PT Jurnal Cendekia Indonesia
Journal of Information Technology and Informatics Engineering
Homepage: https://journal.jci.co.id/jitie
Vol. 02 No. 04 (2026) Page: 01-15
E-ISSN: 3123-7827

Critical-Path and Uncertainty-Aware Dynamic Scheduling for Resource-Constrained LLM Multi Agent Teams

Shawn Tan[1], Emma Wilson[2], Vera Lu[3]

[1]Department of Computer Science, Boston University, Boston, MA, USA
[2]Department of Computer Science, University of Notre Dame, Notre Dame, IN, USA
[3]Department of Computer Science, University of Minnesota, Minneapolis, MN, USA

Email: [1*]s_tan_bu@gmail.com

Abstract

Large-language-model multi-agent systems can decompose complex work, but adding agents does not guarantee shorter completion time: precedence constraints, uneven task sizes, and cross-agent handoffs can erase the benefit of parallelism. This study evaluates six deterministic scheduling policies on the released MultiAgentBench/MARBLE coding corpus. All 100 tasks were converted into requirement-level directed acyclic graphs using temporal, integration, testing, adaptive-feedback, and final-stage dependency evidence in the task specifications. The resulting corpus contains 621 requirement nodes, 780 edges, 40 application topics, and four coordination categories. Word counts serve as normalized work units; consequently, the study measures scheduling behavior rather than live model latency or software correctness. We executed 1,800 schedules spanning FIFO, Round-Robin, static-role, critical-path, uncertainty-weighted critical-path, and a critical-path policy with uncertainty and communication control (CP-UC), each at 2, 4, and 8 concurrent workers. Critical-path scheduling produced modest mean improvements over FIFO of 0.80%, 0.96%, and 0.51%, respectively. CP-UC reduced mean communication work by 7.27%-9.27% and propagation exposure by 1.66%-2.26% relative to FIFO, but increased mean makespan by 2.10%-3.39% on the paired percentage measure. Category analysis locates this cost mainly in test-case and dependency-heavy tasks. The results show that critical-path priority is a useful low-risk default, whereas communication control should be treated as an explicit latency-coordination trade-off rather than a universally faster scheduler..

Keywords: LLM Agents; Multi-Agent Systems; DAG Scheduling; Critical Path; Uncertainty-Aware Scheduling; Communication Budget; Multiagentbench; MARBLE.

Abstrak

Sistem multi-agen model bahasa besar dapat menguraikan pekerjaan kompleks, tetapi penambahan agen tidak menjamin waktu penyelesaian yang lebih singkat: kendala prioritas, ukuran tugas yang tidak merata, dan serah terima antar agen dapat menghilangkan manfaat paralelisme. Studi ini mengevaluasi enam kebijakan penjadwalan deterministik pada korpus pengkodean MultiAgentBench/MARBLE yang telah dirilis. Semua 100 tugas dikonversi menjadi grafik asiklik terarah tingkat persyaratan menggunakan bukti ketergantungan temporal, integrasi, pengujian, umpan balik adaptif, dan tahap akhir dalam spesifikasi tugas. Korpus yang dihasilkan berisi 621 node persyaratan, 780 edge, 40 topik aplikasi, dan empat kategori koordinasi. Jumlah kata berfungsi sebagai unit kerja yang dinormalisasi; akibatnya, studi ini mengukur perilaku penjadwalan daripada latensi model langsung atau kebenaran perangkat lunak. Kami menjalankan 1.800 penjadwalan yang mencakup FIFO, Round-Robin, peran statis, jalur kritis, jalur kritis berbobot ketidakpastian, dan kebijakan jalur kritis dengan kontrol ketidakpastian dan komunikasi (CP-UC), masing-masing pada 2, 4, dan 8 pekerja bersamaan. Penjadwalan jalur kritis menghasilkan peningkatan rata-rata yang moderat dibandingkan FIFO sebesar 0,80%, 0,96%, dan 0,51%, berturut-turut. CP-UC mengurangi rata-rata pekerjaan komunikasi sebesar 7,27%-9,27% dan paparan propagasi sebesar 1,66%-2,26% relatif terhadap FIFO, tetapi meningkatkan rata-rata makespan sebesar 2,10%-3,39% pada ukuran persentase berpasangan. Analisis kategori menunjukkan biaya ini terutama terletak pada tugas-tugas yang banyak melibatkan kasus uji dan ketergantungan. Hasilnya menunjukkan bahwa prioritas jalur kritis adalah pilihan default berisiko rendah yang bermanfaat, sedangkan kontrol komunikasi harus diperlakukan sebagai pertimbangan eksplisit antara latensi dan koordinasi, bukan sebagai penjadwal yang secara universal lebih cepat.

***Kata Kunci**: Agen LLM; Sistem Multi-Agen; Penjadwalan DAG; Jalur Kritis; Penjadwalan Sadar Ketidakpastian; Anggaran Komunikasi; Multiagentbench; MARBLE.*

A. INTRODUCTION

Agentic systems built around large language models (LLMs) increasingly assign planning, implementation,

Received: 01-08-2026 | Revisio: 09-08-2026 | Published Online: 10-08-2026

testing, critique, and synthesis to distinct agents. MultiAgentBench formalizes cooperation and competition across heterogeneous environments [1]. AutoGen supplies event-driven multi-agent conversation [2], while CAMEL studies role-playing coordination [3]. AgentVerse [4], MetaGPT [5], and ChatDev [6] organize agents around explicit roles or software-development procedures; GPTSwarm treats agents and their interactions as an optimizable graph [7]. Together, these systems make structured communication operational. Yet most evaluations emphasize answer quality or task success. They say much less about a prior systems question: given a fixed set of work items and a limited number of concurrent agents, which ready item should run next, and which agent should receive it?

This question matters because the apparent parallelism of an LLM team is bounded by its dependency graph. A tester cannot validate functionality that has not been implemented, an integration step may wait for several components, and an adaptive requirement may depend on earlier feedback. ReAct interleaves model reasoning with externally grounded actions [12]; Graph of Thoughts [13] and Tree of Thoughts [14] expose graph-structured and branching reasoning; and LLMCompiler schedules tool calls with dependency-aware parallelism [15]. These approaches demonstrate that intermediate structure is useful, but they do not eliminate resource contention among concurrent workers.

Classical scheduling theory formalizes precedence as a DAG. HEFT combines upward ranks with earliest-finish assignment for heterogeneous processors [16], and PEFT uses an optimistic cost table to improve look-ahead [17]. The Kwok-Ahmad survey situates such algorithms within static DAG scheduling [18]. Graham's list-scheduling analysis [19] and the critical-path method of Kelley and Walker [20] supply the simpler foundations used here. LLM teams add two complications: work estimates are text-derived rather than profiled kernels, and a dependency edge can require an expensive context transfer. The present experiment therefore retains deterministic list scheduling but makes the work unit, edge evidence, and handoff penalty explicit.

Communication is not a secondary detail. Multi-agent debate can improve factual reasoning through independent proposals and critique [8], while DyLAN dynamically selects agents and communication structures during inference [9]. Scaling results are not uniform: More Agents Is All You Need finds benefits from repeated sampling and aggregation [10], and Mixture-of-Agents layers outputs from multiple models [11], but both increase inference and coordination demand. A scheduling analysis should therefore separate the value of a larger candidate pool from the ability to execute dependent work concurrently.

Uncertainty also needs a bounded interpretation. Verbalized confidence can be elicited from LLMs but requires calibration [21], and semantic entropy detects meaning-level variation across generations [22]. Neither measurement is available in static task specifications, so this study uses only a lexical ordering proxy and never interprets it as probability. Communication-efficient agent work further motivates the locality objective: Cut the Crap develops an economical communication pipeline [23], AFlow searches over workflow structures [24], WorFBench evaluates workflow generation at node, edge, subgraph, and path levels [25], and Optima jointly trains multi-agent communication for effectiveness and efficiency [26]. Our narrower question is complementary: with the task graph fixed, what latency and risk-ordering changes arise from deterministic scheduling alone?

The present study examines this systems layer with the 100 released coding tasks in MultiAgentBench/MARBLE [1], [27]. The coding corpus is suitable because each record provides a task description, a list of implementation requirements, a topic category, and a coordination category. The repository also contains a corresponding coding configuration for every record, enabling a direct consistency check. We convert the requirements into evidence-based DAGs and compare six policies under team capacities of two, four, and eight. The work model is deliberately narrow: one word in a requirement equals one normalized scheduling work unit. This unit is not seconds, tokens billed by a provider, or observed model inference time. Schedule makespan therefore characterizes the structure and allocation of specified work; it does not establish MARBLE task success, generated-code quality, or end-to-end latency.

Three research questions organize the evaluation. RQ1: Do critical-path and uncertainty-weighted priorities shorten requirement-DAG makespan relative to FIFO, Round-Robin, and static-role assignment? RQ2: How do team size and coordination category change utilization, handoffs, and the gap to an analytical lower bound? RQ3: Can CP-UC lower communication and uncertainty exposure without an unacceptable makespan penalty? The third question is intentionally two-sided: a communication-aware policy is useful even if it is slower, provided the size and location of that cost are measured rather than hidden.

The study makes four contributions. First, it supplies a traceable transformation from 621 released requirements to 780 precedence edges, with every edge grounded in explicit lexical or category-specific evidence. Second, it evaluates a complete 6-by-3-by-100 scheduling matrix and reports ten tables and ten figures from the saved run-level records. Third, it introduces two scheduling diagnostics: normalized completion time for the highest-uncertainty requirement quartile and risk-weighted propagation exposure. Fourth, it reports negative findings alongside gains. Critical-path priority improves mean makespan only slightly, most tasks tie FIFO, and the communication-aware policy pays a measurable latency cost.

Forecasting, Admission Control, And Resource-Aware Orchestration

Scheduling begins after a workload has been admitted and decomposed, so it should be viewed as one layer above classical DAG allocation [16]-[20] and within a broader

resource-control stack. Safe capacity forecasting for heterogeneous GPU clusters combines time-series foundation models with calibrated uncertainty [28], while learned GPU-memory prediction estimates task-level resource requirements before execution [29]. Cross-cloud transfer learning addresses workload and topology shift [30], multi-horizon demand models connect workload semantics to operational risk curves [31], and conformal demand envelopes bound oversubscription risk [32]. These studies motivate treating duration and uncertainty estimates as auditable scheduler inputs rather than implicit constants.

