. It does not establish that the gate saturates or that γ controls external deadline reliability.

Table 5. Target-factor sensitivity on the first 30 DAGs at σ = 0.3.

Target Slack Gamma	Expected Makespan	Realized Makespan	Miss Probability
1.0	8.32	10.11	0.803
1.05	8.35	10.03	0.810
1.15	8.36	10.01	0.797
1.3	8.36	10.01	0.797

Note: Miss probability uses each variant's own nominal plan, rather than its target T.

5.4 | Stress across Noise Levels RQ3

Table 6 and *Fig. 2* vary true noise while calibration stays at σ_cal = 0.3. CHARM's realized mean is 8.89 versus HEFT's 8.90 at σ = 0.2, 9.80 versus 9.94 at 0.3, and 11.05 versus 11.33 at 0.4. The gap grows over these three conditions. The multiplier parameterization also changes its mean, so this is not a pure variance or estimation-error experiment. Own-plan exceedance fractions rise for every method; at σ = 0.4 they exceed 0.9 for HEFT and CHARM, but remain 0.662, 0.837, and 0.820 for random, EFT, and greedy. These different thresholds preclude a cross-method deadline-reliability ranking. The sweep does not establish linear degradation, heavy-tail robustness, or correlated-contention performance.

Table 6. Stress across noise levels: realized makespan and deadline-miss probability per method at sigma = 0.2, 0.3, 0.4.

Method	Realized @ sigma=0.2	Miss @ 0.2	Realized @ 0.3	Miss @ 0.3	Realized @ 0.4	Miss @ 0.4
Random	20.63	0.294	21.82	0.501	23.55	0.662
EFT list	10.10	0.369	10.95	0.663	12.18	0.837
Greedy (exp.)	10.35	0.355	11.21	0.652	12.46	0.820
HEFT	8.90	0.674	9.94	0.891	11.33	0.961
CHARM	8.89	0.474	9.80	0.797	11.05	0.939

*Note: Calibration traces are fixed at sigma_cal = 0.3, so the outer levels test quantile misspecification.

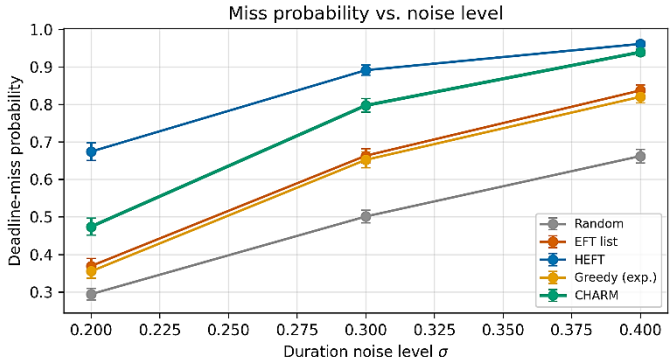


Fig. 2. Deadline-miss probability versus noise level sigma (mean over 100 DAGs; error bars are standard errors).

Note (*Fig. 2*): The inversion between the tight schedules (HEFT worst) and the conservative ones (random best) is the own-expectation artifact discussed in Section 5.2; CHARM sits below HEFT at every level.

5.5 | Statistical Analysis

Table 7 retains the reported paired Wilcoxon and Friedman results. The primary realized-makespan comparison uses 100 DAG means, rather than treating 1,000 duration draws as independent instances. A four-comparison CHARM-versus-baseline family uses Holm correction; the source's "five-way family" wording refers to five methods, not five pairwise tests. The miss-probability row reports 54 better and 46 tied DAGs alongside W = 455; for the usual signed-rank statistic, 54 same-direction nonzero differences would have minimum rank sum zero. Raw paired outcomes and the statistic convention are needed to resolve this inconsistency. Displayed p = 0.000 entries denote rounded values, not zero probability. Until regenerated, these statistics remain source-reported calculations rather than independently confirmed tests.

Table 7. Statistical analysis at sigma = 0.3: paired Wilcoxon signed-rank tests (primary endpoint, miss probability, and the Holm-corrected five-way family) and the Friedman test.

