Academic Journal of Applied Sciences
ISSN: 3105-0549 (Print), ISSN: 3105-0557 (Online) | Vol. 3, No. 3, 2026

Graph-Based Detection of Policy Conflicts in Zero-Trust Multi-Cloud Environments

Noor Bakker [1, *], Bas van der Meer [2]

[1] Information Technology Group, Wageningen University & Research, Wageningen, Netherlands

[2] Erasmus School of Economics, Erasmus University Rotterdam, Rotterdam, Netherlands

Abstract: A zero-trust deployment spanning multiple clouds may contain independently maintained authorization rules that express incompatible decisions for overlapping requests. This paper presents a compact, graph-indexed approach for detecting such cross-control-plane inconsistencies. Policies are normalized into finite sets of principals, actions, resource groups, network zones, and assurance levels, together with a cloud identifier and an allow/deny effect. An incidence graph connects policies with their canonical tokens. Candidate pairs are obtained by intersecting opposite-effect identity and resource postings, after which the remaining policy dimensions are checked exactly. The procedure reports the same conflict edges as exhaustive pairwise evaluation under the stated model. We implemented both algorithms in Python and evaluated 500–4,000 synthetically generated policies for three abstract cloud providers, using five seeds per size. In all 20 trials, the conflict sets matched. At 4,000 policies, median execution time decreased from 922.14 ms to 140.58 ms, a 6.81× median trial-level speedup, while the indexed method removed 98.43% of eligible pairs from exact verification. The results demonstrate practical candidate reduction for a transparent, bounded cross-cloud policy model.

Keywords: Zero trust, multi-cloud security, access control, graph indexing, policy conflict, authorization.

1. Introduction

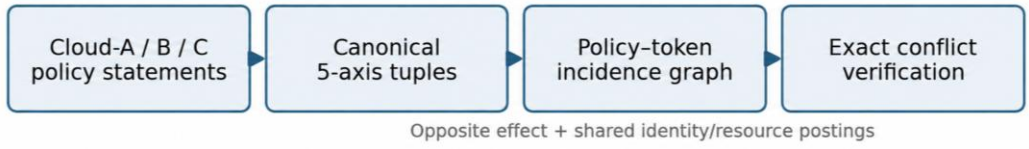


Figure 1. Policy normalization, incidence graph construction, and exact conflict verification.

Zero-trust architecture (ZTA) treats every access attempt as an authorization decision rather than granting implicit trust from network placement. NIST SP 800-207 defines the architectural principles, while SP 800-207A extends the guidance to cloud-native applications spread across cloud and on-premises environments [1, 2]. A practical deployment must therefore coordinate identities, actions, resources, and changing contextual attributes across several enforcement systems. Service-mesh identity and authorization guidelines further motivate resource-specific policy inspection [4–6].

In a multi-cloud application, an organization may intend to prohibit a workload from reading a sensitive dataset while another control plane grants the same workload an equivalent read operation. Such a discrepancy is not necessarily a violation of a single cloud provider's evaluation rules: providers differ in policy types, scope inheritance, and deny precedence [35–42]. It is a cross-control-plane inconsistency that should be surfaced to administrators before policy changes are deployed. The central engineering challenge is to identify potentially overlapping rules without comparing every pair.

This paper studies a deliberately bounded policy space that retains the fields needed to express this kind of overlap. It makes three contributions: (i) a canonical set-based representation for comparing provider-tagged authorization statements; (ii) a bipartite policy–token graph with a sound candidate filter; and (iii) an executed, multi-seed runtime and equivalence evaluation against exhaustive comparison. Unlike generic similarity-only screening, the final decision in our model is an exact set-intersection test. The implementation has no dependence on a particular IAM vendor or machine-learning model.

2. Background and Related Work

2.1. Authorization Models and Zero-Trust Context

RBAC links subjects to roles and permissions [8, 9]; ABAC extends authorization to subject, object, action, and environment attributes [3]. Usage-control models introduce conditions and decision continuity [10, 11]. Distributed identity and trust can be represented by principal relationships, delegation, or attestations [12–17]. Formal access-control safety is a long-standing research problem [13]. Modem authorization systems also exploit relationship graphs and expressive policy languages: Zanzibar supports global relationship-based evaluation [30], and Cedar provides analyzable policy definitions [28, 29]. The present work uses explicit, finite identity sets and categorical context bins instead of reproducing these richer semantics.