Cold-start forecasting for new inference tenants [33], token-burst-aware capacity planning [34], and power-aware inventory planning [35] extend this logic from aggregate demand to tenant-, request-, and job-level decisions. Hybrid-cloud LLM deployment introduces dynamic local-versus-cloud routing [36], while profit-aware spot-GPU admission control couples capacity decisions to cost-sensitive policy memos [37]. Relative to LLMCompiler's dependency-aware tool-call scheduling [15], these studies operate one level earlier by shaping the pool of work and capacity available to the scheduler. The present experiment holds node work fixed to isolate allocation behavior, but a live system would need these signals to estimate service times and admission risk.

Evidence-grounded orchestration and incident intelligence

Resource decisions are useful only if the underlying work graph and operational state are trustworthy. ReAct's action grounding [12] and graph-structured reasoning [13] show why intermediate evidence matters. Multi-source root-cause attribution fuses CPU and network signals [38], evidence-grounded DevOps RAG answers questions over private operations documents [39], and bilingual OpsLLM combines retrieval with root-cause analysis and alert summarization [40]. Retrieved normal prototypes have also been used to explain OpenStack failure-injection anomalies [41]. Together, these systems motivate explicit edge evidence and provenance in the requirement DAG.

Other work moves from diagnosis to actionable automation. Findable-then-explainable code intelligence integrates retrieval with summary generation [42]; self-supervised LogBERT-style detection evaluates sequence anomalies on a reproducible HDFS benchmark [43]; and retrieval-grounded HDFS analysis produces deterministic failure narratives [44]. Conformal alert control translates anomaly scores into auditable incident tickets [45], a DevOps copilot evaluates runbook generation together with command safety [46], and cost-aware routing selects among parsing, detection, diagnosis, and summarization operations [47]. Trajectory reliability prediction [48] and noisy-neighbor-aware VM degradation modeling [49] offer complementary mechanisms for prioritizing work whose failure may propagate. These extensions parallel AFlow and WorFBench [24], [25] in treating workflows as inspectable objects rather than opaque conversations.

Human-facing visualization completes the control loop. Evidence-constrained incident cards translate distributed log evidence into SRE actions [50], and visual brief cards convert infrastructure forecasts into dashboard decisions [51]. Optima's communication-efficiency objective [26] supports the same separation adopted here: communication work, uncertainty exposure, and latency should remain visible as distinct quantities rather than be collapsed into one opaque score.

Evidence Reliability, Adversarial Safety, And Selective Action

RAG reliability offers design patterns for agent handoffs. Word-level hallucination detection and source attribution [52], evidence-calibrated abstention for unanswerable questions [53], and deterministic provenance explanations [54] distinguish whether evidence is present from whether a generated claim is acceptable. Budgeted multi-hop retrieval makes the resource constraint explicit [55], while retrieval-quality prediction estimates when additional search or review is warranted [56]. Combined with semantic-entropy diagnostics [22], such signals could weight dependencies, trigger verification nodes, or defer a risky handoff.

Adversarial safety adds a different failure mode. Continual red-teaming uses online guardrail updates and distillation [57]; privacy and data-integrity risk cards make escalation states legible to human overseers [58]; and VerifySafe applies evidence-based self-verification under adversarial prompts [59]. Patient-facing safety cards illustrate how calibrated risk can be communicated without hiding uncertainty [60], while behavior-level jailbreak resistance combines staged refusal with utility preservation [61]. Multi-agent debate [8] can provide useful critique, but these studies show that critique must itself be governed and evidenced.

Security analytics further explains why early triage can be valuable. Host-based intrusion detection localizes anomalous system-call windows and generates forensic narratives [62], interpretable CloudTrail models smooth attack-chain stages over event sequences [63], and deep autoencoders classify IoT traffic anomalies [64]. Language-guided feature selection for CICIDS2017 [65] and TinyLLM-assisted intrusion detection for real-time IoT networks [66] illustrate how model cost, explanation, and response time interact. In scheduling terms, a high-impact security node may deserve early execution even when it is not on the nominal longest path.

Calibration, Uncertainty, And Human-Legible Priorities

Uncertainty-aware systems repeatedly separate ranking quality from decision quality. Confidence elicitation in LLMs requires calibration [21], and late fusion for 3D perception combines confidence calibration with learned fusion rules [67]. Calibrated credit-default tiering adds uncertainty-driven rejection [68], while model-risk-oriented probability-of-default workflows combine calibration, distribution-free uncertainty quantification, and

SHAP explanations [69]. Counterfactual fair-credit scoring [70], extreme-imbalance fraud detection [71], layout-aware progressive PDF rendering [72], and calibrated resume-job matching with selective refusal [73] all emphasize operating points and deferral rather than raw accuracy alone. These examples motivate reporting both latency and exposure here, although the lexical uncertainty score is deliberately not interpreted as probability.

Formal uncertainty quantification for spectral estimators [74] reinforces the need to characterize estimator error under structural constraints. Scientific-search evidence cards [75] show how provenance and ranking trust can be communicated through visual hierarchy. A future live scheduler could similarly expose why a requirement was advanced, delayed, verified, or kept on the same worker, instead of presenting the priority score as self-explanatory.

Offline policy evaluation, drift, and state management

Because agent policies may be costly or unsafe to test online, offline evaluation is a useful methodological analogue. Privacy-robust incrementality estimation [76], offline counterfactual evaluation for advertising and recommendation slots [77], and conservative off-policy selection for dynamic bidding and ranking [78] emphasize support, propensity, and policy-risk diagnostics. Risk-aware budget-constrained auto-bidding [79] and adaptive strategies for hybrid auctions [80] show how a controller can trade reward against explicit constraints. GPTSwarm's graph optimization [7] and DyLAN's dynamic network selection [9] pursue related policy-level adaptation; the deterministic protocol here is narrower, but its paired task-level comparisons serve a similar governance purpose.

Operational policies also require drift and state management. DriftGuard couples multi-signal warning with safe retraining or rollback [81], LLM-assisted uplift modeling links feature interactions to interpretable audience-creative-channel policies [82], and counterfactual learning-to-rank evaluates advertising policies under logged feedback [83]. Confidence-gated long-term memory introduces time-aware retrieval together with update and forgetting rules [84], while federated topic-preference learning adds differential privacy [85]. Role-based agent frameworks [3]-[6] make persistent specialization attractive, but these studies suggest that scheduler state, worker profiles, and historical duration estimates should be updated under explicit confidence and forgetting controls rather than accumulated indefinitely.

B. METHODS

Dataset, Task Audit, And DAG Construction

The source corpus is the coding benchmark distributed with MultiAgentBench [1] in the official MARBLE repository [27]. The benchmark file contained 100 JSONL records, and its SHA-256 digest in the run manifest was 5b2165d9ca80758c58cd42d889d3c4b8c6f658770652578bb81583a1ae65e90f. We also matched each record to its repository coding configuration. All 100 content fields and all 621 requirements occurred in their corresponding configurations. Tasks cover 40 topic categories, with two or three tasks per topic, and four coordination categories: 28 adaptive tasks, 25 dependency tasks, 19 multi-domain tasks, and 28 test-case tasks.

Each nonempty requirement became one node. Node work (w_i) was the number of alphanumeric word tokens in its requirement text. Across the corpus, node work ranged from 12 to 129 units, with mean 30.562 and median 27. A task contained 3-12 nodes (mean 6.21) and 110-360 total work units (mean 189.79). We assigned a functional role by counting fixed keyword sets in repository order: implementation mapped to developer, testing to tester, optimization to performance engineer, debugging to debugger, and documentation to technical writer. Ties followed insertion order. This produced 539 developer, 52 tester, 26 performance-engineer, 2 debugger, and 2 technical-writer nodes. Roles were used only by the static-role baseline.

Precedence extraction always created forward edges, so acyclicity followed from construction and was checked afterward. Temporal phrases such as "after," "before," "once," "depends on," "in place," and "prior to" linked a requirement to the earlier or later requirement with the strongest content-term overlap; when no overlap existed, the nearest valid requirement supplied the edge. Category rules then represented coordination semantics. Testing requirements followed earlier implementation, performance, or debugging work. Integration phrases created fan-in from earlier build work. Multi-domain integration requirements received edges from relevant earlier build nodes; adaptive and feedback requirements depended on up to the two latest build nodes; and final-stage phrases linked all preceding requirements. Duplicate and self-edges were removed. The process extracted 780 edges, or 7.8 per task on average. These rules operationalize task specifications; they are not a claim that the graph captures every dependency an executing model might discover.

Table 1 profiles the four coordination categories. Dependency tasks had the greatest mean work (215.96) and a long mean critical path (180.68), while test-case tasks had more requirements (7.39) and edges (321 total) but a shorter mean critical path (76.82). This contrast is useful: dependency tasks are predominantly serial, whereas test-case graphs expose more parallel width before validation fan-in.

Table 1 Coding Benchmark And Extracted DAG Characteristics.

Category	Tasks	Req.	Edges	Mean work	Mean CP
Adaptive	28	186	201	171.6	109.9
Dependency	25	124	161	216.0	180.7
Multi-domain	19	104	97	196.4	111.1
Test-case	28	207	321	180.1	76.8
Total	100	621	780	189.8	118.5

Note. Work and critical path (CP) are measured in requirement word-work units.

Figure 1 summarizes the audit-to-evaluation pipeline, and Figure 2 shows the empirical distributions of requirement

count, lexical work, and uncertainty by coordination category.

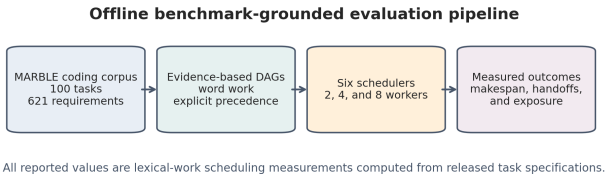

Figure 1 Offline Evaluation Pipeline From The Released MARBLE Coding Corpus Through DAG Extraction, Resource-Constrained Scheduling, And Metric Computation.

