PT Jurnal Cendekia Indonesia
Journal of Information Technology and Informatics Engineering
Homepage: https://journal.jci.co.id/jitie
Vol. 02 No. 04 (2026) Page: 16-28
E-ISSN: 3123-7827

Constraint-Verified LLM Graph Workflow Synthesis with Execution-Guided Self-Repair, Parallel Optimization and Trajectory Failure Prediction

Liam Miller[1], Jason Peng[2], Ava Taylor[3], Melody Ding[4]

[1]Department of Computer Science, Case Western Reserve University, Cleveland, OH, USA
[2]Department of Computer Science, University of California, Santa Barbara, Santa Barbara, CA, USA
[3]Department of Computer Science, Tufts University, Medford, MA, USA
[4]Department of Computer Science, Georgetown University, Washington, DC, USA

Email: [1*]l.miller.cs@icloud.com

Abstract

Large language model agents increasingly express multi-step work as directed graphs, yet a plausible action list does not guarantee correct dependencies, structural executability, or useful parallelism. This paper presents CVR-Graph, a verification layer combining dependency scoring, typed priors, diagnostic-guided repair, transitive reduction, graph-quality failure prediction, and critical-path scheduling. Evaluation used 18,679 WorFBench training records and all 2,146 official test workflows from nine domains. After overlap removal, de-duplication, and structural filtering, 17,128 workflows remained for development. A topology-controlled protocol held reference task nodes fixed and measured edge construction directly. Linear and parallel-optimized methods achieved edge F1 0.875, path F1 0.963, and strict graph success 0.754. Reduction removed 3,836 redundant predicted edges and increased edge F1 by 0.083 without changing path F1. Validator feedback also restored structural executability for all 31 released graphs with incomplete START-to-END coverage. Gradient-boosted trajectory screening achieved ROC-AUC 0.944, PR-AUC 0.952, Brier score 0.102, and expected calibration error 0.051; accepting the lowest-risk 50% reduced failure risk by 68.7%. Gold-DAG scheduling produced mean speedup 1.369 with eight workers, whereas predicted graphs remained chains. The results show that verification and calibrated control are effective, while non-linear dependency induction remains the limiting factor under domain shift.

Keywords: LLM Agents; Workflow Synthesis; Directed Acyclic Graphs; Constrained Decoding; Structural Verification; Self-Repair; Failure Prediction; Confidence Calibration; Critical-Path Scheduling; Worfbench.

Abstrak

Agen model bahasa berskala besar (LLM) semakin sering merepresentasikan pekerjaan yang melibatkan banyak langkah sebagai graf berarah; namun, daftar tindakan yang masuk akal tidak menjamin adanya dependensi yang benar, kemampuan eksekusi secara struktural, ataupun paralelisme yang bermanfaat. Makalah ini menyajikan CVR-Graph, sebuah lapisan verifikasi yang mengombinasikan penilaian dependensi, prior berbasis tipe, perbaikan yang dipandu diagnostik, reduksi transitif, prediksi kegagalan berdasarkan kualitas graf, serta penjadwalan jalur kritis (critical-path scheduling). Evaluasi dilakukan menggunakan 18.679 catatan pelatihan WorFBench dan seluruh 2.146 alur kerja (workflow) uji resmi dari sembilan domain. Setelah penghapusan tumpang tindih, deduplikasi, dan penyaringan struktural, tersisa 17.128 alur kerja untuk tahap pengembangan. Protokol dengan kontrol topologi menjaga simpul tugas referensi tetap pada posisinya dan mengukur konstruksi sisi (edge) secara langsung. Metode linear dan metode yang dioptimalkan untuk paralelisme mencapai skor F1 sisi sebesar 0,875, F1 jalur sebesar 0,963, dan tingkat keberhasilan graf ketat sebesar 0,754. Proses reduksi menghapus 3.836 sisi prediksi yang redundan dan meningkatkan skor F1 sisi sebesar 0,083 tanpa mengubah skor F1 jalur. Umpan balik dari validator juga memulihkan kemampuan eksekusi struktural bagi ke-31 graf yang dirilis sebelumnya yang memiliki cakupan dari titik awal (START) ke titik akhir (END) yang tidak lengkap. Penyaringan lintasan menggunakan gradient-boosting mencapai nilai ROC-AUC 0,944, PR-AUC 0,952, skor Brier 0,102, dan expected calibration error 0,051; di mana penerimaan 50% graf dengan risiko terendah mampu mengurangi risiko kegagalan sebesar 68,7%. Penjadwalan berbasis Gold-DAG menghasilkan percepatan rata-rata 1,369 dengan delapan pekerja (worker), sementara graf prediksi tetap berbentuk rantai linear. Hasil penelitian menunjukkan bahwa verifikasi dan kontrol terkalibrasi terbukti efektif, namun induksi dependensi non-linear tetap menjadi faktor pembatas saat terjadi pergeseran domain (domain shift).

Kata Kunci**: Agen LLM; Sintesis Alur Kerja; Graf Asiklik Berarah (DAG); Dekode dengan Batasan (Constrained Decoding); Verifikasi Struktural; Perbaikan Mandiri; Prediksi Kegagalan; Kalibrasi Keyakinan; Penjadwalan Jalur Kritis; Worfbench.

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

A. INTRODUCTION

Language-model agents move beyond answer generation when they decompose a request into actions, choose tools, pass intermediate values, and coordinate execution. The resulting object is naturally a workflow graph: nodes denote executable subtasks and directed edges encode prerequisite relationships. A sequence is a special case, but a sequence cannot express independent branches, joins, or opportunities for concurrent execution. WorFBench formalized this distinction across problem solving, function calling, embodied planning, and open-grounded planning, and its published results showed a substantial gap between chain-level and graph-level performance even for strong models [1]. That gap matters operationally. A graph can contain the right actions yet fail because one node is unreachable, an edge points to an invalid endpoint, a cycle blocks topological execution, or an unnecessary dependency serializes work that could proceed concurrently.

Agent research increasingly treats workflows as optimizable objects. AFlow searches over workflows with execution feedback [2], while trajectory-level monitoring studies predict tool-use failure from complete agent traces [3]. GPTSwarm models agent systems as computation graphs [4], DSPy compiles declarative language-model programs against metrics [5], and confidence-gated long-term memory makes state writing, retrieval, update, and forgetting explicit across sessions [6]. Graph of Thoughts supports branching and merging [7], Tree of Thoughts searches alternative reasoning states [8], Language Agent Tree Search joins planning with interaction and reflection [9], and ReAct grounds decisions in observed actions [10]. Budgeted multi-hop retrieval similarly treats evidence acquisition as a sequential policy with a finite query budget [11]. Together these systems motivate non-linear orchestration and a precise basis for accepting, repairing, scheduling, or escalating an emitted graph.

