Tracing Failures through Plant Topology: Propagation Reasoning by a Knowledge-Graph-Grounded LLM Agent

Muhamad Imannulhakim
PT. Pupuk Indonesia
m.imannul@pupuk-indonesia.com

Abstract

Diagnosing anomalies in process plants requires reasoning about how a failure at one unit propagates through interconnected equipment — knowledge that is scattered across engineering documents, and that unconstrained large language model (LLM) agents tend to fill with plausible but unsupported conclusions. We present an LLM agent, being piloted at an operating ammonia plant, whose domain memory is a knowledge graph of the plant's process: a topology backbone translated from the process flow, enriched with knowledge from over 30 engineering documents. We evaluate its propagation reasoning against the graph itself using 46 generated probes, each run three times. Entity grounding holds at every depth: all 565 failure-mode attributions are correct, and the agent rejects every question that attaches a failure mode to the wrong unit. Mean path recall falls from 1.00 at two hops to 0.60 at eight, and 40% of eight-hop answers deny a route the graph records. Scoping the same tools to the topology backbone, with model, prompt and graph unchanged, raises eight-hop recall to 0.81 and halves these denials, suggesting that unscoped neighbourhood queries struggle to reconstruct long process routes. Retrieval over the source documents matches the agent up to six hops but not at eight.

1 Introduction

A process plant is a tightly coupled network: a fault at one unit reaches others through shared process streams, heat integration and protective interlocks. Diagnosing an anomaly therefore means following propagation paths, using knowledge scattered across process diagrams, failure-mode libraries and maintenance records. Large language models (LLMs) can read and connect this material, but they also produce fluent claims their sources do not support (Ji et al., 2023), which is a serious risk when a wrong diagnosis leads to a wrong maintenance action. Grounding LLMs in knowledge graphs (KGs) is a widely studied mitigation (Agrawal et al., 2024; Pan et al., 2024), and tool-calling agents (Yao et al., 2023) can explore a KG step by step. KG–LLM combinations have been applied to industrial fault diagnosis (Ma et al., 2025), but a recent systematic review finds the field concentrated on single machines and short of evaluation metrics (Razaq et al., 2026).

Two problems make plants harder. Propagation is a property of *topology*: what matters is not only which failure modes a unit has, but how an effect travels across the plant. And deployed diagnostic systems rarely have field-verified labels, since confirming a hypothesised failure usually means opening the machine. We address both by encoding the plant's topology explicitly, which lets the graph serve as its own ground truth.

All results characterise the deployed architecture — the LLM together with its graph tools, prompt and output contract — rather than the LLM in isolation. The benchmark tests whether the agent can trace the route a failure would follow and report it faithfully. Our contributions are:

- **A KG design for process plants** that separates a topology backbone from a document-derived knowledge layer, so that an agent can reason about how failures propagate rather than only retrieve documents.
- **A graph-grounded benchmark** that decomposes propagation reasoning into six checks — path fidelity, direction integrity, intervention recall, inference-boundary placement, entity grounding and termination — verified against the graph in Cypher, with no LLM judge; direction and the protection controls are adjudicated by hand.
- **A diagnosis of the depth failure**: scoring entity grounding apart from path fidelity shows

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 131–137
November 10, 2026 ©2026 Association for Computational Linguistics

that the agent loses long routes without inventing entities, and an ablation that scopes traversal to the topology backbone recovers about half of the loss, pointing to the tools' traversal rather than the model. Against retrieval over the source documents and over the graph's nodes as text, the agent's advantage appears at depth and is largest with scoped traversal.

2 Related Work

LLMs reasoning over knowledge graphs. Grounding LLMs in KGs is a widely studied way to reduce unsupported output (Agrawal et al., 2024; Pan et al., 2024). Some methods give the model interfaces to structured data and let it read iteratively before reasoning (Jiang et al., 2023). Others treat the LLM as an agent that explores the graph step by step (Sun et al., 2024), building on interleaved reasoning and tool use (Yao et al., 2023), or have it generate relation paths as plans and retrieve matching paths as evidence (Luo et al., 2024). These methods are evaluated on open-domain KG question answering with gold answers. Our setting differs in three respects: the graph is a private, engineering-authored model of a single plant; the paths that matter follow process flow; and the diagnostic questions users ask have no gold answers. The concern we study applies to any KG that layers rich annotation over a structural backbone, although we test it on one graph only.