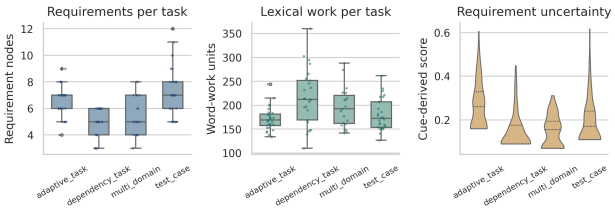

Figure 2 Distribution Of Requirement Counts, Lexical Work, And Uncertainty Across The Four Coordination Categories.

For each node, a deterministic lexical proxy, u_i in [0.02, 1], combined a coordination-category prior and cue density. The prior was 0.16 for adaptive, 0.09 for dependency, 0.07 for multi-domain, and 0.11 for test-case tasks. Fourteen cues - including "adaptive," "dynamic," "unexpected," "feedback," "conflict," "failure," "unknown," "robust," and "concurrent" - contributed one count each; conditional terms ("if," "when," "unless," and "depending") contributed half a count. For a requirement containing n_i words, u_i = clip[p_c + 0.58(q_i + 0.5d_i)/sqrt(n_i), 0.02, 1]. The observed mean was 0.196, the median was 0.165, and the maximum was 0.618. This score is a priority feature derived from wording, not calibrated predictive confidence.

Let G = (V, E) be a task DAG, let K be 2, 4, or 8, and let P_i denote the predecessors of node i. A worker could start node i after becoming free and after every predecessor completed. If predecessor j ran on another worker, its transfer delay was h_j = max[1, ceil(0.15w_j)]; otherwise the delay was zero. The resulting timing equations were s_i = max[a_k, max over j in P_i of (f_j + h_j I[k_j != k])] and f_i = s_i + w_i. Except where pinned by a baseline, a node was assigned to the worker giving earliest finish, with fewer new handoffs and lower worker index as deterministic tie-breakers.

The six policies in Table 2 were evaluated. FIFO selected the lowest-index ready node. Round-Robin used the same node order but pinned node i to worker i modulo K. StaticRole pinned the five extracted roles to workers in fixed cyclic order. CriticalPath selected the ready node with the largest bottom level, b_i = w_i + max over each successor j of b_j. UncertaintyCP used b_i^u = w_i(1 + u_i) + max over each successor j of b_j^u. The proposed CP-UC used the same uncertainty-weighted node priority and a locality-aware worker objective. For each candidate worker it minimized, lexicographically, budget violation and then f_i + 3w_med ΔH + 10w_med ΔB, where w_med is median node work, ΔH is the new handoff count, and ΔB is the amount above B = ceil(0.85|E|). The budget is therefore a strong scheduling preference; feasibility still depends on the already committed assignments.

Table 2 Scheduling Policies And Fixed Protocol Parameters.

Policy	Ready-task priority	Worker selection	Control
FIFO	Ready index	Earliest finish	None
Round-robin	Ready index	Fixed cyclic worker	No migration
Static-role	Ready index	Repository role pinned to worker	No migration
CP	Longest bottom level	Earliest finish	15% handoff delay
U-CP	Risk-adjusted bottom level	Earliest finish	Uncertainty priority
CP-UC	Risk-adjusted bottom level	Locality-aware finish	85% edge budget; locality penalty 3.0

Note. All policies are nonpreemptive and use identical extracted work and precedence constraints.

This formulation connects classical precedence scheduling [16]-[20] with uncertainty-sensitive agent orchestration [21]-[25]. It also differs from approaches that ask another LLM to act as a manager: all priorities, costs, and tie-breaks are fixed functions, making scheduler behavior directly auditable. Optima shows why communication efficiency is increasingly central to multi-agent reasoning systems [26]. Forecasting and admission-control studies [28], [31], [34], [37] identify live workload and capacity signals that could replace fixed word-work units, while evidence-grounded operations pipelines [38], [41], [46], [50] motivate retaining traceable dependency and handoff provenance.

Metrics And Statistical Protocol

The primary outcome was makespan, C_max = max_i(f_i), in word-work schedule units. The lower bound was L = max(CP, W/K), where CP is the unadjusted critical-path length and W is total node work. We measured lower-bound gap (C_max/L - 1), productive utilization W/(KC_max), load coefficient of variation, load Gini coefficient, number and fraction of dependency edges crossing workers, total communication work, and communication share H/(H + W).

Two risk-ordering measures used finish times rather than assumed failures. First, high-uncertainty completion was the work-weighted mean finish time of nodes in the within-task upper uncertainty quartile, divided by makespan. Lower values mean these nodes finish earlier. Second, descendant work D_i was the total work reachable from node i, and risk weight was r_i = u_i(w_i + D_i). Propagation exposure was the sum of r_i f_i divided by C_max times the sum of r_i. A lower value means uncertain, high-downstream-impact work remains unresolved for less of the normalized schedule.

Every policy ran once on each task at each team size, yielding 100 × 6 × 3 = 1,800 deterministic schedules. For mean per-task percentage change from FIFO, 2,000

bootstrap resamples with seed 20250805 formed percentile 95% intervals. CP-UC was additionally compared with FIFO, CriticalPath, and UncertaintyCP using paired two-sided Wilcoxon signed-rank tests for makespan, handoff rate, and propagation exposure at each K. Holm correction covered all 27 comparisons. Sensitivity runs varied handoff cost over 0, 0.10, 0.20, and 0.30 and the CP-UC budget ratio over 0.45, 0.65, and 0.85. The primary experiment completed in 1.204 s under Python 3.12.13 on Linux; scheduler CPU measurements were local orchestration costs, not agent inference times.

Execution Controls And Internal Validity

The experimental implementation used a single deterministic path from a benchmark record to its graph, schedule, and metric row. Task IDs joined each policy output to the same FIFO task before calculating percentage change or a paired test. Schedule validation checked that every predecessor finished before its dependent node started, including any cross-worker delay; each node was assigned exactly once; makespan was no smaller than the analytical lower bound; utilization remained in [0,1]; and handoffs did not exceed the number of DAG edges. The extracted graph file, node-feature file, 1,800-row schedule file, sensitivity runs, audit manifest, and representative trace were retained separately. These controls prevent a category aggregate or plot from silently using a different task set. The paired, task-joined protocol follows the same governance logic as offline policy evaluation [76]-[80] and drift-aware rollback [81], although it estimates schedule outcomes rather than causal treatment effects.

The lower bound excludes communication and is therefore optimistic. It answers how close a policy comes to the best schedule permitted by work and precedence alone, not how close it comes to an unknown optimum under handoff delays. Similarly, the scheduler's CPU time measures the Python decision procedure only. Mean policy CPU costs ranged from 28.77 microseconds for Round-Robin at four workers to 136.93 microseconds for CriticalPath at two workers; these sub-millisecond values make policy computation negligible within this offline model but say nothing about production orchestration overhead.

The evaluation isolates scheduling from model stochasticity. That improves paired comparability because all policies receive identical node work and uncertainty values, but it also narrows external validity. A policy cannot gain from better code, a shorter answer, or a model's discovery of a missing dependency. The analysis should therefore be read as a controlled study of task-graph structure and allocator behavior.

C. RESULTS AND DISCUSSION

Aggregate Makespan And Efficiency

Table 3 reports all 18 policy-capacity combinations. FIFO mean makespan decreased from 135.32 at (K=2) to 121.13 at (K=4) and 119.67 at (K=8). CriticalPath was the fastest aggregate policy at every capacity: 134.55, 120.41, and 119.32. Its mean task-level improvements over FIFO were 0.80% (95% bootstrap interval 0.05%-1.61%), 0.96% (0.42%-1.61%), and 0.51% (0.20%-0.91%). These are consistent but small gains. CriticalPath beat FIFO on only 22, 13, and 9 tasks at (K=2,4,8), tied it on 68, 87, and 91, and was worse on 10 tasks only at (K=2). The large tie share shows that many graphs either have a single consequential ready order or reach the critical-path lower bound under both policies.

UncertaintyCP closely tracked CriticalPath. Its mean makespans were 134.57, 120.52, and 119.44, with mean paired improvements over FIFO of 0.76%, 0.80%, and 0.33%. The (K=2) interval (−0.05%-1.65%) and (K=8) interval (−0.002%-0.76%) crossed or touched zero. Lexical risk weighting therefore changed ordering in some tasks but did not improve aggregate latency beyond the unadjusted bottom level.

Table 3 Aggregate Scheduling Performance Over 100 Task Dags.

Policy	K	Cmax	Gap (%)	Util. (%)	Handoff (%)
FIFO	2	135.32	7.92	73.94	25.99
FIFO	4	121.13	3.79	45.29	34.33
FIFO	8	119.67	1.45	23.41	35.82
Round-robin	2	152.36	21.74	65.83	62.28
Round-robin	4	135.66	16.45	40.41	90.42
Round-robin	8	131.84	10.48	21.76	98.59
Static-role	2	186.74	57.09	51.15	22.83
Static-role	4	182.16	80.30	26.40	30.27
Static-role	8	182.05	80.45	13.22	30.43
CP	2	134.55	6.87	74.73	28.54
CP	4	120.41	2.59	45.98	34.58
CP	8	119.32	0.90	23.59	35.63
U-CP	2	134.57	6.90	74.71	28.27
U-CP	4	120.52	2.76	45.89	34.32
U-CP	8	119.44	1.08	23.54	35.71
CP-UC	2	137.97	10.07	72.64	24.16
CP-UC	4	124.64	7.61	44.05	32.24
CP-UC	8	122.61	4.25	22.90	34.09

Note. Cmax is mean makespan. Gap is relative to max(CP, W/K).

Figure 3 visualizes mean makespan and task-level uncertainty bands. Round-Robin was 8.99%-13.12% worse than FIFO on the paired improvement measure, while StaticRole was 44.66%-76.88% worse. Pinned assignments prevented the earliest-finish allocator from responding to dependency release times. StaticRole also left most additional workers ineffective: its mean makespan changed from 186.74 to 182.05 between two and eight workers.