Comparison (Sigma = 0.3, Realized Makespan Unless Stated)	Statistic	p	Holm p	Outcome
CHARM vs HEFT (primary endpoint)	W = 1413	1.32e-04	-	CHARM better (66/100 DAGs; medians 9.68 vs 9.67)
CHARM vs HEFT (miss probability)	W = 455	1.77e-06	-	CHARM lower (54/100 DAGs, 46 ties)
Friedman over five methods	chi2(4) = 350.4	1.4e-74	-	Reject equality
CHARM vs random (Holm family)	W = 0	3.90e-18	1.56e-17	CHARM better (100/100)
CHARM vs eft (Holm family)	W = 0	3.90e-18	1.56e-17	CHARM better (100/100)
CHARM vs heft (Holm family)	W = 1413	1.32e-04	1.32e-04	CHARM better (66/100)
CHARM vs greedy (Holm family)	W = 3	4.27e-18	1.56e-17	CHARM better (99/100)
CHARM vs w/o chance constraint	W = 1348	0.000	-	CHARM better (69/100)
CHARM vs w/o repair	W = 504	0.690	-	Not significant

5.6 | Discussion Findings Losses and Scope

The reported realized-makespan means favour CHARM over HEFT-based assignment at intermediate and high noise, with a modest nominal-plan increase. The local-gate ablation supports a construction-time effect in this simulation. Static repair has no demonstrated mean effect at the displayed precision, and its own-plan exceedance change cannot establish fixed-deadline tail protection. These are the empirical findings supported by the reported design.

The close realized medians and 66 favourable DAGs show that CHARM does not improve every instance. HEFT has the smaller nominal plan, whereas plan-relative exceedance measures adherence to that plan and

uses method-dependent thresholds. Target-factor variation is small on the 30-DAG subset. None of these observations provides a certified workflow reliability guarantee or an operating recommendation for a production cluster.

The evidence covers 100 synthetic DAGs, one eight-worker cluster, no communication cost, log-normal multipliers, and a makespan objective. Production traces, correlated stragglers, heavy-tailed models, runtime rescheduling, replication, elasticity, energy, and multi-job fairness are untested. A future experiment should evaluate all schedulers at shared external deadlines and report feasibility, calibration, scheduler overhead, and outcome uncertainty.

The numerical results are reported at proof-of-concept scale. Independent regeneration requires the harness, raw draws, calibration samples, assignments, and execution logs. Resolution of the signed-rank inconsistency remains necessary; a verified release is not specified.

6 | Discussion Limitations and Counterarguments

6.1 | Limitations

Risk assessment is conditional on a duration model and its calibration. Shared contention can introduce dependence absent from the simulation. Quantiles estimated from 500 samples have sampling uncertainty, and changes in workload or cluster conditions can invalidate them. The local task gate used here is weaker than the joint deadline condition in *Eq. (1)*, while the own-plan exceedance measure does not assess that condition. Small-scale synthetic results cannot certify production reliability.

The general formulation concerns one job on a fixed cluster. Multi-job fairness, preemption, and elasticity require additional constraints. The tested repair pass is static and does not require demonstrated runtime preemption or replication. Runtime extensions should specify engine support and measure their cost.

6.1 | Alternative Viewpoints and Responses

Robust optimization protects against an uncertainty set [9], [10], while chance constraints depend on a probability law. Their relative conservatism depends on how the set, law, and risk tolerance are chosen. Neither inherently collapses the risk–cost trade-off to a single setting. A comparison should expose these choices and assess both at shared deadline and resource conditions.

Learned schedulers provide another possible comparator, whose calibration, sample needs, and distribution-shift behaviour require evidence. Transparent local risk scores are inspectable, but are not automatically probability certificates. The reported ablations isolate a static local gate and repair pass, without testing learning-based policies or reactive runtime rescheduling.

7 | Conclusion and Future Directions

CHARM uses worker-calibrated quantiles to guide static assignment and local schedule repair. Reported results on 100 synthetic DAGs give mean realized makespan 9.80 versus 9.94 for HEFT-based assignment at $\sigma = 0.3$ under common dispatch, with nominal makespan 8.07 versus 7.82. The local-gate ablation raises realized mean by 0.13, whereas static repair leaves the displayed mean unchanged. The gap in realized means grows across the three tested noise levels. These results support further evaluation of the heuristic, without demonstrating the general framework's chance constraint or runtime-repair aims.