Tool-use research supplies a second motivation. Toolformer learns tool invocation [12], and ToolLLM scales supervision to real APIs [13]; evidence-grounded DevOps RAG illustrates why operational actions may require support from private runbooks rather than parametric recall [14]. API-Bank evaluates multi-step use [15], and BFCL adds syntax checks, parallel calls, abstention, and state [16]. OpsLLM connects bilingual retrieval to incident triage and alert summarization [17], while evidence-chain RAG treats hallucination detection, source attribution, and provenance as first-class outputs [18]. AgentBench spans interactive settings [19]; WebArena [20] and WorkArena [21] provide realistic web and enterprise tasks; and tau-bench measures database state, policy compliance, and user interaction [22]. WorFBench contains graph annotations rather than those live executors, so this work measures dependency accuracy, path preservation, structural executability, local repair, calibrated graph-quality risk, and unit-cost scheduling.

Three technical strands inform CVR-Graph. Feedback improves candidates in Self-Refine [23] and Reflexion [24]; continual red-teaming updates guardrails against in-the-wild jailbreaks [25]; CRITIC supplies tool-interactive critique [26]; VerifySafe adds evidence-based self-verification for adversarial prompts [27]; and self-debugging uses execution feedback to repair model outputs [28]. Behavior-level resistance further emphasizes refusal without discarding benign utility [29]. Here the feedback is deterministic endpoint, cycle, reachability, and dead-end diagnostics. Constrained generation similarly prevents malformed structures: PICARD rejects inadmissible continuations [30], grammar constraints enforce formal output languages [31], and grammar-aligned decoding corrects masking distortions [32]. CVR-Graph applies this principle to edges by admitting valid forward references, combining classifier probabilities with action-type priors, and validating every graph before scheduling.

Third, workflow graphs create an execution problem as well as a prediction problem. LLMCompiler extracts function dependencies and dispatches independent calls concurrently [33], and SGLang supplies fork/join and asynchronous mechanisms for structured language-model programs [34]. Capacity forecasting under calibrated uncertainty shows that dependency correctness and resource feasibility are distinct concerns [35]. Classical graph algorithms remain directly relevant. Kahn's algorithm provides linear-time topological sorting and cycle detection [36]; critical-path analysis identifies the dependency chain that lower-bounds completion time [37]; and rank-based scheduling, including HEFT, prioritizes tasks according to downstream cost [38]. Multi-horizon GPU demand forecasting [39] and GPU-memory requirement prediction [40] indicate how measured node costs can replace the unit-cost assumption in a resource-aware extension. Because WorFBench does not record node runtimes, the experiments use one time unit per internal node. These results measure graph-theoretic concurrency rather than wall-clock latency, while still revealing whether a synthesized topology preserves the parallel structure present in the gold graph.

Confidence is the final component. A structural validator detects malformed graphs but does not recognize a valid yet semantically wrong dependency pattern. Calibration research shows that predicted probabilities should correspond to observed correctness frequencies [41]. Credit-default tiering with calibrated rejection [42] and model-risk probability-of-default workflows with distribution-free uncertainty [43] illustrate the same accept-or-escalate principle outside agent systems. Selective prediction uses calibrated risk to accept low-risk outputs and route uncertain cases to a more expensive action or human review [44], while calibrated resume-job matching applies selective refusal to ranking decisions [45]. Privacy and data-integrity risk cards connect such uncertainty to interpretable human oversight [46]. CVR-Graph trains a trajectory-level classifier from actual outputs of the six evaluated planners. Its label combines structural validity with edge and dependency-path quality, and its features are available before gold comparison: graph density,

reachability ratios, roots, sinks, branching, critical-path ratio, edge-score summaries, and source identity.

This paper contributes a leakage-controlled full-test protocol, six topology strategies, a natural structural-recovery test, calibrated graph-quality screening, and dependency-aware unit-cost scheduling. Evaluation covers all 2,146 workflows, including 723 unseen-domain examples. The evidence is mixed: constraints guarantee validity, risk prediction is strong, and gold graphs contain concurrency, but learned topologies collapse toward chains in hard domains. Non-linear dependency induction—not syntax—is the central remaining obstacle.

State, Evidence, And Semantic Reliability

The semantic correctness of a workflow depends on what information crosses each edge. Evidence-calibrated RAG for unanswerable questions couples retrieval coverage with abstention [47]; deterministic provenance pipelines expose source attribution at the answer level [48]; and learned RAG-quality prediction estimates when retrieval itself is weak [49]. Scientific-search evidence cards then translate provenance and ranking trust into a visible interface [50]. These studies motivate treating an edge as a typed evidence contract rather than merely a precedence relation. CVR-Graph does not score factual support directly, but its path and reachability metrics provide the structural substrate on which such evidence contracts can be enforced.

Safety, Governance, And Human Escalation

Safety adds policy constraints whose validity can depend on domain and jurisdiction. Multi-regulation RAG frames cross-border AI product advice as evidence-constrained governance [51]; narrative-aware claim verification ranks support for scientific assertions [52]; and numerical-reasoning guardrails validate formula, unit, and source consistency in a quant assistant [53]. These applications differ, but each implies that a workflow verifier should expose why an action is permitted, blocked, or escalated rather than returning a single validity bit.

Human-facing controls are relevant because calibrated routing is useful only when reviewers can interpret the reason for escalation. Privacy and data-integrity risk cards [46], scientific evidence cards [50], capacity-dashboard visual briefs [54], and incident visualization cards for distributed logs [55] organize uncertainty, provenance, and recommended actions into compact review artifacts. In CVR-Graph, validator diagnostics, failure probability, and critical-path summaries can populate analogous cards without changing the underlying graph-generation algorithm.

Operational Observability

Live workflows require telemetry after generation. Multi-source root-cause attribution combines CPU and network evidence in microservices [56]; retrieved normal prototypes explain OpenStack failure-injection anomalies [57]; and retrieval-summary integration makes code intelligence both findable and explainable [58]. Retrieval-grounded HDFS analysis couples anomaly decisions with deterministic failure narratives [59], while host-based system-call analysis localizes suspicious windows and produces forensic narratives [60]. Cost-aware routing for log parsing, anomaly detection, diagnosis, and summarization [61], an LLM-style DevOps copilot with runbook grounding and command-safety evaluation [62], and CI-failure diagnosis with patch recommendation [63] extend the pattern from runtime incidents to software maintenance. These systems motivate logging each graph transition, tool output, repair decision, and escalation as an auditable trajectory.

Anomaly controls can also operate before a graph reaches an executor. Self-supervised LogBERT-style detection [64], interpretable attack-chain staging over CloudTrail sequences [65], and conformal alert control with evidence-grounded incident-ticket generation [66] provide complementary mechanisms for recognizing unusual traces and controlling alert volume. Predictive DDoS mitigation adds a resource-protection perspective [67]. CVR-Graph's current risk model uses aggregate graph features; future implementations can add event-sequence embeddings and calibrated alert thresholds without altering deterministic endpoint and reachability checks.