2.2. Policy Analysis and Graph Screening

Policy anomaly detection predates cloud computing. Distributed firewall analysis, formal firewall models, and high-level rule extraction examine shadowing, inconsistency, and redundancy [20–24]. XACML provides a standardized

policy vocabulary [34], while Margrave uses symbolic representations for verification and policy-change analysis [19]. Heterogeneous-policy conflict detection has also been studied with decision diagrams and policy translation [33]. More recently, graph-based formal access control has been used for conflict and redundancy detection [31], and graph similarity filtering has been proposed to pre-screen cloud-policy verification tasks [32].

In parallel, cloud-specific policy verification has employed satisfiability-modulo-theories (SMT) reasoning: Zelkova analyzes AWS policies [25], and subsequent work models multi-policy authorization engines and IP-related counting questions [26, 27]. This line addresses a broader class of language semantics. Our implementation instead targets a small, checkable cross-provider overlap predicate and uses graph incidence to avoid unnecessary exact comparisons. Identity and context translation are explicit preprocessing obligations, rather than properties inferred by the graph.

3. Policy Model and Conflict Definition

3.1. Canonical Rule Representation

A normalized policy is $p = (c, e, I, A, R, Z, L)$, where c is a provider tag; $e \in \{allow, deny\}$; and I, A, R, Z, and L are nonempty finite sets of canonical identities, operations, resource classes, network zones, and assurance levels. The sets are interpreted as alternatives within each axis, while all five axes are jointly required. Thus the request space covered by p is $W(p) = I \times A \times R \times Z \times L$. A request is represented by one token from each axis. The model captures contextual matching without assuming a global role hierarchy or the native evaluation order of any cloud IAM product.

Provider-specific operation strings are converted to canonical action labels by a mapping prepared for the environment. In the synthetic evaluation, this mapping simply removes the provider prefix (e.g., Cloud-A::ac00 and Cloud-B::ac00 both become ac00). Identities and resources are already shared canonical tokens in the generator. For real integrations, federated principal mappings, resource ownership, policy inheritance, and provider-specific effects would require independently validated adapters. This distinction matters because similar service names do not automatically imply equivalent permission semantics.

3.2. Exact Inconsistency Predicate

Two policies p and q conflict if (1) $c(p) \neq c(q)$, (2) $e(p) \neq e(q)$, and (3) $W(p) \cap W(q) \neq \emptyset$. Because the request spaces are Cartesian products, the last condition is equivalent to five nonempty intersections: $I(p) \cap I(q)$, $A(p) \cap A(q)$, $R(p) \cap R(q)$, $Z(p) \cap Z(q)$, and $L(p) \cap L(q)$. A witness is a single normalized request satisfying both rules but receiving opposite policy-intent decisions. The predicate detects disagreement between distinct control planes; it does not calculate an effective allow/deny result after applying AWS, Azure, or Google Cloud precedence semantics [36, 39, 41].

For example, suppose policy p0 in Cloud-A permits identity id00 to perform action ac00 on resource class re00, in zone zo00 and assurance class as00. Policy p1 in Cloud-B denies exactly the same canonical request. Their policy-intent decisions conflict. Replacing the second policy's zone with zo01 removes the overlap. Similarly, two allow policies do not form an opposing-decision conflict. The choice of a categorical network zone and assurance class is compatible with common identity and context requirements in authorization guidance [2, 3, 42, 44, 47].

3.3. Semantics and Conflict Witnesses

The proposed detector is an intent-consistency analyzer, not a replacement for a native policy decision point. It flags intersecting decision regions maintained by different providers even if a later deployment-layer rule would resolve access in practice. This definition is useful for reviewing cross-cloud policy drift, redundant exception approvals, or inconsistent migration rules. A conflict record can retain the two policy identifiers, two providers, the opposed effects, and one witness token on each of the five axes, enabling an administrator to inspect a concrete matched request.

A complete vocabulary mapping is a precondition for a meaningful comparison: two identities representing the same federated principal must resolve to one identity token, and two resource operations with different semantics must not be mapped to the same operation token. Normalization errors can cause missed or spurious conflicts independently of the indexing algorithm. In the evaluation this potential source of error is controlled by generating all rules from one known canonical vocabulary and encoding action names with provider prefixes only.

4. Graph-Indexed Conflict Detection

4.1. Incidence Graph and Candidate Pairs

The policy graph $G = (P \cup T, E)$ is bipartite: P contains policy vertices and T contains typed token vertices such as (identity, id03) or (resource, re12). An edge (p, t) is inserted when the policy contains token t. For efficient access, the Python implementation represents selected incidence neighborhoods as inverted postings indexed by (token, effect). The policy effect and provider remain vertex attributes, and the original sets are retained for final verification. This is a sparse graph representation rather than a materialized all-pairs adjacency matrix.