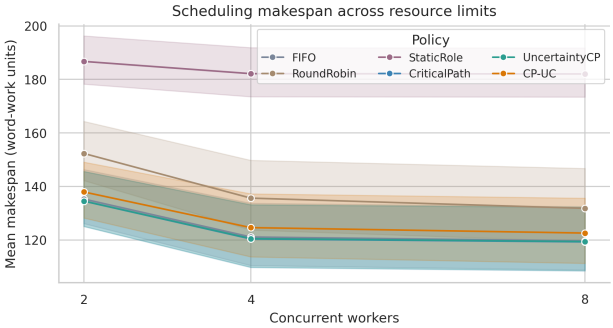

Figure 3 Mean Makespan Across Six Scheduling Policies And Three Worker Limits; Shaded Bands Show 95% Bootstrap Confidence Intervals Over 100 Tasks.

in Figure 4 clarify the capacity limit. CriticalPath's mean lower-bound gap fell from 6.87% at (K=2) to 0.90% at (K=8), yet utilization fell from 74.73% to 23.59%. This is not contradictory. As (K) increases, (W/K) declines until the critical path dominates (L); spare workers then remain idle even when the realized schedule is near optimal for the extracted graph.

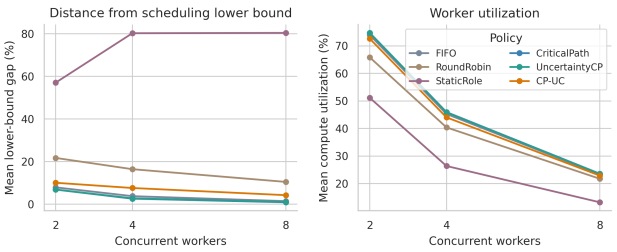

Figure 4 Distance From The Scheduling Lower Bound And Compute Utilization As Concurrency Increases.

Table 4 separates coordination categories. CriticalPath benefited test-case tasks most, improving on FIFO by 1.47%, 2.80%, and 1.58% across (K=2,4,8). At (K=2), it improved adaptive tasks by 1.63%, but dependency and multi-domain means were slightly worse (−0.41% and −0.21%). At larger capacities, CriticalPath tied FIFO for dependency and multi-domain categories because their longest chains or fan-in constraints already determined completion time. These results agree with the corpus audit: test-case tasks combine many nodes with relatively short critical paths, leaving useful ordering freedom.
Table 4. Mean makespan change relative to FIFO by Coordination Category.

Table 4 Mean Makespan Change Relative To Fifo By Coordination Category.

Category	K	Cp (%)	U-Cp (%)	Cp-Uc (%)
Adaptive	2	1.63	1.65	1.00
Adaptive	4	0.19	0.19	-0.39
Adaptive	8	0.00	0.00	-0.58
Dependency	2	-0.41	-0.50	-3.37
Dependency	4	0.00	0.00	-2.91
Dependency	8	0.00	0.00	-2.91
Multi-Domain	2	-0.21	-0.12	-0.12
Multi-Domain	4	0.00	0.00	0.00
Multi-Domain	8	0.00	0.00	0.00
Test-Case	2	1.47	1.44	-4.50
Test-Case	4	2.80	2.33	-8.78
Test-Case	8	1.58	1.04	-6.50

Note. Positive Values Indicate Lower Makespan Than Fifo; Negative Values Indicate A Penalty.

Figure 5 Focuses On Cp-Uc And Reveals Its Main Negative Result. Cp-Uc Improved Adaptive Tasks By 1.00% At (K=2), But It Regressed By 0.39% And 0.58% At (K=4,8). Its Penalties Were 2.91%-3.37% For Dependency Tasks And 4.50%-8.78% For Test-Case Tasks; Multi-Domain Differences Stayed Between −0.12% And 0%. Locality Pressure Is Most Expensive When Test Nodes Need Parallel Predecessor Completion: Keeping Related Work Together Reduces Transfers But Delays Available Parallel Work.

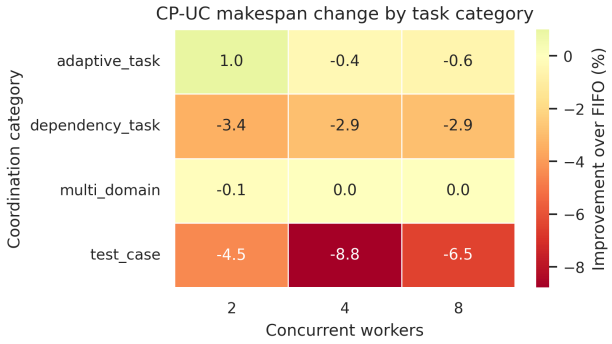

Figure 5 Cp-Uc Makespan Change Relative To Fifo By Coordination Category And Worker Limit.

Resource Scaling And Communication Trade-Offs

Table 5 Reports Task-Level Speedup Relative To Each Policy's Own (K=2) Schedule. Criticalpath Achieved Mean Speedups Of 1.197 At (K=4) And 1.225 At (K=8); Fifo Reached 1.195 And 1.230. Median Speedup Was Only 1.035 For Criticalpath And 1.000 For Fifo, Again Indicating That A Minority Of Parallel Graphs Raises The Mean. Round-Robin Had The Highest (K=8) Mean Ratio, 1.279, But Remained Slower In Absolute Makespan Because It Started From A Poor (K=2) Schedule. Staticrole Achieved Only 1.034 At (K=8). Capacity Scaling Must Therefore Be Interpreted Jointly With Absolute Time.

Table 5 Scale-Out Speedup Relative To Each Policy's Two-Worker Schedule.

Policy	Mean K4	Mean K8	Median K4	Median K8
FIFO	1.195	1.230	1.000	1.000
Round-robin	1.198	1.279	1.169	1.167
Static-role	1.033	1.034	1.000	1.000
CP	1.197	1.225	1.035	1.035
U-CP	1.195	1.223	1.035	1.035
CP-UC	1.183	1.229	1.000	1.000

Note. A median of 1.000 denotes no improvement for at least half of the tasks.

CP-UC incurred a clear aggregate latency cost. Its mean makespans were 137.97, 124.64, and 122.61. On the paired percentage measure it was 2.10%, 3.39%, and 2.72% worse than FIFO, with fully negative bootstrap intervals. The simple ratio of means gives penalties of 1.96%, 2.90%, and 2.46%; the difference occurs because the reported paired measure weights tasks equally before averaging. Relative to CriticalPath, CP-UC added 3.42, 4.23, and 3.29 work units. The cost bought lower communication. As Table 6 shows, CP-UC reduced mean communication work from FIFO's 10.04, 13.39, and 14.02 to 9.31, 12.29, and 12.72 - a reduction of 7.27%, 8.22%, and 9.27%. Its handoff-rate reductions were 1.83, 2.09, and 1.73 percentage points. Relative to CriticalPath, reductions in communication work were 11.33%, 7.59%, and 7.83%. The primary 0.85 edge budget had zero violations in all 300 CP-UC schedules.

Table 6 Cross-Worker Dependency Handoffs And Normalized Communication Work.

Policy	H% K2	H% K4	H% K8	Comm K2	Comm K4	Comm K8
FIFO	25.99	34.33	35.82	10.04	13.39	14.02

Policy	H% K2	H% K4	H% K8	Comm K2	Comm K4	Comm K8
Round-robin	62.28	90.42	98.59	22.82	33.66	37.75
Static-role	22.83	30.27	30.43	9.41	11.61	11.78
CP	28.54	34.58	35.63	10.50	13.30	13.80
U-CP	28.27	34.32	35.71	10.64	13.30	13.84
CP-UC	24.16	32.24	34.09	9.31	12.29	12.72

Note. H% is the percentage of DAG edges crossing workers. Communication work uses ceil(0.15 times predecessor work) per crossing.

Figure 6 places these values on a communication-makespan plane. CP-UC shifts left from FIFO and CriticalPath but upward in latency. Round-Robin is dominated: it combines mean makespans of 152.36, 135.66, and 131.84 with 22.82-37.75 communication work. StaticRole has low communication at (K=2), yet its severe load imbalance produces the longest makespan. No single policy minimizes every quantity.

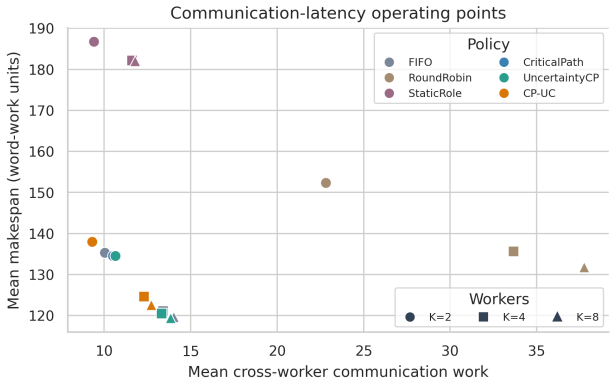

Figure 6 Communication-latency operating points for each policy and worker limit.

Figure 7 confirms that handoffs and load balance are separate mechanisms. Round-Robin nearly saturates cross-worker edges at (K=8), reaching a 98.59% handoff rate, even though cyclic placement gives it the lowest mean load coefficient of variation. StaticRole lowers the handoff rate to 30.43% but concentrates roles and work; at eight workers, its mean load coefficient of variation reaches 2.30. CP-UC is slightly less balanced than FIFO because it deliberately preserves locality. Thus even lexical load balance can coexist with poor completion time when dependencies repeatedly cross workers.

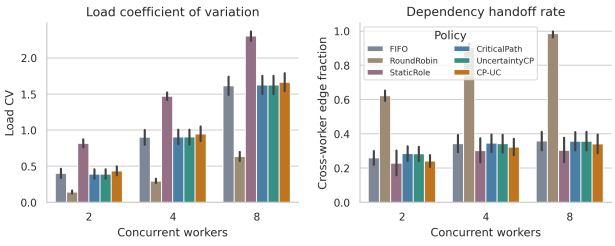

Figure 7 Load Imbalance And Dependency Handoff Rate Across Scheduling Policies.