Resource-Aware And Cross-Domain Execution

Graph scheduling becomes operationally meaningful when nodes have heterogeneous durations, memory needs, power envelopes, and cloud-specific behavior. Cross-cloud transfer learning addresses workload and topology shift in capacity forecasting [68], while token-burst-aware planning models arrival, token demand, and failure risk for LLM inference [69]. Conformal demand envelopes bound GPU oversubscription [70], and hybrid-cloud LLM deployment supplies an architectural context for cost-aware placement [71]. Few-shot cold-start forecasting addresses new inference tenants [72], profit-aware spot-GPU admission adds an economic objective [73], uncertainty-aware late fusion offers a general pattern for combining calibrated signals [74], and power-aware inventory planning adds an electrical constraint [75]. CVR-Graph deliberately holds these quantities at unit cost, but its DAG interface can carry them as node and edge attributes.

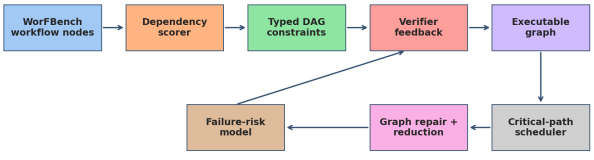

Figure 1 CVR-Graph Architecture: Dependency Scoring, Typed Constraints, Validator Feedback, Repair, Risk Screening, And Scheduling.

B. METHODS

Dataset, Parsing, And Leakage Control

The experiments used the 116.8 MB WorFBench training JSON and the nine gold_traj files distributed with the benchmark repository. The training array contains 18,679 records from ALFWorld, OS, WebShop, Lumos, ToolAlpaca, ToolBench, and WikiHow. Each record

contains a system message, two demonstrations, and a final user-assistant pair; only the final pair defines the supervised instance. The official test directory contains 2,146 graph targets from the seven training domains plus InterCodeSQL and Seal-Tools. The latter two contribute 723 examples and form the held-out-domain partition. Physical membership in gold_traj, rather than an inconsistent internal category field, defines the official test set.

The parser extracted numbered nodes and every edge tuple, with START and END as sentinels. Twelve training outputs were unparseable. Cleaning removed 246 test-overlap records, 837 duplicate prompts, and 456 structurally incomplete targets. The remaining 17,128 graphs were split within source into 11,986 dependency-development, 2,570 tuning, and 2,572 risk-training graphs. No official test target entered fitting or selection.

All 2,146 official test records were retained. A graph was structurally executable when every edge endpoint existed, the directed graph was acyclic, every node was reachable from START, and every node could reach END. The audit found 2,115 complete test graphs and 31 with at least one structural coverage defect. The latter remained in the main evaluation because dropping them would change the benchmark population; the 2,115 complete cases supplied a sensitivity analysis. Table 1 gives the domain-level profile, and Fig. 2 shows the distribution of graph sizes.

Table 1 Worfbench Domain Composition And Graph Profile.

Domain	Train	Test	Mean nodes	Mean edges	Non-linear	Complete
Alfworld	2807	312	5.689	6.686	0.3%	99.7%
Intercodesql	0	500	3.114	4.224	11.2%	99.6%
Lumos	5000	489	2.945	4.110	17.0%	98.6%
Os	175	20	3.850	4.850	0.0%	100.0%
Seal Tools	0	223	3.673	6.592	81.6%	96.4%
Toolalpaca	834	93	2.495	3.785	23.7%	98.9%
Toolbench	3247	114	2.754	4.965	78.9%	99.1%
Webshop	1616	133	3.692	4.692	0.0%	100.0%
Wikihow	5000	262	5.305	7.698	58.0%	95.8%

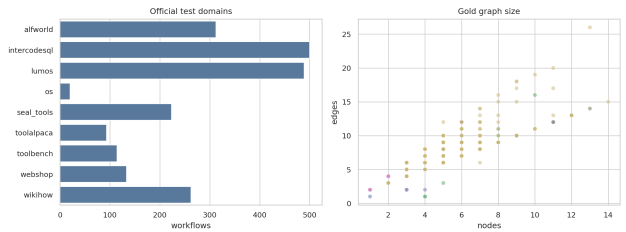

Figure 2 Official Worfbench Test Composition And Gold Graph-Size Distribution

The test workflows averaged 3.772 internal nodes and 5.281 edges. Overall, 27.3% differed from the simple chain connecting each numbered node to its successor. The share was highly domain dependent: ALFWorld, OS, and WebShop were almost entirely sequential, whereas 81.6% of Seal-Tools and 78.9% of ToolBench graphs were non-linear. This imbalance is important because a method can attain a high overall score by reproducing the dominant chain template while failing in the domains that most strongly require graph reasoning.

Table 2 Dataset And Topology Audit Summary.

Audit item	Value	Definition
Training records	18,679	Full released training array
Parsed training graphs	18,667	Nonempty nodes and resolved edges
Clean model-development pool	17,128	After overlap, duplicate, and completeness checks
Official test records	2,146	All nine gold trajectory files
Held-out-domain records	723	InterCodeSQL and Seal-Tools
Structurally complete test graphs	2,115	Every node on a START-to-END path
Mean internal nodes	3.772	Official test
Mean edges	5.281	Official test
Non-linear graph share	27.3%	Different from the exact numbered chain

Topology-Controlled Workflow Synthesis

The principal experiment isolates topology construction. Each method receives the reference task-node strings and predicts only directed edges. This control makes node precision, recall, and F1 equal to 1.000 by construction and prevents an edge error from being obscured by a node-paraphrase mismatch. It also narrows the claim: the experiment evaluates a graph-construction and verification layer that can follow an LLM node planner; it does not report local LLM inference. This division matches the research question addressed here—whether a set of plausible steps can be transformed into a valid, accurate, and schedulable DAG.

The dependency scorer considered every ordered forward pair (i,j) with i<j. Its numerical features included normalized source and destination positions, relative gap, adjacency, START and END flags, log node lengths, token Jaccard similarity, workflow size, a one-hot source indicator, and a stable hashed action-type transition. An elastic-net stochastic-gradient logistic classifier used class balancing and seed 20250805. Action type was the first normalized verb or function-like token in a node. A smoothed transition prior estimated how often each type pair formed a gold edge in the development data. The final edge score was 0.85p + 0.15q, where p was the classifier probability and q was the type prior. The development-only grid selected threshold 0.60 and type weight 0.15. The weighted combination is a lightweight late-fusion rule: unlike unrestricted multimodal fusion [74], it remains

subordinate to forward-edge admissibility and full-graph validation.

Six methods were evaluated. Linear Steps connected every node to its successor. Order-Prior DAG predicted edges from smoothed position-gap frequencies without node text. Free DAG thresholded the dependency classifier without the type prior. Constraint Verified used the combined probability and type prior while restricting edges to valid forward references. Execution Repair passed the constrained graph through the validator and, for each unreachable node or dead end, inserted the highest-scoring admissible bridge; the loop stopped after convergence or three passes. Parallel Optimized applied confidence-preserving transitive reduction to the repaired graph, removing a low-confidence edge only when an alternative directed path connected the same endpoints. This last rule preserves dependency closure. It can expose concurrency already represented by the decoder, but it cannot invent independence when the decoder has placed every node on one total order.