For a policy p, let OI(p) be the union of policies in postings for its identity tokens with the opposite effect; define OR(p) analogously for resource tokens. Potential partners lie in $C(p) = OI(p) \cap OR(p)$. We discard previously visited identifiers and policies from the same provider. Each surviving partner is tested for nonempty action, zone, and assurance overlap. The resulting undirected edge (p, q) is output only when all parts of the conflict predicate hold. The identity and resource intersection tests are already guaranteed by the postings.

4.2. Algorithm and Correctness

Input: normalized policies $P = \{p0, ..., pn-1\}$

1. Index each identity and resource token by policy effect.
2. For every p in P, retrieve opposite-effect identity neighbors NI.
3. Retrieve opposite-effect resource neighbors NR.
4. For q in $NI \cap NR$ with $q.id > p.id$ and $q.cloud \neq p.cloud$:
5. if action, zone, and assurance sets overlap: emit (p, q).

Output: sorted cross-provider conflict edges.

Proposition 1 (equivalence). For the finite normalized policy model, the graph procedure returns exactly the pairs returned by exhaustive pairwise evaluation. Proof: every emitted pair has opposite effects, distinct providers, shared identity and resource tokens, and verified intersections on the other three dimensions, so it satisfies the definition (soundness). Conversely, every conflicting pair shares at least

one identity token and one resource token and has opposite effects, placing its two vertices in both corresponding postings. The pair enters the candidate intersection and passes the remaining checks (completeness). The identifier condition removes only duplicate orderings; uniqueness of policy identifiers is checked before indexing, and IDs need not be consecutive. This proves equality of the output sets.

Exhaustive enumeration performs $\Theta(n^2)$ pair visits. Graph construction costs time proportional to the number of policy–token incidences. Candidate enumeration depends on posting-list lengths and overlaps and remains quadratic in the worst case (e.g., every policy contains the same tokens). With selective identity and resource memberships, however, the intersection can remove most pairs before checks on three additional axes. Consequently, empirical candidate reduction is reported alongside runtime; no asymptotic improvement is assumed for dense workloads.

4.3. Screening Cost and Implementation Choices

The baseline performs the two categorical gate checks (provider and effect) and then intersects each of the five sets in sequence. The indexed implementation stores an inverted lookup from each identity or resource token and effect to the set of policy identifiers that possess the token. For each source policy, it unions the opposite-effect postings on each indexed axis, intersects the two identifier sets, and evaluates each surviving identifier once. These operations use Python sets and preserve the original normalized policy records. No approximate membership tests, learned predictors, or probabilistic sketches are involved; therefore the initial graph filter cannot introduce false negatives relative to the defined predicate. Policy ID lookup is independent of list order, enabling nonconsecutive identifiers in imported inventories.

Posting construction requires one insertion per identity or resource membership. The identifier intersections are efficient when vocabulary tokens have sufficiently selective postings. In a worst-case policy collection with many universal rules, posting sets become dense, and index construction can provide little benefit. Both runtime and the fraction of eliminated candidates should therefore be measured when adopting graph filtering for a specific policy collection. The sensitivity experiment in Section V-D explicitly varies vocabulary selectivity while keeping the policy count fixed.

5. Experimental Evaluation

5.1. Generation, Baseline, and Protocol

We evaluated an executable Python 3.13.5 implementation on a Linux x86-64 environment reporting an AMD EPYC 9V74 processor (five available logical CPUs). The input is a synthetic, cloud-inspired population—not policies collected from production AWS, Azure, or Google Cloud deployments. For each run, the generator independently samples three abstract provider labels and a binary effect, with 24 identity tokens, 12 action tokens, 40 resource classes, four network zones, and three assurance classes. Identity, action, and resource subset sizes are uniformly chosen from one to three; zone and assurance subset sizes from one to two. All selections use Python's seeded pseudo-random generator.

The benchmark includes $n \in \{500, 1{,}000, 2{,}000, 4{,}000\}$ policies, with seeds 17, 29, 43, 71, and 101 for every size. Two single-process, sequential implementations are compared: exhaustive checking of all $n(n-1)/2$ pairs and graph-indexed candidate retrieval followed by the same exact overlap predicate. Both implementations receive identical policy objects. Python perf_counter measures wall-clock elapsed time for each algorithm; input generation and correctness assertions are excluded from the recorded timing. Runtime, candidate counts, and per-run speedup ratios are each summarized by their median across five seeds; the reported median speedup is not necessarily the ratio of the two median times. No cloud API calls, external datasets, or pre-trained models are involved.