Uncertainty ordering, inference, and sensitivity

Table 7 reports the risk-ordering metrics. CP-UC produced high-uncertainty completion values of 0.6281, 0.6281, and 0.6261, lower than FIFO by 0.0218, 0.0161, and 0.0108. Its propagation exposure was 0.4862, 0.4805, and 0.4757, representing relative reductions from FIFO of 2.15%, 2.26%, and 1.66%. Against CriticalPath, exposure fell by 3.71%, 3.20%, and 2.07%. Thus the proposed policy does prioritize consequential uncertain work earlier in normalized time, even though locality penalties lengthen the schedule.

Table 7 Uncertainty Timing And Structural Failure-Propagation Exposure.

Policy	K	High-U completion	Exposure	P95 exposure
CP-UC	2	0.6281	0.4862	0.5831
CP-UC	4	0.6281	0.4805	0.6127
CP-UC	8	0.6261	0.4757	0.5752
CP	2	0.6625	0.5049	0.6164
CP	4	0.6516	0.4964	0.6128
CP	8	0.6401	0.4857	0.5792
FIFO	2	0.6499	0.4969	0.5965
FIFO	4	0.6442	0.4916	0.6127
FIFO	8	0.6369	0.4837	0.5792
Round-robin	2	0.6518	0.4993	0.6142
Round-robin	4	0.6420	0.4836	0.6010
Round-robin	8	0.6272	0.4682	0.5646
Static-role	2	0.6250	0.4897	0.5823
Static-role	4	0.6279	0.4912	0.5836
Static-role	8	0.6275	0.4908	0.5836
U-CP	2	0.6391	0.4956	0.5830
U-CP	4	0.6453	0.4932	0.6127
U-CP	8	0.6382	0.4846	0.5752

Note. Lower values are better. These are schedule-structural exposure measures, not observed LLM failure rates.

Figure 8 shows that this advantage is modest and not unique under every capacity. At (K=8), Round-Robin achieved the lowest propagation exposure, 0.4682, but its makespan and communication were substantially worse. StaticRole had the lowest high-uncertainty completion at (K=2), 0.6250, while posting a makespan of 186.74. Risk metrics cannot substitute for latency or feasibility; they define an additional operating dimension.

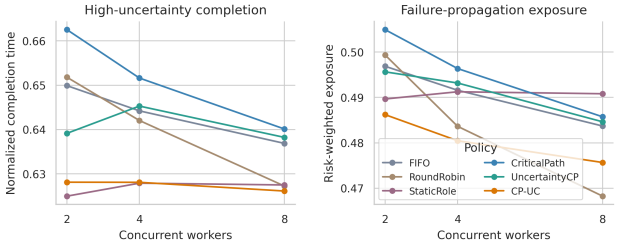

Figure 8 Normalized Completion Time For High-Uncertainty Requirements And Risk-Weighted Failure-Propagation Exposure.

The paired tests in Table 8 support a cautious interpretation. CP-UC's makespan difference from FIFO was not significant after Holm correction at K=2 (adjusted p=0.264) or K=8 (p=0.086); it was significant at K=4 (p=0.041). All CP-UC makespan differences from CriticalPath and UncertaintyCP were significant after correction ($p \leq 0.011$). Handoff-rate reductions against CriticalPath and

UncertaintyCP were significant for every team size ($p \leq 0.014$). Propagation-exposure reductions against these two schedulers were also significant ($p \leq 0.011$). Comparisons with FIFO were weaker: adjusted exposure p-values were 0.060, 0.041, and 0.060 for K=2, 4, and 8. Median paired differences were zero in all reported tests because most task-policy pairs tied, so significance arises from the magnitude and direction of changes among affected tasks rather than universal improvement.

Table 8 Paired Cp-Uc Contrasts With Holm-Adjusted Wilcoxon P-Values.

K	Baseline	Delta Cmax (pH)	Delta handoff (pH)	Delta exposure (pH)
2	CP	3.420 (0.005353)	-0.0438 (0.0001939)	-0.0187 (6.181e-07)
2	U-CP	3.400 (0.001923)	-0.0411 (0.0001206)	-0.0094 (0.005939)
4	CP	4.230 (0.0005237)	-0.0234 (0.002296)	-0.0159 (1.625e-05)
4	U-CP	4.120 (0.002296)	-0.0208 (0.002296)	-0.0127 (0.002296)
8	CP	3.290 (0.005353)	-0.0154 (0.01416)	-0.0101 (0.005353)
8	U-CP	3.170 (0.011)	-0.0162 (0.011)	-0.0090 (0.011)

Note. Differences are CP-UC minus baseline. Negative handoff and exposure differences favor CP-UC.

Sensitivity results in Table 9 and Figure 9 preserve the central trade-off. As the handoff-cost rate increased from 0 to 0.30, mean (K=8) makespan rose from 118.54 to 120.86 for CriticalPath, from 118.54 to 120.99 for UncertaintyCP, and from 121.42 to 123.90 for CP-UC. CP-UC remained slower throughout, although its mean handoff count stayed nearly fixed. Tightening the budget ratio exposed a limitation of greedy locality control. At ratio 0.65, summed violation amounts were 5 and 21 for (K=4,8); at 0.45 they reached 54 and 77. Mean handoffs changed only from 3.03 to 3.00 at (K=4) and 3.26 to 3.23 at (K=8). The 0.85 setting was the only tested ratio with zero violations at every capacity. Harder communication limits would require graph partitioning or backtracking rather than a local worker choice.

Table 9 CP-UC Sensitivity To The Cross-Worker Edge Budget.

K	Baseline	Delta Cmax (pH)	Delta handoff (pH)	Delta exposure (pH)
2	CP	3.420 (0.005353)	-0.0438 (0.0001939)	-0.0187 (6.181e-07)
2	U-CP	3.400 (0.001923)	-0.0411 (0.0001206)	-0.0094 (0.005939)
4	CP	4.230 (0.0005237)	-0.0234 (0.002296)	-0.0159 (1.625e-05)
4	U-CP	4.120 (0.002296)	-0.0208 (0.002296)	-0.0127 (0.002296)
8	CP	3.290 (0.005353)	-0.0154 (0.01416)	-0.0101 (0.005353)
8	U-CP	3.170 (0.011)	-0.0162 (0.011)	-0.0090 (0.011)

Note. The main protocol uses 0.85; stricter caps expose unavoidable joins created by earlier parallel assignments.

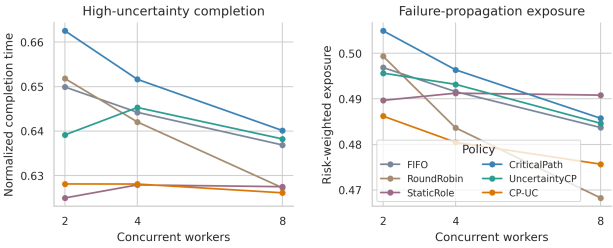

Figure 9 Sensitivity Of CP, U-CP, And CP-UC Makespan To Normalized Cross-Worker Communication Cost.

Table 10 documents coverage across 40 application topics. Travel had the largest mean edge count (19.0) among three-task topics; Shopping and Security followed at 15.67 and 14.33. Shooter Game had the greatest mean work (272.0), while Puzzle Game had only 1.5 mean edges. Since each topic contains two or three tasks, these rows establish breadth rather than support topic-level significance tests.

Table 10 Coverage Of The 40 Coding Application Topics.

Topic	Tasks	Req.	Mean work	Mean edges
Action Game	3	22	163.00	11.00
Budgeting	3	21	192.67	4.00
Culture	3	15	182.33	7.67
Development	3	17	242.33	8.67
Family Kids	3	18	242.33	6.67
Graphics	3	15	223.33	4.67
Language	3	20	170.33	10.33
Medical	3	18	205.00	5.67
News	3	20	174.33	7.00
Office	3	22	155.00	10.33
Photo	3	20	221.00	5.00
Racing Game	3	18	208.00	5.67
Restaurants Delivery	3	24	182.33	10.33
Schedule	3	20	166.00	4.33
Security	3	23	171.67	14.33
Shopping	3	24	194.67	15.67
SocialNetwork	3	20	215.33	8.33
Sports	3	14	162.00	3.00
Tools Utilities	3	18	212.33	6.67
Travel	3	24	249.67	19.00
Board Game	2	12	164.50	4.50
Business	2	12	185.50	7.00
Data	2	12	195.50	5.00
Entertainment	2	10	186.00	6.00
Finance	2	10	157.50	4.50
Health Fitness	2	11	164.00	7.50
Management Game	2	12	226.00	9.00
Music	2	13	198.00	5.50
Notebook	2	16	188.00	14.00
Personalisation	2	15	204.50	12.50
Puzzle Game	2	12	143.50	1.50
Reference Books	2	7	175.50	3.50
Role Playing Game	2	11	173.00	6.00
Science	2	8	149.50	6.00
Shooter Game	2	12	272.00	11.50
Simulation Game	2	10	163.50	6.50
Sport Game	2	12	190.00	8.00
Strategy Game	2	8	146.00	4.00
Transportation	2	13	156.00	8.00
Video	2	12	150.50	7.00

Note. Each topic contains two or three tasks; topic rows describe coverage rather than inferential strata.

The representative CP-UC trace in Figure 10 is task 50, a five-node Personalisation test-case graph with 235 work units and four edges. Three developer nodes ran in parallel from time 0; a 129-unit tester node began at 32 after three predecessors, and the final 23-unit developer node ended at 184. All earliest-finish policies produced the same task-50 schedule at (K=4), whereas Round-Robin and StaticRole ended at 207 and 259. The trace makes the structural result concrete: once the 129-unit validation step lies on a 182-unit critical path, four workers cannot shorten it through scheduling alone.

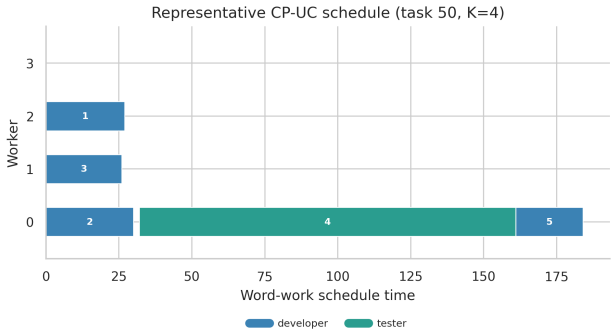

Figure 10 Representative CP-UC schedule for task 50 with four available workers.