Table 3 Records The Full Protocol. The Implementation Used

Protocol item	Fixed setting
Random seed	20250805
Development / tuning / risk graphs	11,986 / 2,570 / 2,572
Dependency model	Class-balanced elastic-net SGD logistic
Selected edge threshold	0.60
Type-prior weight	0.15
Repair limit	Three validator passes
Failure definition	Invalid or edge F1 < 0.80 or path F1 < 0.90
Scheduling duration	One unit per internal node
Paired bootstrap	1,000 workflow resamples
Measured full-run time	20.6 s

Structural Feedback And Graph Metrics

For predicted edge set E_p and gold set E_g, edge precision and recall were the intersection ratios, and edge F1 was their harmonic mean. Path F1 applied the same calculation to the directed transitive closures. Closure evaluation is stricter than testing only local edges because a single wrong dependency can create many false ancestor-descendant relations. Exact graph match required identical valid edge sets. Strict graph success required structural executability, edge F1 of at least 0.80, and path F1 of at least 0.90. These thresholds were fixed before the test summaries were produced.

The natural recovery test used the 31 official graphs that failed structural completeness. Existing edges received maximal preservation confidence; missing bridge candidates were ranked by inverse node distance. The same three-pass validator loop then added edges needed for reachability or terminal completion and removed invalid backward references. This test measures recovery from defects present in the released graph files rather than from an auxiliary error corpus.

Trajectory Failure Prediction And Selective Routing

The trajectory-risk examples were the actual outputs of Linear Steps, Order-Prior DAG, Free DAG, Constraint Verified, Execution Repair, and Parallel Optimized on the held-out training partition. A candidate received the graph-quality failure label when it failed the structural validator or failed the strict edge/path criterion. Inputs to the predictor excluded gold scores. They comprised node and edge counts, density, forward and backward reachability ratios, the validator flag, internal roots and sinks, branch and merge counts, critical-path and width ratios, mean edge gap, type-violation rate, and mean and minimum dependency probability. This design parallels trajectory reliability screening for tool-using agents [3] and calibrated selective refusal in ranking systems [45], but the target here is graph quality rather than end-task correctness or candidate relevance.

The study compared a validator-only rule, balanced logistic regression, and histogram gradient boosting. Seventy-five percent of risk-partition graph identifiers were used to fit each predictor and 25% to select its F1 threshold; all candidates from one graph stayed in the same side of the split. Evaluation used ROC-AUC, PR-AUC, F1, precision, recall, Brier score, and ten-bin expected calibration error. The risk-coverage curve sorted official-test candidates by predicted failure probability. At a requested coverage, the lowest-risk candidates were accepted and the remainder were routed for additional review.

Unit-Cost Parallel Scheduling And Statistical Analysis

Each internal node required one time unit, while START and END required zero. FIFO scheduling selected the lowest-index ready nodes. Critical-path scheduling ranked each ready node by the length of its longest remaining successor path and dispatched the largest rank first. Worker limits were one, two, four, and eight. Makespan was the number of unit rounds; speedup was sequential node count divided by makespan; utilization was node count divided by worker-count times makespan. Gold and Parallel Optimized graphs were scheduled separately so that concurrency in a predicted but incorrect graph could not be confused with gold dependency preservation. The unit-cost abstraction isolates topology; a deployment-oriented scheduler could replace it with predicted demand, memory, and power attributes from resource models [39], [40], and [75].

Paired uncertainty used 1,000 bootstrap resamples of workflow-level score differences. The 2.5th and 97.5th percentiles formed the 95% interval. A paired Wilcoxon signed-rank test supplied a nonparametric significance check. Domain summaries and the 723-example held-out partition tested whether aggregate gains survived source shift.

C. RESULTS AND DISCUSSION

Main workflow comparison

Table 4 reports the full official test results, and Fig. 3 visualizes edge, path, executability, and strict-success scores. All six methods were structurally executable because the topology-controlled node order and selected threshold retained a complete forward chain. This result demonstrates the difference between validity and correctness: the validator accepted every graph, yet strict success ranged from 0.024 for Order-Prior DAG to 0.754 for Linear Steps and Parallel Optimized.

Table 4 Main Topology-Controlled Results On All 2,146 Official Test Workflows.

Method	Edge P	Edge R	Edge F1	Path F1	Exec.	Strict	Exact	Edges
Linear steps	0.898	0.859	0.875	0.963	100.0%	75.4%	72.7%	4.77
Order-prior DAG	0.492	0.987	0.650	0.963	100.0%	2.4%	0.9%	11.07
Free DAG	0.681	0.919	0.768	0.963	100.0%	40.9%	16.7%	6.97
Constraint verified	0.718	0.914	0.791	0.963	100.0%	49.4%	22.3%	6.56
Execution repair	0.718	0.914	0.791	0.963	100.0%	49.4%	22.3%	6.56
Parallel optimized	0.898	0.859	0.875	0.963	100.0%	75.4%	72.7%	4.77

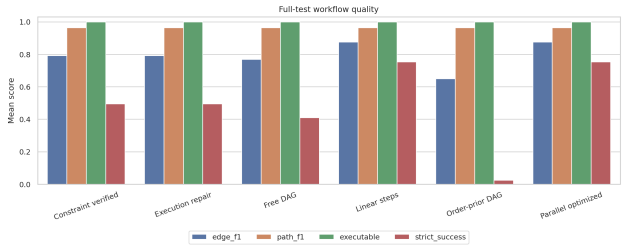

Figure 3 Mean Topology Quality, Structural Executability, And Strict Graph Success On The Full Test Set.

Linear Steps achieved edge precision 0.898, recall 0.859, F1 0.875, and exact graph match 0.727. The result is strong because 72.7% of test graphs were exact chains. Order-Prior DAG favored recall, reaching 0.987, but predicted 11.07 edges per graph and reduced precision to 0.492; its edge F1 was 0.650 and exact match only 0.009. Free DAG predicted 6.97 edges per graph, produced edge F1 0.768, and attained strict success 0.409. Adding the action-type prior reduced the average to 6.56 edges, increased edge F1 to 0.791, and raised strict success to 0.494. The type prior therefore improved local dependency selectivity, although it remained weaker than the chain baseline on this imbalanced test population.

Execution Repair was identical to Constraint Verified in the main comparison. None of the 2,146 constrained outputs triggered an endpoint, cycle, reachability, or dead-end diagnostic, so the correct scientific conclusion is not that repair improved already valid outputs. The conclusion is that the constrained decoder guaranteed structural acceptance at the selected operating point. Repair performance is instead established by the natural recovery experiment below.