5.2. Runtime and Conflict Equality

Table I presents observed medians. Every one of the 20 scale/seed runs produced identical conflict-edge lists for the exhaustive and indexed approaches, using an equality assertion on the sorted pairs. Independently, nine designed edge cases—covering each empty-intersection axis, provider equality, effect equality, and overlapping nonidentical sets—passed, as did 50 smaller randomized equivalence tests. Another 20 randomized cases verified invariance under policy-ID remapping and input shuffling; witness validation checked each reported conflict in a separate 200-policy case, and duplicate IDs were rejected. These checks establish implementation agreement for the defined conflict relation rather than accuracy against a production-policy ground truth.

Table I. Median Results Over Five Seeds Per Policy Count

Policies	Pairwise (ms)	Graph (ms)	Speedup
500	12.81	2.61	4.84×
1,000	51.04	8.22	6.28×
2,000	231.00	34.55	5.89×
4,000	922.14	140.58	6.81×

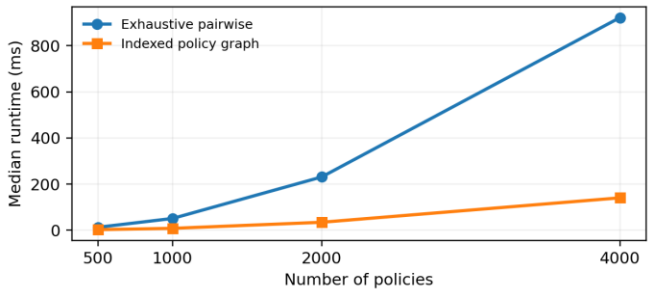


Figure 2. Measured median elapsed time; each point aggregates five synthetic runs.

At $n = 500$, median runtime was 12.81 ms for exhaustive comparison and 2.61 ms for graph indexing (median trial-level speedup: 4.84×). At $n = 4{,}000$, the corresponding medians were 922.14 ms and 140.58 ms (median trial-level speedup: 6.81×). Fig. 2 shows that the absolute gap grows as the rule count increases in this generated workload. These reported values are directly computed from the 20 recorded executions; per-run times and seed-specific conflict counts are retained in the result tables.

5.3. Candidate Reduction and Output Size

The graph candidate ratio is computed using as denominator only cross-provider, opposite-effect pairs; within-provider pairs and equal-effect pairs are ineligible under the conflict definition. The indexed method reduced exact verifications by 98.52%, 98.47%, 98.46%, and 98.43% at $n = 500$, 1,000, 2,000, and 4,000, respectively (five-seed

medians). At n = 4,000, the median number of candidates checked after the postings intersection was 41,996, versus a median of 4,438 reported conflict edges. Because the retained candidates still require action and context checks, candidate count is greater than the final conflict count. Fig. 3 depicts pruning results.

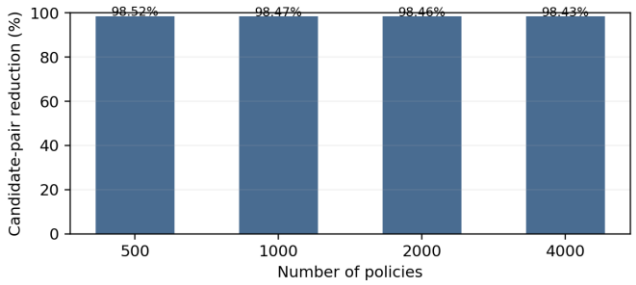


Figure 3. Median reduction in opposite-effect, cross-provider candidate pairs.

5.4. Resource-Vocabulary Sensitivity

To test whether the effect depends on selective token memberships, an independent set of 20 runs fixed the policy population at 2,000 records while varying the number of resource classes among 10, 20, 40, and 80. The other four vocabularies, subset-size sampling rules, implementation, and five random seeds were unchanged. All conflict-pair outputs again matched the exhaustive baseline for each run. Increasing the resource vocabulary from 10 to 80 reduced the median graph candidate count from 37,608 to 5,061, because two randomly generated rules were less likely to share a resource class.

The corresponding median candidate reductions were 94.35%, 97.01%, 98.46%, and 99.24%. Median trial-level graph speedups were 3.78×, 5.00×, 6.72×, and 8.30×, respectively. The median number of genuine conflict edges also declined from 4,011 to 552 as the resource universe expanded. These values are from a separate execution sweep, so small timing differences from the main n = 2,000 experiment reflect measurement variability. Fig. 4 visualizes the observed dependency on posting-list selectivity: the representation is exact at each setting, but its computational benefit varies with workload structure.

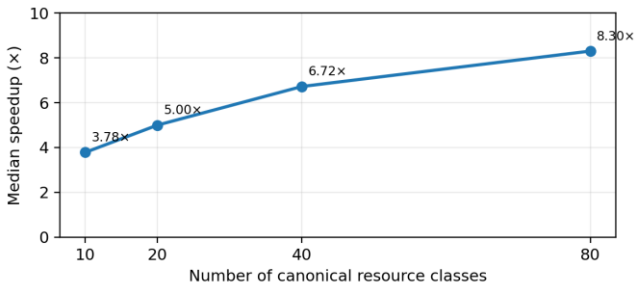


Figure 4. Observed graph speedup at 2,000 policies as resource-class vocabulary varies.