Operational Interpretation And Validity Boundaries

The combined results suggest a three-part scheduler-selection rule. First, use CriticalPath when the service objective is normalized completion time and the graph is known. It never regressed against FIFO at four or eight workers and reached the analytical lower bound on 70 and 73 of 100 tasks, compared with 67 and 68 for FIFO. Second, add uncertainty ranking when earlier completion of risky requirements is valuable, but do not expect a latency gain: UncertaintyCP's aggregate exposure was lower than CriticalPath only at two workers, and its makespan was slightly higher at every capacity. Third, use CP-UC only when reducing handoffs or exposure has an explicit value. Its primary configuration lowered both, but 17 tasks at four workers and 12 at eight workers were slower than FIFO, while only 5 and 4 were faster.

The capacity results are also actionable. At two workers, the critical path dominated the work-per-worker lower bound on 60 tasks. At four workers this count rose to 98, and at eight it reached all 100 tasks. Once every task is critical-path dominated, adding workers cannot reduce the lower bound; only graph transformation, faster nodes, or removal of precedence can do so. This explains why mean makespan changed little from four to eight workers even though nominal capacity doubled. In an LLM system, the analogous intervention would be to merge unnecessary stages, begin validation earlier, split an oversized requirement, or change the artifact interface so that more work becomes independent. In a live service, this structural diagnosis would be paired with tenant and token-demand models [33], [34], service placement [36], and admission control [37] before deciding whether to add capacity or transform the DAG.

The communication findings do not imply that fewer messages always improve reliability. CP-UC penalizes cross-worker dependency transfers, not all dialogue, and propagation exposure is computed from completion order rather than observed errors. A handoff may carry beneficial review information, especially on test-case tasks. The result is narrower: under a fixed 15% transfer cost, preserving locality reduces modeled communication work and brings high-impact uncertain nodes earlier in normalized time, while delaying some parallel fan-in. A deployment should price these outcomes using measured context size, retry rate, and quality loss rather than adopt the 0.85 budget universally. Operational RAG and incident systems [39], [45]-[50] and safety-control interfaces [57]-[61] likewise imply that handoff volume, evidence quality, and escalation state should be recorded separately.

Several limitations bound the conclusions. Word-work units ignore provider latency, prompt length, caching, tool execution, and heterogeneous model speed. Lexical cues provide an auditable uncertainty ordering but are not probabilities of failure. Forward-edge extraction favors dependencies expressed in the requirement order and can miss implicit feedback loops. Finally, the experiment evaluates scheduling transformations over task specifications, not generated programs, so it cannot connect faster schedules to software correctness. A live extension should record per-agent service times and outcomes, fit duration and failure models on a training split, and retain the same DAG audit to test whether the small critical-path gains and CP-UC trade-off persist. A live extension could calibrate duration and failure estimates with selective and model-risk workflows [67]-[75], evaluate scheduler updates with offline-policy controls [76]-[83], and manage accumulated state through confidence-gated memory and privacy-aware preference learning [84], [85].

D. CONCLUSIONS

This study evaluated resource-constrained scheduling on all 100 released MultiAgentBench/MARBLE coding tasks. The requirement-level transformation produced 621 nodes and 780 evidence-based edges, and the full experiment generated 1,800 schedules across six policies and three team sizes. CriticalPath was the strongest latency default, but its improvement over FIFO remained below 1% on average and most tasks tied. Increasing capacity from two to eight workers yielded only 1.225 mean task-level speedup for CriticalPath and reduced utilization to 23.59%, reflecting limited DAG width rather than scheduler failure. CP-UC achieved its intended coordination effects: it reduced communication work by 7.27%-9.27% and propagation exposure by 1.66%-2.26% relative to FIFO, with no budget violations at the primary ratio. Those gains cost 2.10%-3.39% in paired mean makespan, especially on test-case graphs where locality competed with parallel fan-in. The practical conclusion is therefore conditional. Use critical-path priority when completion time dominates; use uncertainty and locality penalties when cross-agent transfers or unresolved high-impact work carry operational cost, and expose the resulting latency trade-off to system

designers. Stronger communication constraints should employ global partitioning or rescheduling, and live deployment claims require measured agent runtimes and task outcomes.

E. DAFTAR PUSTAKA

1. K. Zhu, H. Du, Z. Hong, X. Yang, S. Guo, Z. Wang, Z. Wang, C. Qian, X. Tang, H. Ji, and J. You, "MultiAgentBench: Evaluating the collaboration and competition of LLM agents," in Proc. 63rd Annu. Meeting Assoc. Comput. Linguistics (ACL), vol. 1, Vienna, Austria, 2025, pp. 8580-8622, doi: 10.18653/v1/2025.acl-long.421.
2. Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," in Proc. 1st Conf. Language Modeling (COLM), 2024. [Online]. Available: https://openreview.net/forum?id=tEAF9LBdgu
3. G. Li, H. A. A. K. Hammoud, H. Itani, D. Khizbullin, and B. Ghanem, "CAMEL: Communicative agents for 'mind' exploration of large language model society," in Adv. Neural Inf. Process. Syst., vol. 36, 2023, pp. 51991-52008. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/a3621ee907def47c1b952ade25c67698-Abstract-Conference.html
4. W. Chen et al., "AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors," in Proc. 12th Int. Conf. Learning Representations (ICLR), 2024. [Online]. Available: https://openreview.net/forum?id=EHg5GDnyq1
5. S. Hong et al., "MetaGPT: Meta programming for a multi-agent collaborative framework," in Proc. 12th Int. Conf. Learning Representations (ICLR), 2024. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html
6. C. Qian et al., "ChatDev: Communicative agents for software development," in Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics (ACL), vol. 1, Bangkok, Thailand, 2024, pp. 15174-15186, doi: 10.18653/v1/2024.acl-long.810.
7. M. Zhuge, W. Wang, L. Kirsch, F. Faccio, D. Khizbullin, and J. Schmidhuber, "GPTSwarm: Language agents as optimizable graphs," in Proc. 41st Int. Conf. Machine Learning (ICML), PMLR, vol. 235, 2024, pp. 62743-62767. [Online]. Available: https://proceedings.mlr.press/v235/zhuge24a.html
8. Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving factuality and reasoning in language models through multiagent debate," in Proc. 41st Int. Conf. Machine Learning (ICML), PMLR, vol. 235, 2024, pp. 11733-11763. [Online]. Available: https://proceedings.mlr.press/v235/du24e.html
9. Z. Liu, Y. Zhang, P. Li, Y. Liu, and D. Yang, "A dynamic LLM-powered agent network for task-oriented agent collaboration," in Proc. 1st Conf. Language Modeling (COLM), 2024. [Online]. Available: https://openreview.net/forum?id=i43XCU54Br
10. J. Li, Q. Zhang, Y. Yu, Q. Fu, and D. Ye, "More agents is all you need," Trans. Mach. Learn. Res., 2024. [Online]. Available: https://openreview.net/forum?id=bgzUSZ8aeg
11. J. Wang, J. Wang, B. Athiwaratkun, C. Zhang, and J. Zou, "Mixture-of-agents enhances large language model capabilities," in Proc. 13th Int. Conf. Learning Representations (ICLR), 2025. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/hash/5434be94e82c54327bb9dcaf7fca52b6-Abstract-Conference.html
12. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proc. 11th Int. Conf. Learning Representations (ICLR), 2023. [Online]. Available: https://openreview.net/forum?id=WE_vluYUL-X
13. M. Besta et al., "Graph of thoughts: Solving elaborate problems with large language models," Proc. AAAI Conf. Artif. Intell., vol. 38, no. 16, pp. 17682-17690, 2024, doi: 10.1609/aaai.v38i16.29720.
14. S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," in Adv. Neural Inf. Process. Syst., vol. 36, 2023, pp. 11809-11822. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html
15. S. Kim, S. Moon, R. Tabrizi, N. Lee, M. W. Mahoney, K. Keutzer, and A. Gholami, "An LLM compiler for parallel function calling," in Proc. 41st Int. Conf. Machine Learning (ICML), PMLR, vol. 235, 2024, pp. 24370-24391. [Online]. Available: https://proceedings.mlr.press/v235/kim24y.html
16. H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260-274, Mar. 2002, doi: 10.1109/71.993206.
17. H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 3, pp. 682-694, Mar. 2014, doi: 10.1109/TPDS.2013.57.
18. Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," ACM Comput. Surv., vol. 31, no. 4, pp. 406-471, Dec. 1999, doi: 10.1145/344588.344618.