The next evaluation should first regenerate the paired statistics and specify whether nominal durations are centered stochastic means. Shared external deadlines are needed to compare actual deadline reliability, and joint independent certification is needed before using a risk guarantee. Faithful insertion-schedule execution, communication costs, trace-derived workloads, calibration uncertainty, and correlated durations would strengthen the benchmark. Runtime repair should then be tested as a separate implemented mechanism, with its computation and resource costs reported.

Author Contributions

M. P.: Study conception and design, data collection, data analysis, interpretation of results, and drafting of the original manuscript. S. K.: Study conception and design, data collection, data analysis, interpretation of results, and drafting of the original manuscript. A. C.: Data analysis, interpretation of findings, and critical revision of the manuscript. S. Q. H.: Data analysis, interpretation of findings, and critical revision of the manuscript. P. J. D. W.: Study supervision, critical review, and editing of the manuscript. All authors read and approved the final version of the manuscript.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data Availability

The reported workload consists of 100 generated layered DAGs on an eight-worker simulated cluster. Exact graph instances, worker factors, multiplier calibration traces, and raw duration draws require the experiment record.

Code Availability

A verified public release of the simulation harness and raw outcomes is not specified.

Conflicts of Interest

The authors declare no conflicts of interest.

Generative AI Disclosure

AI-assisted tools were used in drafting and revising the manuscript text. The authors retain responsibility for verification of the scientific content and the applicable disclosure.

References

[1] Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., … ., & Wenger, K. (2015). Pegasus, a workflow management system for science automation. *Future generation computer systems*, *46*, 17–35. https://doi.org/10.1016/j.future.2014.10.008

[2] Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., & Vahi, K. (2013). Characterizing and profiling scientific workflows. *Future generation computer systems*, *29*(3), 682–692. https://doi.org/10.1016/j.future.2012.08.015

[3] Topcuoglu, H., Hariri, S., & Wu, M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, *13*(3), 260–274. https://doi.org/10.1109/71.993206

[4] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., & Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. *Proceedings of the third ACM symposium on cloud computing* (pp. 1–13). Association for Computing Machinery. https://doi.org/10.1145/2391229.2391236

[5] Pinedo, M. L. (2012). *Scheduling: Theory, algorithms, and systems*. Springer. https://doi.org/10.1007/978-1-4614-2361-4

[6] Möhring, R. H., Schulz, A. S., & Uetz, M. (1999). Approximation in stochastic scheduling: The power of LP-based priority policies. *Journal of the association for computing machinery,*, *46*(6), 924–942. https://doi.org/10.1145/331524.331530

[7] Kwok, Y. K., & Ahmad, I. (1999). Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM computing surveys.*, *31*(4), 406–471. https://doi.org/10.1145/344588.344618

[8] Prékopa, A. (1995). *Stochastic programming*. Springer. https://doi.org/10.1007/978-94-017-3087-7

[9] Bertsimas, D., & Sim, M. (2004). The price of robustness. *Operations research*, *52*(1), 35–53. https://doi.org/10.1287/opre.1030.0065

[10] Ben-Tal, A., Nemirovski, A., & El Ghaoui, L. (2009). *Robust optimization*. Princeton university press. https://doi.org/10.1515/9781400831050

[11] Boucheron, S., Lugosi, G., & Massart, P. (2013). *Concentration inequalities: A nonasymptotic theory of independence*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199535255.001.0001

[12] Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and hall/cRc. https://doi.org/10.1201/b16018

[13] Abrishami, S., Naghibzadeh, M., & Epema, D. H. J. (2013). Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. *Future generation computer systems*, *29*(1), 158–169. https://doi.org/10.1016/j.future.2012.05.004

[14] Mao, M., & Humphrey, M. (2011). Auto-scaling to minimize cost and meet application deadlines in cloud workflows. *Proceedings of 2011 international conference for high performance computing, networking, storage and analysis*. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2063384.2063449

[15] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … ., & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[16] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., … ., & Contributors, S. 1. . (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[17] Seabold, S., Perktold, J., & others. (2010). Statsmodels: Econometric and statistical modeling with python. *Proceedings of the python in science conference* (pp. 92–96). SciPy. https://doi.org/10.25080/Majora-92bf1922-011

[18] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, *1*(6), 80–83. https://doi.org/10.2307/3001968

[19] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of machine learning research*, *7*(1), 1–30. https://jmlr.org/papers/v7/demsar06a.html

[20] Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, *6*(2), 65–70. https://www.jstor.org/stable/4615733