5.5. Index-Axis Ablation

We next isolated the benefit of intersecting two posting neighborhoods. For each of the same five seeds at n = 2,000, we evaluated three exact methods that index identity only, resource only, or both fields. A single-axis variant still verifies every unindexed axis; therefore the output definition and reference conflict set are unchanged. Each method includes its own posting construction time, and the measurements form a separate timing sweep from Table I.

Table II. Index-Axis Ablation At 2,000 Policies (Five Seeds)

Indexed axes	Candidates	Time (ms)
Identity	105,757	78.78
Resource	64,833	44.15
Identity + resource	10,262	31.40

The dual-axis filter retained a median of 10,262 candidates, compared with 64,833 for resource-only indexing and 105,757 for identity-only indexing. The corresponding median times were 31.40 ms, 44.15 ms, and 78.78 ms. All 15 ablation executions matched the exhaustive output exactly. These results attribute the lower verification workload to the intersection of selective posting neighborhoods rather than to a different conflict predicate.

5.6. Interpretation and Scope

The speedup is explained by inexpensive graph-neighborhood intersections over the two most selective indexed axes. The resulting candidate set is then tested by the same predicate as the exhaustive oracle, so filtering does not change the output. The design also supplies a straightforward witness for a flagged discrepancy: any element selected from each of the five nonempty intersections forms a request covered by both policies. Graph edges may therefore be accompanied by concrete identities, actions, resources, and contexts during an operational review.

The experiment isolates graph filtering from ingestion and translation costs. The canonical token spaces are finite, and resource classes are represented as discrete labels; wildcard semantics, inherited permissions, conditional expression languages, and policy combining algorithms are not evaluated. Their provider-specific rules are described in the relevant cloud documentation [35–42]. An operational pipeline would normalize those expressions before graph construction and recheck reported witness requests under each vendor's enforcement semantics. The tested contribution is exact conflict enumeration in a clearly defined shared model, together with measured computational savings.

5.7. Deployment Workflow

A deployable implementation would place provider-specific translation ahead of the shared graph stage. First, native policy inventories are exported from each control plane, retaining rule identifiers and scope metadata. Second, explicit dictionaries associate federated principals, business actions, and resource-group identifiers across providers. Third, conditions that can be represented by finite categorical domains are transformed into the five-axis model; rules containing unsupported expressions are routed to native analysis rather than silently simplified. Fourth, the graph stage emits conflict pairs and concrete witness requests. Finally, an administrator or policy test harness checks those witnesses against the original enforcement points before remediation. This sequence preserves the advantage of a common cross-provider index without equating distinct provider policy languages.

The comparison can also complement organizational policy-as-code checks. Rego is an example of a portable policy language [43], while Kubernetes RBAC and network policies supply additional authorization and connectivity scopes that may need separate normalization [45, 46]. Workload identity documents can provide canonical identity inputs [44], and JWT claims can carry structured context attributes [47]. In each case, the index requires a shared

meaning for tokens, not merely shared strings. An implementation can cache token postings and update the affected memberships when a policy is added or removed; evaluating incremental-update throughput is outside the present runtime protocol [48-55].

5.8. Validity Considerations

The experiments measure wall-clock behavior of the two complete Python detectors, including index creation for the graph method. They do not compare runtime across programming languages or distributed configurations. Input statements are generated independently from fixed categorical spaces, making the workload suitable for controlled correctness and selectivity analysis. Enterprise IAM inventories can differ through correlated permissions, broad wildcard rules, hierarchical scopes, and skewed group memberships. As a result, the measured acceleration should be interpreted for the stated generator and finite predicate; the observed sensitivity trend provides a way to reason about policy collections with different posting densities [56-61].

A cross-provider allow/deny pair should be investigated as inconsistent policy intent, not automatically labeled an exploitable privilege escalation. Provider-specific precedence, infrastructure placement, and actual request routing determine whether both statements are enforceable for the same target. Reporting the witness intersections alongside source policy identifiers provides the necessary starting point for that validation. These distinctions separate an exact detector for its formal relation from an end-to-end audit of a deployed multi-cloud security architecture.