19. R. L. Graham, "Bounds for certain multiprocessing anomalies," Bell Syst. Tech. J., vol. 45, no. 9, pp. 1563-1581, Nov. 1966, doi: 10.1002/j.1538-7305.1966.tb01709.x.
20. J. E. Kelley, Jr. and M. R. Walker, "Critical-path planning and scheduling," in Proc. Eastern Joint IRE-AIEE-ACM Comput. Conf., Boston, MA, USA, 1959, pp. 160-173, doi: 10.1145/1460299.1460318.
21. M. Xiong, Z. Hu, X. Lu, Y. Li, J. Fu, J. He, and B. Hooi, "Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs," in Proc. 12th Int. Conf. Learning Representations (ICLR), 2024. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/hash/6733cf15e10e2cd1d59af033c3bb8507-Abstract-Conference.html
22. S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting hallucinations in large language models using semantic entropy," Nature, vol. 630, no. 8017, pp. 625-630, Jun. 2024, doi: 10.1038/s41586-024-07421-0.
23. G. Zhang, Y. Yue, Z. Li, S. Yun, G. Wan, K. Wang, D. Cheng, J. Xu Yu, and T. Chen, "Cut the crap: An economical communication pipeline for LLM-based multi-agent systems," in Proc. 13th Int. Conf. Learning Representations (ICLR), 2025. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/hash/bbc461518c59a2a8d64e70e2c38c4a0e-Abstract-Conference.html
24. J. Zhang et al., "AFlow: Automating agentic workflow generation," in Proc. 13th Int. Conf. Learning Representations (ICLR), 2025. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html
25. S. Qiao, R. Fang, Z. Qiu, X. Wang, N. Zhang, Y. Jiang, P. Xie, F. Huang, and H. Chen, "Benchmarking agentic workflow generation," in Proc. 13th Int. Conf. Learning Representations (ICLR), 2025. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/hash/adbe936993aa7cf41e45054d8b72f183-Abstract-Conference.html
26. W. Chen, J. Yuan, C. Qian, C. Yang, Z. Liu, and M. Sun, "Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system," in Findings Assoc. Comput. Linguistics: ACL 2025, Vienna, Austria, 2025, pp. 11534-11557, doi: 10.18653/v1/2025.findings-acl.601.
27. ULab-UIUC, "MARBLE: Multi-Agent CooRdination Backbone with LLM Engine," GitHub repository, 2025. [Online]. Available: https://github.com/ulab-uiuc/MARBLE (accessed Aug. 5, 2026).
28. Shilu He, Haowei Tu, and Isa Liu, "Safe PD Capacity Forecasting with Time-Series Foundation Models and Calibrated Uncertainty for Heterogeneous GPU Clusters," J. Adv. Comput. Syst., vol. 3, no. 4, pp. 48-66, Apr. 2023, doi: 10.69987/JACS.2023.30404.
29. C. Wang, Z. Wen, R. Zhang, P. Xu, and Y. Jiang, "GPU Memory Requirement Prediction for Deep Learning Task Based on Bidirectional Gated Recurrent Unit Optimization Transformer," in Proc. 5th Int. Conf. Artificial Intelligence, Virtual Reality and Visualization (AIVRV), Chengdu, China, 2025, pp. 31-35, doi: 10.1109/AIVRV67401.2025.11350369.
30. Shilu He, Xiaohan Chang, and Eddin Sun, "Cross-Cloud Transfer Learning for AI Training Capacity Forecasting under Workload and Topology Distribution Shift," J. Adv. Comput. Syst., vol. 4, no. 1, pp. 100-120, Jan. 2024, doi: 10.69987/JACS.2024.40108.
31. S. Zhao, J. Bai, and D. Roberson, "Multi-Horizon GPU Demand Forecasting with Workload Semantics and Operational Risk Curves: An Empirical Study on Alibaba Clusterdata GPU Trace," J. Technol. Informatics Eng., vol. 4, no. 3, pp. 544-571, Dec. 2025, doi: 10.51903/jtie.v4i3.498.
32. Siyu Chen, Shilu He, and Eddin Sun, "Risk-Bounded GPU Resource Oversubscription via Conformal Demand Envelopes in Production AI Clusters," J. Adv. Comput. Syst., vol. 4, no. 5, pp. 119-134, May 2024, doi: 10.69987/JACS.2024.40509.
33. S. He, C. Li, and H. Rao, "Few-Shot Cold-Start Workload Forecasting for New AI Inference Tenants with Time-Series Foundation Models," J. Technol. Informatics Eng., vol. 4, no. 1, pp. 306-324, Apr. 2025, doi: 10.51903/jtie.v4i1.546.
34. J. Nie, Y. Shi, and L. Zhao, "Token-Burst-Aware Capacity Planning for LLM Inference Services: Request Arrival, Token Demand, and Failure Risk Modeling from BurstGPT Traces," J. Technol. Informatics Eng., vol. 5, no. 2, pp. 142-164, Aug. 2026, doi: 10.51903/jtie.v5i2.565.
35. S. He, J. Nie, and C. Li, "Power-Aware Inventory Planning for AI Infrastructure Using Job-Level Forecasting and LLM Workload Explanations," J. Technol. Informatics Eng., vol. 5, no. 1, pp. 341-359, Apr. 2026, doi: 10.51903/jtie.v5i1.548.
36. Q. Xin, "Hybrid Cloud Architecture for Efficient and Cost-Effective Large Language Model Deployment," J. Inf. Syst. Informatics, vol. 7, no. 3, pp. 2182-2195, Sep. 2025, doi: 10.51519/journalisi.v7i3.1170.
37. S. Zhao, Y. Ren, and X. Chang, "Profit-Aware Spot GPU Admission Control with Cost-Sensitive Loss and Evidence-Grounded Policy Memos for AI Workload Supply-Demand Matching," J. Technol. Informatics Eng., vol. 5, no. 2, pp. 45-59, Aug. 2026, doi: 10.51903/jtie.v5i2.545.
38. Ge Liu, Shilu He, and Isa Liu, "LLM-Augmented Multi-Source Root Cause Attribution for CPU and Network Faults in Microservices," J. Adv. Comput.

Syst., vol. 3, no. 6, pp. 39-57, Jun. 2023, doi: 10.69987/JACS.2023.30604.
39. Boning Zhang, Hengning Rao, and Derek Zhao, "Evidence-Grounded RAG for Cloud-Native DevOps: Hallucination-Resistant AIOps Question Answering over Private Operations Documents," J. Adv. Comput. Syst., vol. 4, no. 3, pp. 109-125, Mar. 2024, doi: 10.69987/JACS.2024.40308.
40. Ge Liu, Chenyu Li, and Eric Zhang, "OpsLLM for Cloud Incident Triage: Bilingual RAG-Based Root Cause Analysis and Alert Summarization for AI Infrastructure Operations," J. Adv. Comput. Syst., vol. 4, no. 4, pp. 97-111, Apr. 2024, doi: 10.69987/JACS.2024.40408.
41. Q. Xin, "Explaining OpenStack Failure-Injection Log Anomalies with Retrieved Normal Prototypes," Emerg. Inf. Sci. Technol., vol. 6, no. 2, pp. 125-146, Nov. 2025, doi: 10.18196/eist.v6i2.31232.
42. Yunhe Li, "Findable then Explainable: Retrieval-Summary Integration for Code Intelligence on a Lightweight CodeSearchNet Subset," J. Adv. Comput. Syst., vol. 4, no. 7, pp. 65-82, Jul. 2024, doi: 10.69987/JACS.2024.40706.
43. Q. Xin, "Self-Supervised Log Anomaly Detection with LogBERT-Style Transformers: Full Empirical Evaluation on a Reproducible SynHDFS Benchmark," J. Electr. Eng. Comput. Sci., vol. 11, no. 1, pp. 23-35, May 2026, doi: 10.54732/jeecs.v11i1.3.
44. X. Sun, Z. S. Zhong, and Q. Wu, "Retrieval-Grounded HDFS Log Anomaly Detection and Deterministic Failure Narrative Generation," J. Comput. Syst. Appl., vol. 3, no. 1, pp. 15-30, Jun. 2026, doi: 10.64229/j6d7fr94.
45. Q. Xin, "Log Anomaly Detection with Conformal Alert Control and Evidence-Grounded Incident Ticket Generation," AVITEC, vol. 8, no. 2, pp. 247-264, Aug. 2026, doi: 10.28989/avitec.v8i2.3974.
46. B. Zhang, X. Sun, G. Liu, and B. Zhou, "LLM-Style DevOps Copilot for Cloud-Native Troubleshooting: Retrieval-Augmented Runbook Generation and Command-Safety Evaluation," J. Technol. Informatics Eng., vol. 5, no. 2, pp. 104-118, Aug. 2026, doi: 10.51903/jtie.v5i2.534.
47. C. Li, G. Liu, and Z. Zhao, "Cost-Aware LLM-Style Routing for AIOps Log Analysis: Log Parsing, Anomaly Detection, Fault Diagnosis, and Incident Summarization on LogEval Task Files," J. Technol. Informatics Eng., vol. 5, no. 2, pp. 91-103, Jun. 2026, doi: 10.51903/jtie.v5i2.538.
48. Z. S. Zhong, C. Li, and H. Rao, "Trajectory Reliability Prediction for Generalist AI Agents: Tool-Use Failure Analysis and Success Forecasting on ZClawBench," J. Technol. Informatics Eng., vol. 5, no. 1, pp. 341-360, Apr. 2026, doi: 10.51903/jtie.v5i1.539.
49. Jiayi Nie and David Zheng, "Noisy-Neighbor-Aware VM Degradation Risk Modeling with Unsupervised Residual Fusion," J. Adv. Comput. Syst., vol. 4, no. 4, pp. 112-123, Apr. 2024, doi: 10.69987/JACS.2024.40409.
50. J. Nie, G. Liu, C. Li, and T. Zou, "Evidence-Constrained Incident Visualization Cards for Distributed Cloud Logs: A UI/UX Framework for Turning Hadoop, OpenStack, and ZooKeeper Logs into Actionable SRE Design Interfaces," Int. J. Graph. Des., vol. 4, no. 1, pp. 179-185, Apr. 2026, doi: 10.51903/ijgd.v4i1.3703.
51. G. Liu, S. He, and H. Wong, "LLM-Compatible Visual Brief Cards for AI Infrastructure Capacity Dashboards: A UI/UX Framework for Turning Forecast Risk into Graphic Design Decisions," Int. J. Graph. Des., vol. 3, no. 1, pp. 196-213, May 2025, doi: 10.51903/ijgd.v3i1.3723.
52. Chenyu Li, Jingwen Bai, and Samuel Wang, "Evidence-Chain Reliable RAG: Word-Level Hallucination Detection, Source Attribution, and Provenance Explanation for LLM Applications," J. Adv. Comput. Syst., vol. 4, no. 2, pp. 76-92, Feb. 2024, doi: 10.69987/JACS.2024.40207.
53. Z. S. Zhong, J. Chen, E. Zhong, and X. Sun, "Evidence-Calibrated RAG for Unanswerable Question Answering: Retrieval Coverage, Abstention Calibration, and Hallucination-Proxy Analysis on SQuAD 2.0," J. Technol. Informatics Eng., vol. 4, no. 2, pp. 502-520, Aug. 2025, doi: 10.51903/jtie.v4i2.536.
54. J. Jin, "Evidence-Chain Reliable RAG: Hallucination Detection, Source Attribution, and Deterministic Provenance Explanations," J. Technol. Informatics Eng., vol. 4, no. 2, pp. 520-533, Aug. 2025, doi: 10.51903/jtie.v4i2.535.
55. W. Su, S. Chen, and C. Zhao, "Budgeted Multi-Hop Retrieval Agent for Compositional Question Answering: A Retrieval-Policy Evaluation on the Official MultiHop-RAG Benchmark," J. Technol. Informatics Eng., vol. 4, no. 3, pp. 649-662, Dec. 2025, doi: 10.51903/jtie.v4i3.543.
56. R. Zhang, Z. Wen, C. Wang, C. Tang, P. Xu, and Y. Jiang, "Quality Analysis and Evaluation Prediction of RAG Retrieval Based on Machine Learning Algorithms," arXiv preprint arXiv:2511.19481, 2025, doi: 10.48550/arXiv.2511.19481.
57. Daren Zheng, Chenyu Li, and Harvey Davidson, "Continual Red-Teaming for In-the-Wild Jailbreaks via Online Guardrail Updates and Guardrail Distillation," J. Adv. Comput. Syst., vol. 3, no. 2, pp. 35-49, Feb. 2023, doi: 10.69987/JACS.2023.30203.
58. W. Su, H. Rao, and E. Ma, "Privacy and Data-Integrity Risk Cards for LLM Agents: A UI/UX Design Framework for Secure Human Oversight under Prompt-Injection Attacks," Int. J. Graph. Des., vol. 4, no. 1, pp. 186-191, Apr. 2026, doi: 10.51903/ijgd.v4i1.3699.
59. Daren Zheng, Boning Zhang, and Julie Geibel, "VerifySafe: Toxicity-Safe Agent Responses under Adversarial Prompts with Evidence-Based Self-Verification," J. Adv. Comput. Syst., vol. 4, no. 1,