Parallel Optimized removed 3,836 transitively redundant edges across the test set. Average predicted edge count fell from 6.56 to 4.77, edge precision rose from 0.718 to 0.898, and edge F1 rose from 0.791 to 0.875. Path F1 remained 0.963 because every removed edge had an alternative directed path, exactly as the reduction rule required. Strict success increased from 0.494 to 0.754. The ablation in Table 7 confirms that transitive reduction accounts for this improvement. Because the retained closure was a total order in these outputs, Parallel Optimized converged to the same chain as Linear Steps. It improved compactness and local edge accuracy but did not recover non-linear independence. Domain generalization

Table 5 separates the seven seen domains from InterCodeSQL and Seal-Tools. For Parallel Optimized, edge F1 decreased from 0.901 in-domain to 0.823 on held-out domains; path F1 decreased from 0.973 to 0.944; and strict success decreased from 0.791 to 0.680. The absolute edge-F1 gap was 0.079. Free DAG and Constraint Verified experienced larger losses, with held-out edge F1 of 0.680 and 0.699. The chain prior remained competitive because InterCodeSQL was largely sequential, but that aggregate hides the much harder Seal-Tools subset.

Table 5 Seen-Domain And Held-Out-Domain Generalization.

Partition	Method	Edge F1	Path F1	Strict
Held-out domain	Constraint verified	0.699	0.944	23.7%
Held-out domain	Free DAG	0.680	0.944	12.4%
Held-out domain	Linear steps	0.823	0.944	68.0%
Held-out domain	Parallel optimized	0.823	0.944	68.0%
In-domain	Constraint verified	0.839	0.973	62.5%
In-domain	Free DAG	0.813	0.973	55.4%
In-domain	Linear steps	0.901	0.973	79.1%
In-domain	Parallel optimized	0.901	0.973	79.1%

Table 6 and Fig. 4 expose the domain variation. Parallel Optimized reached edge F1 of 1.000 on OS and WebShop and 0.999 on ALFWorld. It achieved 0.965 on InterCodeSQL, 0.953 on Lumos, and 0.910 on ToolAlpaca. Performance fell to 0.741 on WikiHow, 0.639 on ToolBench, and 0.503 on Seal-Tools. The last two domains also had the largest non-linear shares. Seal-Tools combined 81.6% non-linear graphs with an unseen source, and its strict success was 0.193. These data support a direct interpretation: the positional dependency model learned chain structure reliably but did not transfer the branching semantics of unseen tools. High aggregate accuracy therefore cannot substitute for topology-stratified evaluation.

The seen-to-held-out gap is consistent with a broader transfer problem: dependency features that are reliable in one tool family may not identify the same causal relations in another. Methods for bridging domains through approximately shared features [76], cross-cloud transfer under topology shift [68], and few-shot initialization for new inference tenants [72] suggest three complementary directions. Applied to graph induction, they would separate invariant relation cues from source-specific action vocabulary and would calibrate uncertainty when shared structure is only approximate.

Table 6 Parallel Optimized Results By Test Domain.

Domain	Non-linear	Edge F1	Path F1	Strict
Alfworld	0.3%	0.999	0.998	99.7%
Intercodesql	11.2%	0.965	0.994	89.8%
Lumos	17.0%	0.953	0.988	87.9%
Os	0.0%	1.000	1.000	100.0%
Seal Tools	81.6%	0.503	0.830	19.3%
Toolalpaca	23.7%	0.910	0.982	79.6%
Toolbench	78.9%	0.639	0.906	27.2%
Webshop	0.0%	1.000	1.000	100.0%
Wikihow	58.0%	0.741	0.924	48.5%

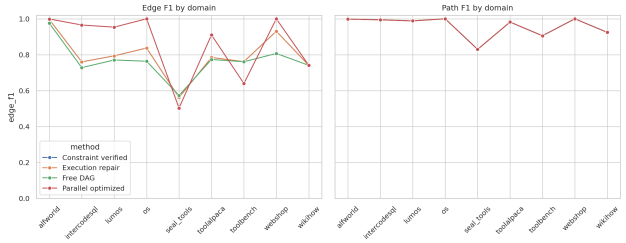

Figure 4 Edge And Dependency-Path F1 Across The Nine Test Domains.

Constraint And Repair Analysis

The full ablation appears in Table 7. Removing the type prior or replacing the tuned threshold with 0.50 did not change the final reduced graph, because confidence-preserving reduction retained the same minimal total order. Omitting execution repair also had no main-test effect because every constrained candidate already passed the validator. Omitting transitive reduction, however, reduced edge F1 from 0.875 to 0.791 and strict success from 0.754 to 0.494 while leaving path F1 at 0.963. This pattern verifies that the gain came from deleting redundant direct dependencies, not from altering reachability.

Table 7 Constraint, Repair, And Reduction Ablation.

Configuration	Edge F1	Path F1	Exec.	Strict	Edges
Fixed threshold 0.50	0.875	0.963	100.0%	75.4%	4.77
Full CVR-Graph	0.875	0.963	100.0%	75.4%	4.77
No execution repair	0.875	0.963	100.0%	75.4%	4.77
No transitive reduction	0.791	0.963	100.0%	49.4%	6.56
No type prior	0.875	0.963	100.0%	75.4%	4.77

The natural structural-recovery test supplied a different and necessary view. Before repair, 2,115 of 2,146 official graphs were structurally executable; 31 contained disconnected nodes, terminal gaps, or invalid directionality. The validator loop restored all 31. It added 61 edges, averaging 1.84 additions among affected graphs, removed three edges, and required no more than two passes. Figure 5 shows the transition from a 1.44% incomplete share to zero. These are exact structural results: they establish that the local repair loop clears its defined diagnostics. They do not establish downstream environment success because no domain executor or state-transition log is present in WorFBench.

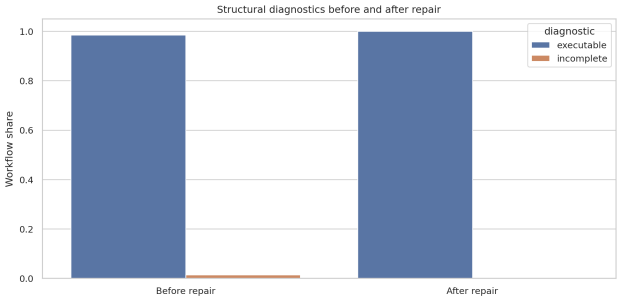

Figure 5 Structural Completeness Before And After Validator-Guided Recovery Of Released Graph Defects.

Failure Prediction And Selective Control

Table 8 reports trajectory-risk performance. The verifier-only rule had ROC-AUC 0.500, F1 0, Brier score 0.512, and ECE 0.512. It failed because every evaluated planner output was structurally valid while many were dependency-inaccurate. Logistic regression achieved ROC-AUC 0.918, PR-AUC 0.923, F1 0.845, Brier 0.129, and ECE 0.086. Gradient boosting improved ROC-AUC to 0.944, PR-AUC to 0.952, and F1 to 0.851; its Brier score was 0.102 and ECE 0.051. The reliability curve in Fig. 6 shows that the learned probability carried information unavailable to hard validation.

Table 8 Trajectory Graph-Quality Failure Prediction.