6. Conclusion

This paper presented a graph-indexed detector for contradictory policy intent across zero-trust multi-cloud control planes. By representing identity and resource membership as opposite-effect postings and verifying the remaining request dimensions exactly, the detector preserved exhaustive enumeration results while reducing verification workload. Across 20 synthetic runs, equivalence held at every tested scale; at 4,000 policies median trial-level acceleration was 6.81× and candidate reduction was 98.43%. The method can serve as a lightweight consistency-checking layer when canonical identity, resource, action, and context mappings are available. A natural next step is to evaluate such mappings against provider-native policy semantics and organizational policy inventories.

References

[1] Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero trust architecture. NIST SP 800-207. https://doi.org/10.6028/NIST.SP.800-207

[2] Mo, T., Wang, G., Feng, E., & Li, P. (2023). Execution-feedback-guided context selection for large language model-based software testing and debugging. World Journal of Information and Knowledge Management, 1(1), 55–64.

[3] Wang, Y., & Zhang, C. (2023). Aeris: Adaptive executor-scoped resource management for privacy-preserving data processing in Apache Spark. Eurasia Journal of Science and Technology, 5(2), 25–33.

[4] Lin, Y., Yue, X., Bian, M., Zhu, R., & Teng, D. (2026). Subgraph-level temporal graph transformer with laundering-motif prototypes for anti-money laundering on blockchains. IEEE Access.

[5] Teng, D. (2025). TRIAD: Deadline- and reliability-aware scheduling for post-earthquake structural assessment workflows. World Journal of Engineering Research, 3(6), 55–66.

[6] Wang, J., Ge, Y., Liu, J., & Jiang, B. (2024). Early warning of liquidity stress in U.S. banks with seasonally aligned frailty hazards and delayed-feedback conformal alarms. Journal of Trends in Finance and Economics, 1(1), 85–100.

[7] Cao, J., Zheng, W., Ge, Y., & Wang, J. (2025). DriftShield: Autonomous fraud detection via actor-critic reinforcement learning with dynamic feature reweighting. IEEE Open Journal of the Computer Society, 6, 1166–1177. https://doi.org/10.1109/OJCS.2025.3495211

[8] Liang, Y., Jiao, Y., Ping, W., Fan, H., & Han, X. (2026). Adaptive event-driven labeling: A neuro-symbolic multiagent framework for causal inference in non-stationary time series. IEEE Access.

[9] Fan, H., Liang, Y., Jiao, Y., Ping, W., & Lin, Y. (2026). An explainable temporal machine learning framework for financial fraud risk prediction in listed institutions. Highlights in Business, Economics and Management, 70, 173–185.

[10] Zhao, X., Liu, J., Wang, Y., & Wang, J. (2026). CryptoMamba-SSM: Linear complexity state space models for cryptocurrency volatility prediction. IEEE Open Journal of the Computer Society, 7, 226–243. https://doi.org/10.1109/OJCS.2026.3582142

[11] Chen, J., Cui, Y., Zhang, X., Yang, J., & Zhou, M. (2024). Temporal convolutional network for carbon tax projection: A data-driven approach. Applied Sciences, 14(20), 9213. https://doi.org/10.3390/app14209213

[12] Zhang, F., Shen, A., & Huang, S. (2026). TRIAD: Co-designing mixed-precision quantization, operator fusion, and on-chip memory reuse for real-time vision–language–action inference on automotive NPUs. Results in Engineering.

[13] Bian, M., Wang, Y., & Lin, Y. (2024). Explainable supplier risk prediction for critical telecommunications infrastructure procurement: An empirical XGBoost–SHAP framework. Social Science and Management, 1(1), 64–77.

[14] Wang, G., Li, P., Feng, E., Shen, A., & Rhee, M. (2026). A dual-process memory-augmented large language model agent for explainable sequential recommendation. IEEE Access.

[15] Zhang, C., & Wang, Y. (2024). A data-driven multi-objective production scheduling framework for battery pack manufacturing under material arrival uncertainty. AI and Data Science Journal, 1(1), 86–93.

[16] Chen, J., Liu, J., Liang, Y., & Zhou, M. (2026). HeteroGCL: A heterogeneous graph contrastive learning framework for scalable and sustainable cryptocurrency AML. Applied Sciences, 16(6), 2860. https://doi.org/10.3390/app16062860

[17] Chen, J., Liang, Y., Liu, J., & Zhou, M. (2026). Temporal transformer with conditional tabular GAN for credit card fraud detection: A sequential deep learning approach. Mathematics, 14(7), 1183. https://doi.org/10.3390/math14071183

[18] Mo, T., Zhang, C., Zou, J., Guo, Z., & Rhee, M. (2026). Self-evolving AI agents with dual memory for automated software testing and bug localization. IEEE Access.

[19] Ping, W., & Fan, H. (2023). Beyond isolated ratios: Accrual–revenue–cash flow indicators and interaction-aware learning for detecting financial reporting irregularities in U.S. public companies. World Journal of Economics and Business Research, 1(1), 46–55.