pp. 67-82, Jan. 2024, doi: 10.69987/JACS.2024.40106.
60. B. Zhou, C. Li, and L. Liu, "Risk-Calibrated Patient-Facing AI Safety Cards: A UI/UX Design Framework for Rubric-Based Medical Risk Communication," Int. J. Graph. Des., vol. 3, no. 2, pp. 365-380, Oct. 2025, doi: 10.51903/ijgd.v3i2.3696.
61. Daren Zheng and Chenyu Li, "Behavior-Level Jailbreak Resistance via Multi-Stage Refusal + Utility Preservation," J. Adv. Comput. Syst., vol. 4, no. 1, pp. 83-99, Jan. 2024, doi: 10.69987/JACS.2024.40107.
62. Q. Xin, "Host-Based Intrusion Detection with System Call Sequences: Window Localization and Forensic Narratives," AVITEC, vol. 8, no. 2, pp. 325-334, Aug. 2026, doi: 10.28989/avitec.v8i2.3973.
63. J. Bai, S. Chen, D. Zheng, and M.-J. Kuo, "Interpretable Attack-Chain Stage Detection from AWS CloudTrail Event Sequences via Linear Models and HMM Smoothing," Inf. Electr. Electron. Eng., vol. 6, no. 1, pp. 28-43, May 2026, doi: 10.33474/infotron.v6i1.24923.
64. Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, "IoT Traffic Classification and Anomaly Detection Method Based on Deep Autoencoders," Appl. Comput. Eng., vol. 69, no. 1, pp. 64-70, 2024, doi: 10.54254/2755-2721/69/20241511.
65. Y. Li and S. Lu, "Language-Guided Feature Selection for DDoS and Intrusion Detection on CICIDS2017," J. Technol. Informatics Eng., vol. 4, no. 1, pp. 284-305, Apr. 2025, doi: 10.51903/jtie.v4i1.531.
66. Shenghan Lu and David Zhou, "TinyLLM-Assisted Intrusion Detection for Real-Time IoT Networks," J. Adv. Comput. Syst., vol. 4, no. 8, pp. 72-87, Aug. 2024, doi: 10.69987/JACS.2024.40809.
67. Q. Xin, "Uncertainty-Aware Late Fusion for 3D Perception (Confidence Calibration + Fusion Rule Learning)," J. Technol. Informatics Eng., vol. 4, no. 1, pp. 215-238, Apr. 2025, doi: 10.51903/jtie.v4i1.485.
68. Yuanzheng Chen, Yitian Zhang, David Chau, and Matt Sherman, "Credit Card Default Risk Tiering with Probability Calibration and Uncertainty-Driven Rejection: A Reproducible Study on the UCI Credit Card Clients Dataset," J. Adv. Comput. Syst., vol. 3, no. 4, pp. 31-47, Apr. 2023, doi: 10.69987/JACS.2023.30403.
69. Jiaying Jin, Tina Huang, and Sam Lu, "A Model-Risk-Friendly Probability of Default Workflow: Calibration, Distribution-Free Uncertainty Quantification, and SHAP Explanations on the UCI Credit Card Default Dataset," J. Adv. Comput. Syst., vol. 4, no. 6, pp. 74-85, Jun. 2024, doi: 10.69987/JACS.2024.40606.
70. Q. Xin, "Explainable and Fair Credit Risk Scoring with Counterfactual Explanations: A Reproducible Evaluation on the German Credit Dataset (HELOC-Motivated)," J. Inf. Technol., vol. 14, no. 2, pp. 215-231, Jun. 2026, doi: 10.32664/j-intech.v14i02.2228.
71. Jiaying Jin, Tina Huang, and Sam Lu, "Cost-Sensitive Learning, Simulated PU Learning, and One-Class Autoencoding for Extreme-Imbalance Credit Card Fraud Detection," J. Adv. Comput. Syst., vol. 4, no. 6, pp. 64-73, Jun. 2024, doi: 10.69987/JACS.2024.40605.
72. H. Wang, Y. Ren, and X. Chang, "Layout-Aware Progressive PDF Rendering: AI Prioritization of PDF Slices to Reduce Time-to-Functional-First-Frame on FUNSD," J. Technol. Informatics Eng., vol. 4, no. 2, pp. 425-446, Aug. 2025, doi: 10.51903/jtie.v4i2.523.
73. B. Zhou, J. Jin, and D. Zhao, "Calibrated Resume-Job Matching for Trustworthy LLM-Assisted Recruiter Screening: Pairwise Matching, Probability Calibration, and Selective Refusal on Two Public Recruitment Datasets," J. Technol. Informatics Eng., vol. 4, no. 3, pp. 625-648, Dec. 2025, doi: 10.51903/jtie.v4i3.529.
74. Z. S. Zhong and S. Ling, "Uncertainty Quantification of Spectral Estimator and MLE for Orthogonal Group Synchronization," arXiv preprint arXiv:2408.05944, Aug. 2024, doi: 10.48550/arXiv.2408.05944.
75. J. Jin, "LLM-Style Evidence Cards for Scientific Search Interfaces: A UI/UX Design Framework for Retrieval Transparency, Ranking Trust, and Visual Evidence Hierarchy," Int. J. Graph. Des., vol. 3, no. 2, pp. 397-414, Oct. 2025, doi: 10.51903/ijgd.v3i2.3698.
76. J. Bai, H. Wang, Q. Wu, and B. Zhang, "Privacy-Robust Incrementality Estimation in Cookieless Settings via Uplift Modeling: Reproducible Evidence from the Hillstrom E-Mail Experiment," J. Technol. Informatics Eng., vol. 5, no. 1, pp. 17-38, Apr. 2026, doi: 10.51903/jtie.v5i1.468.
77. J. Mu, T. Ye, and P. Patel, "Offline Counterfactual Evaluation for Advertising and Recommendation Slot Policies: A Reproducible Study on the Open Bandit Dataset (Small)," J. Technol. Informatics Eng., vol. 4, no. 3, pp. 521-543, Dec. 2025, doi: 10.51903/jtie.v4i3.500.
78. T. Ye, J. Mu, and J. Hunter, "Off-Policy Evaluation and Conservative Policy Selection for Slot-Level Dynamic Bidding and Ranking on the Open Bandit Dataset (Small)," J. Technol. Informatics Eng., vol. 5, no. 1, pp. 178-199, Apr. 2026, doi: 10.51903/jtie.v5i1.503.
79. H. Zhang, "Risk-Aware Budget-Constrained Auto-Bidding under First-Price RTB: A Distributional Constrained Deep Reinforcement Learning Framework," J. Adv. Comput. Syst., vol. 4, no. 6, pp. 30-47, Jun. 2024.
80. H. Weng, H. Wang, and C. Wei, "Adaptive Bidding Strategies for Hybrid Auction Mechanisms in Programmatic Advertising," J. Adv. Comput. Syst., vol. 4, no. 4, pp. 13-25, Apr. 2024, doi: 10.69987/JACS.2024.40402.

81. H. Zhang, "DriftGuard: Multi-Signal Drift Early Warning and Safe Re-Training/Rollback for CTR/CVR Models," J. Adv. Comput. Syst., vol. 3, no. 7, pp. 24-40, Jul. 2023.
82. Jinyi Mu, Yifei Lu, and Michelle Smith, "LLM-Assisted Incrementality (Uplift) Modeling for Email Advertising: From Feature Interactions to Interpretable Audience-Creative-Channel Policies," J. Adv. Comput. Syst., vol. 3, no. 1, pp. 31-48, Jan. 2023, doi: 10.69987/JACS.2023.30103.
83. H. Zhang, "Counterfactual Learning-to-Rank for Ads: Off-Policy Evaluation on the Open Bandit Dataset," J. Adv. Comput. Syst., vol. 5, no. 12, pp. 1-11, Dec. 2025.
84. Xinzhuo Sun, Yifei Lu, and Jing Chen, "Controllable Long-Term User Memory for Multi-Session Dialogue: Confidence-Gated Writing, Time-Aware Retrieval-Augmented Generation, and Update/Forgetting," J. Adv. Comput. Syst., vol. 3, no. 8, pp. 9-24, Aug. 2023, doi: 10.69987/JACS.2023.30802.
85. M.-J. Kuo, D. Zheng, and J. Hires, "Federated Topic-Preference Learning for Knowledge-Grounded Chat with Differential Privacy," J. Technol. Informatics Eng., vol. 4, no. 2, pp. 385-401, Aug. 2025, doi: 10.51903/jtie.v4i2.502.