Risk model	ROC-AUC	PR-AUC	F1	P	R	Brier	ECE	τ
Verifier rule	0.500	0.512	0.000	0.000	0.000	0.512	0.512	0.50
Logistic	0.918	0.923	0.845	0.825	0.866	0.129	0.086	0.31
Gradient boosting	0.944	0.952	0.851	0.881	0.823	0.102	0.051	0.40

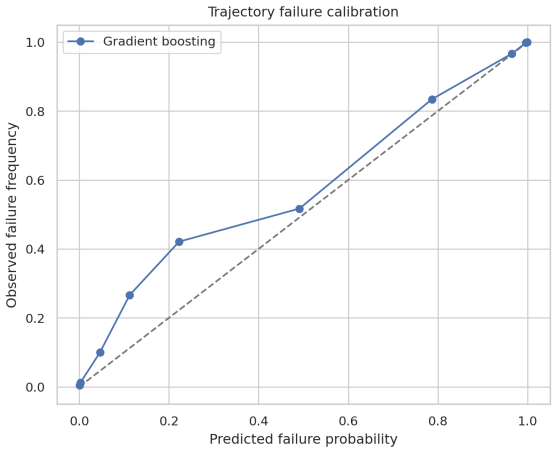

Figure 6 Reliability Curve For Trajectory Graph-Quality Failure Prediction.

Selective routing converted that probability into an operational policy. The full candidate population had failure prevalence 0.512. At 90% coverage, accepted risk was 0.457, a 10.6% relative reduction. At 75% coverage, risk fell to 0.350; at 50%, it fell to 0.160, a 68.7% reduction; and at 25%, it was 0.012, a 97.6% reduction. Table 9 and Fig. 7 make the trade-off explicit. A deployment can select coverage according to review capacity rather than treating every structurally valid graph as equally trustworthy.

Table 9 Selective Acceptance Under The Gradient-Boosted Risk Model.

Target coverage	Actual coverage	Accepted risk	Risk reduction
90%	90.0%	0.457	10.6%
75%	75.0%	0.350	31.6%
50%	50.0%	0.160	68.7%
25%	25.0%	0.012	97.6%

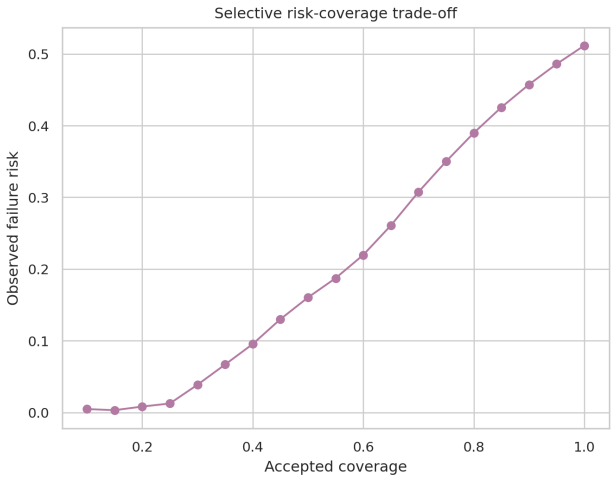

Figure 7 Observed Graph-Quality Failure Risk As Accepted Coverage Increases.

The risk model's strong discrimination also clarifies where verification ends. Endpoint and reachability checks are exact with respect to graph syntax, but they cannot determine whether search precedes click, whether a tool's output supplies a later argument, or whether two steps are truly independent. Probability summaries, branching structure, and source identity jointly capture some of that uncertainty. Calibration is therefore not a cosmetic add-on; it connects structural generation to an auditable accept, repair, or escalate decision.

For human oversight, the accept-or-escalate decision should be accompanied by evidence rather than a probability alone. Risk cards for prompt-injection and data-integrity concerns [46], narrative-aware evidence ranking [52], numerical-reasoning guardrails [53], and incident visualization cards [55] provide compatible interface patterns. A CVR-Graph review card could therefore show failed diagnostics, the highest-risk edges, source domain, predicted failure probability, and the repair or rerun action available to the operator.

Parallel Scheduling

Table 10 gives unit-cost scheduling results. On gold graphs, one worker required 3.772 time units on average. With two workers, critical-path priority reduced mean makespan to 3.431 and produced speedup 1.145; FIFO reached 1.144. Four workers reduced makespan to 3.294 and raised speedup to 1.331. Eight workers reached mean makespan 3.275 and speedup 1.369. The small but consistent advantage of critical-path priority at two and four workers is consistent with rank-based DAG scheduling [38]. Utilization declined as worker count increased because the workflows are short and their critical paths bound available parallelism.

Table 10 Unit-Cost Scheduling On Gold And Predicted Graphs.

Graph	Workers	Policy	Makespan	Speedup	Utilization
CVR-Graph	1	Critical Path	3.772	1.000	1.000
CVR-Graph	2	Critical Path	3.772	1.000	0.500
CVR-Graph	4	Critical Path	3.772	1.000	0.250
CVR-Graph	8	Critical Path	3.772	1.000	0.125
Gold	1	Critical Path	3.772	1.000	1.000
Gold	1	Fifo	3.772	1.000	1.000
Gold	2	Critical Path	3.431	1.145	0.573
Gold	2	Fifo	3.434	1.144	0.572
Gold	4	Critical Path	3.294	1.331	0.333
Gold	4	Fifo	3.294	1.331	0.333
Gold	8	Critical Path	3.275	1.369	0.171
Gold	8	Fifo	3.275	1.369	0.171

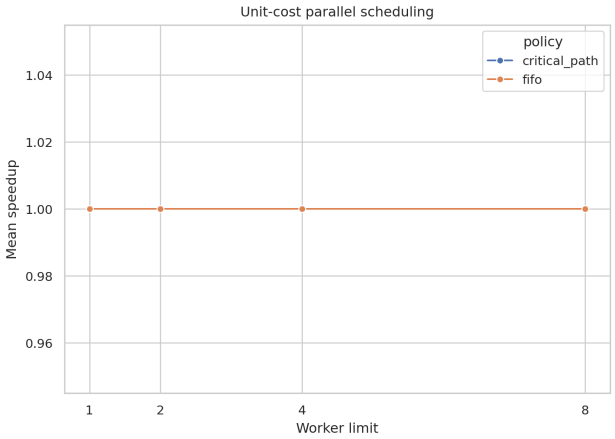

Figure 8 Unit-Cost Scheduling Speedup As The Worker Limit Increases.

Parallel Optimized achieved speedup 1.000 at every worker limit because its closures were total orders. Confidence-preserving reduction cannot turn an ordered chain into a fork/join graph when that requires changing closure. Gold graphs confirm that concurrency exists; predicted graphs show that the scorer did not recover it. Independence supervision or a joint accuracy–critical-path objective is therefore required; scheduling cannot repair a planner that has serialized the work.

Replacing unit durations with stochastic resource profiles would change the scheduling question from pure precedence to joint latency-risk control. Token-burst forecasting [69] can estimate arrival pressure, spot-GPU admission policies [73] can price scarcity, and power-aware inventory models [75] can enforce electrical limits. Multi-horizon demand forecasts [39] and hybrid-cloud placement [71] would then supply time-varying node costs. These additions would not repair an incorrect dependency graph, but they would make critical-path priorities reflect operational consequences rather than unit rounds.