[20] Wang, J., Liu, J., Ge, Y., & Jiang, B. (2023). Risk-aware ensemble learning for credit default prediction under economic uncertainty. World Journal of Economics and Business Research, 1(1), 18–32.

[21] Wang, B., Zhao, W., & Wang, Z. (2024). DAS-GNN: A scalable disagreement-aware graph neural framework for anomaly detection in large-scale networks. AI and Data Science Journal, 1(1), 67–75.

[22] Chen, J., Han, X., Zhang, X., & Yang, Y. (2023). Exact concept-level explanations for early detection of financial misstatements. Eurasia Journal of Science and Technology, 5(2), 34–48.

[23] Teng, D. (2025). RSRBench: Reproducibility and performance benchmarking for regional seismic risk workflows. Academic Journal of Architecture and Civil Engineering, 3(1), 40–52.

[24] Fang, Q. (2024). Bias-aware large language models for equitable educational feedback and assessment. Trends in Social Sciences and Humanities Research, 2(1), 41–50.

[25] Wang, Y., Bian, M., & Lin, Y. (2022). An explainable temporal graph neural network for disruption prediction and risk propagation in multi-tier supply chains. Journal of Computer Science and Electrical Engineering, 4(1), 17–25.

[26] Mai, N. T., Fang, Q., & Cao, W. (2025). Measuring student trust and over-reliance on AI tutors: Implications for STEM learning outcomes. International Journal of Social Sciences and English Literature, 9(12), 11–17.

[27] Feng, E., Li, P, Mo, T., & Wang, G. (2022). PRISM: Prediction-guided risk-aware deep reinforcement learning for energy-aware dynamic resource scheduling in distributed cloud data centers. Journal of Computer Science and Electrical Engineering, 4(1), 53–61.

[28] Zhang, F., & Yang, J. (2024). Robustness-aware model compression for multimodal perception under challenging conditions. AI and Data Science Journal, 1(1), 103–111.

[29] Jiao, Y., Fan, H., Yue, X., Ping, W., Sun, T., & Wang, J. (2026). Dynamic heterogeneous graph contrastive learning for uncovering collusive financial fraud. Scientific Reports.

[30] Jiao, Y., & Liang, Y. (2024). An integrated process-mining and explainable machine-learning framework for bottleneck and internal-control risk detection in the multi-entity financial close. Social Science and Management, 1(3), 96–107.

[31] Fang, Q., & Liu, W. (2025). HARLA-ED: Resolving information asymmetry and enhancing algorithmic symmetry in intelligent educational assessment via hybrid reinforcement learning. Symmetry, 18(1), 58. https://doi.org/10.3390/sym18010058

[32] Qin, Y., Rhee, M., Teng, D., Zhang, C., & Zi, B. (2026). Neuro-symbolic constraint verification for LLM-driven internet finance transaction execution. Scientific Reports.

[33] Wang, J., Tan, Y., Jiang, B., Wu, B., & Liu, W. (2025). Dynamic marketing uplift modeling: A symmetry-preserving framework integrating causal forests with deep reinforcement learning for personalized intervention strategies. Symmetry, 17(4), 610. https://doi.org/10.3390/sym17040610

[34] Wang, G., Li, P., Feng, E., & Mo, T. (2024). Uncertainty-aware LLM-as-a-judge evaluation for reliable search and recommendation systems: Calibrated confidence and impact-aware selective human review. AI and Data Science Journal, 1(1), 112–121.

[35] Fang, Q. (2024). TRUE-EWS: Trustworthy early warning with equalized conformal guarantees for equitable student support in U.S. education. Educational Research and Human Development, 1(2), 44–54.

[36] Zhang, F., & Yang, J. (2022). Accuracy-aware model compression for real-time visual perception under resource constraints. Journal of Computer Science and Electrical Engineering, 4(1), 45–52.

[37] Shen, A., & Liu, W. (2026). Budget-aware adaptive test-time compute with hierarchical self-verification for semantic ID generative recommendation. Computers.

[38] Zhou, F. (2026). Moment-constrained side-force allocation for wind-assisted merchant ships: Hydrodynamic performance of retractable anti-leeway fins and flap rudders. Innovation and Technology Studies, 3(1), 27–41.

[39] Shen, A., & Huang, S. (2022). A systems-aware framework for intelligent large-scale recommendation models. Journal of Computer Science and Electrical Engineering, 4(1), 36–44.

[40] Wang, J., Liu, J., Zheng, W., & Ge, Y. (2025). Temporal heterogeneous graph contrastive learning for fraud detection in credit card transactions. IEEE Access.