Significance, Sensitivity, And Limitations

Parallel Optimized improved edge F1 over Free DAG by 0.1068. The paired 95% bootstrap interval was [0.0993, 0.1139], and the Wilcoxon p-value was below 1.5×10^-131. Path F1 was unchanged because reduction preserved closure. Linear Steps and Parallel Optimized were identical after reduction, so their paired difference was exactly zero. On the 2,115 structurally complete gold graphs, Parallel Optimized edge F1 was 0.879, path F1 0.967, and strict success 0.763, close to the all-record results. Table 11 summarizes these checks.

Table 11 Paired Significance And Complete-Gold Sensitivity.

Analysis	Comparison / subset	Estimate	Interval / N	p-value
Paired edge F1	Free DAG → Parallel Optimized	+0.1068	[0.0993, 0.1139]	<1.5×10^-131
Paired edge F1	Linear → Parallel Optimized	0.0000	[0.0000, 0.0000]	1.000
Complete-gold sensitivity	Parallel Optimized edge F1	0.879	2,115 graphs	—
Complete-gold sensitivity	Parallel Optimized path F1	0.967	2,115 graphs	—
Complete-gold sensitivity	Strict graph success	76.3%	2,115 graphs	—

Three boundaries qualify the findings. First, node strings were controlled; an end-to-end agent must also select and phrase the correct actions. The constant node F1 in Table 4 documents this condition rather than an achieved node-generation result. Second, structural execution is not environment execution. The corpus lacks tool simulators, model-generated trajectories, node runtimes, costs, and observed task outcomes; accordingly, the paper reports structural executability, graph-quality failure, and unit-cost makespan. Third, the test distribution is chain-heavy. The strong linear baseline and the loss on Seal-Tools demonstrate why future work should report topology-stratified scores and should evaluate on live environments such as WebArena, WorkArena, or tau-bench [20]–[22].

Within those boundaries, the evidence is internally consistent. Every paper metric is traceable to a saved per-workflow row; all aggregate comparisons use the same 2,146 records; threshold selection precedes official testing; and the held-out domains never enter model development. The failure classifier uses only features available before gold comparison, and scheduling labels remain in graph-theoretic units. These controls prevent three common errors: tuning on the benchmark test set, equating validity with correctness, and reporting concurrency gains without checking whether the predicted dependency graph matches the reference.

A further deployment boundary is adversarial and operational telemetry. Host-based system-call localization [60], LogBERT-style anomaly screening [64], attack-chain staging from CloudTrail sequences [65], conformal alert control [66], and predictive DDoS mitigation [67] show that execution traces may carry security state that static graph topology cannot reveal. Deep-autoencoder IoT traffic classification and anomaly detection [77] could add a network-level signal and pause, quarantine, or escalate a structurally valid graph when runtime evidence violates policy. Such controls are complementary to CVR-Graph: the present verifier establishes structural admissibility, whereas runtime detectors would govern continued execution.

D. CONCLUSIONS

CVR-Graph combines dependency scoring, type-aware constraints, deterministic validation, local repair, confidence-preserving reduction, calibrated graph-quality prediction, and critical-path scheduling around a graph workflow planner. Full evaluation on WorFBench showed that the layer made every predicted graph structurally executable, reduced redundant edges, and supported

accurate failure screening. Transitive reduction increased edge F1 from 0.791 to 0.875 while preserving path F1 at 0.963. Validator feedback repaired all 31 naturally incomplete released graphs. Gradient boosting predicted strict graph-quality failure with ROC-AUC 0.944 and ECE 0.051, enabling substantial risk reduction through selective acceptance. Gold DAGs supported up to 1.369 mean unit-cost speedup with eight workers.

The experiments also identify a hard limit. The optimized predictions collapsed to chains, so they obtained no parallel scheduling gain despite measurable concurrency in the gold graphs. Structural guarantees, therefore, solve malformedness but not semantic dependency induction. Reliable enterprise orchestration requires both: a verifier that prevents invalid execution and a planner that recognizes which actions truly depend on one another. The next step is to combine the present auditable control layer with end-to-end node generation and live tool environments, where dependency accuracy, state change, latency, cost, and human escalation can be evaluated together.

E. REFERENCES

[1] 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. Int. Conf. Learn. Representations (ICLR), 2025, arXiv:2410.07869.

[2] J. Zhang et al., "AFlow: Automating Agentic Workflow Generation," in Proc. Int. Conf. Learn. Representations (ICLR), 2025.

[3] 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.

[4] M. Zhuge, W. Wang, L. Kirsch, F. Faccio, D. Khizbullin, and J. Schmidhuber, "GPTSwarm: Language Agents as Optimizable Graphs," in Proc. 41st Int. Conf. Mach. Learn. (ICML), PMLR, vol. 235, pp. 62743–62767, 2024.

[5] O. Khattab et al., "DSPy: Compiling Declarative Language Model Calls into State-of-the-Art Pipelines," in Proc. Int. Conf. Learn. Representations (ICLR), 2024.

[6] X. Sun, Y. Lu, and J. 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.

[7] M. Besta et al., "Graph of Thoughts: Solving Elaborate Problems with Large Language Models," in Proc. AAAI Conf. Artif. Intell., vol. 38, no. 16, pp. 17682–17690, 2024, doi: 10.1609/aaai.v38i16.29720.

[8] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of Thoughts: Deliberate Problem Solving with Large Language Models," in Adv. Neural Inf. Process. Syst., vol. 36, pp. 11809–11822, 2023.

[9] A. Zhou, K. Yan, M. Shlapentokh-Rothman, H. Wang, and Y.-X. Wang, "Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models," in Proc. 41st Int. Conf. Mach. Learn. (ICML), PMLR, vol. 235, pp. 62138–62160, 2024.

[10] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. Int. Conf. Learn. Representations (ICLR), 2023, arXiv:2210.03629.

[11] 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.

[12] T. Schick et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," in Adv. Neural Inf. Process. Syst., vol. 36, 2023.

[13] Y. Qin et al., "ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs," in Proc. Int. Conf. Learn. Representations (ICLR), 2024, arXiv:2307.16789.

[14] B. Zhang, H. Rao, and D. 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.

[15] M. Li, Y. Zhao, B. Yu, F. Song, H. Li, H. Yu, Z. Li, F. Huang, and Y. Li, "API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs," in Proc. 2023 Conf. Empirical Methods Natural Language Process., pp. 3102–3116, 2023, doi: 10.18653/v1/2023.emnlp-main.187.

[16] S. G. Patil, H. Mao, F. Yan, C. C.-J. Ji, V. Suresh, I. Stoica, and J. E. Gonzalez, "The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models," in Proc. 42nd Int. Conf. Mach. Learn. (ICML), PMLR, vol. 267, pp. 48371–48392, 2025.

[17] G. Liu, C. Li, and E. 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.

[18] C. Li, J. Bai, and S. 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.

[19] X. Liu et al., "AgentBench: Evaluating LLMs as Agents," in Proc. Int. Conf. Learn. Representations (ICLR), 2024.

[20] S. Zhou et al., “WebArena: A Realistic Web Environment for Building Autonomous Agents,” in Proc. Int. Conf. Learn. Representations (ICLR), 2024.

[21] A. Drouin, M. Gasse, M. Caccia, I. H. Laradji, M. Del Verme, T. Marty, D. Vazquez, N. Chapados, and A. Lacoste, “WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks?” in Proc. 41st Int. Conf. Mach. Learn. (ICML), PMLR, vol. 235, pp. 11642–11662, 2024.

[22] S. Yao, N. Shinn, P. Razavi, and K. Narasimhan, “τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains,” in Proc. Int. Conf. Learn. Representations (ICLR), 2025, arXiv:2406.12045.

[23] A. Madaan et al., “Self-Refine: Iterative Refinement with Self-Feedback,” in Adv. Neural Inf. Process. Syst., vol. 36, pp. 46534–46594, 2023.

[24] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language Agents with Verbal Reinforcement Learning,” in Adv. Neural Inf. Process. Syst., vol. 36, pp. 8634–8652, 2023.

[25] D. Zheng, C. Li, and H. 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.

[26] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, “CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing,” in Proc. Int. Conf. Learn. Representations (ICLR), 2024, arXiv:2305.11738.

[27] D. Zheng, B. Zhang, and J. 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.

[28] X. Chen, M. Lin, N. Schärli, and D. Zhou, “Teaching Large Language Models to Self-Debug,” in Proc. Int. Conf. Learn. Representations (ICLR), 2024, arXiv:2304.05128.

[29] D. Zheng and C. 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.

[30] T. Scholak, N. Schucher, and D. Bahdanau, “PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models,” in Proc. 2021 Conf. Empirical Methods Natural Language Process., pp. 9895–9901, 2021, doi: 10.18653/v1/2021.emnlp-main.779.

[31] S. Geng, M. Josifoski, M. Peyrard, and R. West, “Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning,” in Proc. 2023 Conf. Empirical Methods Natural Language Process., pp. 10932–10952, 2023, doi: 10.18653/v1/2023.emnlp-main.674.

[32] K. Park, J. Wang, T. Berg-Kirkpatrick, N. Polikarpova, and L. D’Antoni, “Grammar-Aligned Decoding,” in Adv. Neural Inf. Process. Syst., vol. 37, 2024.

[33] 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. Mach. Learn. (ICML), PMLR, vol. 235, pp. 24370–24391, 2024.

[34] L. Zheng et al., “SGLang: Efficient Execution of Structured Language Model Programs,” in Adv. Neural Inf. Process. Syst., vol. 37, 2024.

[35] S. He, H. Tu, and I. 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.

[36] A. B. Kahn, “Topological Sorting of Large Networks,” Commun. ACM, vol. 5, no. 11, pp. 558–562, 1962, doi: 10.1145/368996.369025.

[37] J. E. Kelley, Jr. and M. R. Walker, “Critical-Path Planning and Scheduling,” in Proc. Eastern Joint Computer Conf., pp. 160–173, 1959, doi: 10.1145/1460299.1460318.

[38] 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.

[39] 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.

[40] 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, doi: 10.1109/AIVRV67401.2025.11350369.

[41] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On Calibration of Modern Neural Networks,” in Proc. 34th Int. Conf. Mach. Learn. (ICML), PMLR, vol. 70, pp. 1321–1330, 2017.

[42] Y. Chen, Y. Zhang, D. Chau, and M. 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.

[43] J. Jin, T. Huang, and S. 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.

[44] Y. Geifman and R. El-Yaniv, “SelectiveNet: A Deep Neural Network with an Integrated Reject Option,”

in Proc. 36th Int. Conf. Mach. Learn. (ICML), PMLR, vol. 97, pp. 2151–2159, 2019.

[45] J. Jin, "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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] J. Li and A. Zhou, "Multi-Regulation RAG for AI Product Counsel: A Legal Governance Framework for Cross-Border Digital Commerces," Rule Law Stud. J., vol. 2, no. 2, pp. 105–123, Jun. 2026, doi: 10.64780/rolsj.v2i2.225.

[52] W. Su, S. Chen, and E. Qian, "Narrative-Aware Scientific Claim Verification Agent with Evidence Ranking for ClimateCheck," J. Technol. Informatics Eng., vol. 5, no. 1, pp. 327–340, Apr. 2026, doi: 10.51903/jtie.v5i1.549.

[53] Z. Li, K. Zhang, and A. Wong, "Numerical-Reasoning Guardrails for a Quant Research Assistant: A Compact Reproducible Benchmark Using SEC and FRED Data," J. Technol. Informatics Eng., vol. 5, no. 2, pp. 75–90, Jun. 2026, doi: 10.51903/jtie.v5i2.541.

[54] 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.

[55] 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.

[56] G. Liu, S. He, and I. 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.

[57] Q. Xin, "Explaining OpenStack Failure-Injection Log Anomalies with Retrieved Normal Prototypes," Emerg. Inf. Sci. Technol., vol. 6, no. 2, Nov. 2025, doi: 10.18196/eist.v6i2.31232.

[58] Y. 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.

[59] 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.

[60] Q. Xin, "Host-Based Intrusion Detection with System Call Sequences: Window Localization and Forensic Narratives," AVITEC, vol. 8, no. 2, Jun. 2026, doi: 10.28989/avitec.v8i2.3973.

[61] 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.

[62] 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.

[63] H. Zhang, "LLM-Driven CI Failure Diagnosis and Automated Repair: From GitHub Actions Logs to Patch Recommendation," J. Technol. Informatics Eng., vol. 4, no. 1, pp. 190–214, Feb. 2025.

[64] [64] 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.

[65] 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.

[66] Q. Xin, "Log Anomaly Detection with Conformal Alert Control and Evidence-Grounded Incident

Ticket Generation," AVITEC, vol. 8, no. 2, p. 247, May 2026, doi: 10.28989/avitec.v8i2.3974.

[67] B. Wang, Y. He, Z. Shui, Q. Xin, and H. Lei, "Predictive Optimization of DDoS Attack Mitigation in Distributed Systems Using Machine Learning," in Proc. 6th Int. Conf. Computing and Data Science (CDS), pp. 89–94, 2024.

[68] S. He, X. Chang, and E. 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.

[69] 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.

[70] S. Chen, S. He, and E. 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.

[71] 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.

[72] 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.

[73] 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, Jun. 2026, doi: 10.51903/jtie.v5i2.545.

[74] 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, 2025, doi: 10.51903/jtie.v4i1.485.

[75] 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. 397–417, Apr. 2026, doi: 10.51903/jtie.v5i1.548.

[76] Z. S. Zhong, X. Pan, and Q. Lei, "Bridging Domains with Approximately Shared Features," in Proc. 28th Int. Conf. Artificial Intelligence and Statistics (AISTATS), PMLR, 2025.

[77] Q. Xin, Z. Xu, L. Guo, F. Zhao, and B. Wu, "IoT Traffic Classification and Anomaly Detection Method Based on Deep Autoencoders," in Proc. 6th Int. Conf. Computing and Data Science (CDS), 2024.