[41] Zou, J., Qin, Y., & Rhee, M. (2024). Beyond model scale: A task-aware reliability benchmark for large language models in clinical decision support. Innovation and Technology Studies, 1(1), 39–47.

[42] Mai, N. T., Cao, W., & Fang, Q. (2025). A study on how LLMs (e.g., GPT-4, chatbots) are being integrated to support tutoring, essay feedback and content generation. Journal of Computing and Electronic Information Management, 18(3), 43–52.

[43] Fan, H., Jiao, Y., Wang, M., Chen, J., & Yang, S. (2026). Fraud learns too: Continual graph learning under strategic adversarial drift in dynamic networks. Scientific Reports.

[44] Shang, W., & Chen, T. (2024). PrefixKD-Edge: Real-time locomotion-intent prediction with prefix self-distillation on lightweight temporal networks. Innovation and Technology Studies, 1(1), 30–38.

[45] Sun, T., Yang, J., Li, J., Chen, J., Liu, M., Fan, L., & Wang, X. (2024). Enhancing auto insurance risk evaluation with transformer and SHAP. IEEE Access, 12, 116546–116557. https://doi.org/10.1109/ACCESS.2024.3446127

[46] Shen, A. (2023). Compute-efficient scaling of large sequential recommendation models under dynamic user behavior. World Journal of Information Technology, 1(1), 62–71.

[47] Rhee, M., Zou, J., & Qin, Y. (2024). CLARITY: A closed loop framework for duplicate aware failure prediction and budgeted automated remediation in cloud infrastructure. AI and Data Science Journal, 1(1), 76–85.

[48] Ping, W., & Fan, H. (2024). Disclosure quality and valuation risk in Level 3 investments: A disclosure-implied risk framework for U.S. business development companies. Social Science and Management, 1(1), 78–87.

[49] Qin, Y., Zou, J., & Rhee, M. (2022). An AI-driven intelligent transaction execution framework for internet finance platforms: Cost-sensitive real-time integrity interdiction. Eurasia Journal of Science and Technology, 4(2), 36–44.

[50] Zhu, R., & Yue, X. (2024). An explainable machine learning framework for predicting dividend sustainability and financial risk of US REITs under interest-rate shocks. Journal of Trends in Finance and Economics, 1(2), 48–57.

[51] Lin, Y., Wang, Y., & Bian, M. (2024). XOR-EW: An explainable and cost-aware artificial intelligence framework for operational risk early warning in financial institutions. AI and Data Science Journal, 1(1), 94–102.

[52] Guo, Z., Chen, T., & Shang, W. (2022). Class-conditional intermediate-domain adversarial adaptation for cross-device acoustic scene classification. Journal of Computer Science and Electrical Engineering, 4(1), 26–35.

[53] Chen, J., Zhang, X., & Han, X. (2023). Explainable identification of tax-related financial reporting anomalies in U.S. public companies: Cross-fitted tax-consistency residuals and Owen-value SHAPley propagation. World Journal of Economics and Business Research, 1(1), 33–45.

[54] Li, P., Feng, E., Wang, G., & Mo, T. (2024). Hybrid deep reinforcement learning and integer optimization for resource scheduling in heterogeneous AI computing clusters. Innovation and Technology Studies, 1(1), 48–56.

[55] Teng, D., Rhee, M., Qin, Y., Zi, B., & Liu, W. (2026). SW-SpeedDLM: Sliding window speculative decoding for diffusion language models under long context constraints. Mathematics, 14(12), 2137. https://doi.org/10.3390/math14122137

[56] Chen, T., Shang, W., & Guo, Z. (2024). XSLA: An SLA-aware heterogeneous XPU scheduling framework for edge AI inference. World Journal of Information Technology, 2(2), 48–59.

[57] Zhang, H. (2024). Graph-aware multi-task learning for early prediction of timing violations and routing congestion in digital IC physical design. World Journal of Information Technology, 2(1), 13–20.

[58] Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. World Journal of Management Science, 2(1), 63–72.

[59] Liang, Y., & Jiao, Y. (2024). Real-time financial fraud detection using a feature-light temporal dynamic graph neural network with contrastive learning. Journal of Trends in Finance and Economics, 1(1), 76–84.

[60] Guo, Z., Fang, Q., Yang, J. S., & Rhee, M. (2026). Closed-loop audio neuroregulation for ADHD: Graph-based attention decoding and rhythm entrainment. IEEE Access.

[61] Yue, X., Yang, J., & Liu, W. (2026). FraudDebate-Agent: A multi-agent LLM framework with an evidence-based debate mechanism for financial statement fraud detection. Mathematics, 14(15), 2695. https://doi.org/10.3390/math14152695