KG–LLM fault diagnosis in industry. Process-systems engineering has long modelled fault propagation as a causal graph over plant variables, such as signed directed graphs, and diagnosed faults by searching it (Venkatasubramanian et al., 2003; Palmer and Chung, 2000); our topology backbone plays a similar role, but an LLM agent traverses it. Ma et al. (2025) construct a fault-diagnosis KG to strengthen LLM reasoning for diagnosis and maintenance decision support. A systematic review of KG–LLM fault diagnosis finds the field concentrated on rotating machinery, steel and CNC equipment, with gaps in evaluation metrics and in explainable reasoning routes (Razaq et al., 2026). We address a process plant, where diagnosis depends on connections between units rather than on a single machine, and we evaluate the agent specifically on those connections.

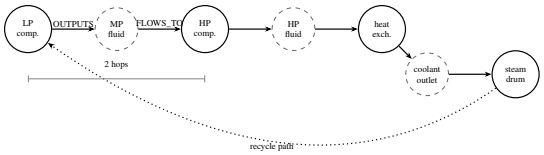


Figure 1: The topology backbone, shown schematically with generic equipment names. LP, MP and HP denote the low-, medium- and high-pressure stages.

3 Methodology

3.1 Knowledge Graph

The graph has a *topology backbone* and *knowledge layers*. In the backbone, derived from piping and instrumentation diagrams, process flow diagrams and process manuals, equipment items and process streams are both nodes, linked as `Equipment` $\xrightarrow{\text{OUTPUTS}}$ `ProcessStream` $\xrightarrow{\text{FLOWS_TO}}$ `Equipment`. Modelling streams as nodes gives an exchanger distinct outlets that lead to different parts of the plant, and because every link is directed, recycle and heat-integration loops are explicit rather than ambiguous (Figure 1). A route between two units is an alternating path of equipment and streams, so a failure's reach is measured in hops. Knowledge layers derived from over 30 engineering documents attach failure modes and causes, protective interlocks, instrumentation, maintenance tasks and past investigations to the backbone (Table 1). The graph holds 1,219 nodes and 1,962 relationships in Neo4j. The graph and documents are proprietary and cannot be released.

3.2 Agent

The agent is a tool-calling LLM (DeepSeek V4 Flash) (Yao et al., 2023) that reaches the graph only through read-only tools served over the Model Context Protocol (MCP): resolution and failure-chain tools, a Cypher tool, and two neighbourhood tools for propagation. `explore` returns everything within k hops of one node (k=2 by default) and `propagation_rca` (root cause analysis) does the same around a target (k=3) with summaries of its failure data and protective wiring. Both follow every relationship in either direction, up to k=8 and 150 returned nodes, so a neighbourhood spans the knowledge layers and grows far faster than the topology (Table 2). When a neighbourhood exceeds the cap, the prompt tells the agent to retry with a smaller k rather than reason from a partial result. Raising the cap is not a real op-

Layer	Node types (count)	Key relationships
Topology backbone	Equipment (158), ProcessStream (169), Stage (16)	OUTPUTS, FLOWS_TO, HAS_COMPONENT
Failure	FailMode (76), Cause (148), Event (3), Concept (13)	HAS_FAILMODE, HAS_CAUSE, CAN_LEAD_TO
Protection	Trigger (30), Interlock (8), Action (18)	INDICATES, TRIGGERS, ACTIVATES
Instrumentation	MeasurementPoint (141), Tag (188)	MONITORS, INSTRUMENTED_AS
Maintenance & procedures	MaintenanceTask (23), MonitoringTask (14), ProcedureStep (87)	MITIGATED_BY, VERIFIES, NEXT
Investigation history	RCADocument (5), AuditReport (1), Recommendation (57)	IDENTIFIES, RECOMMENDS, ADDRESSES

Table 1: Knowledge-graph layers and node counts. Names are as stored in the graph; the evaluation relies on HAS_FAILMODE (a unit's recorded failure modes), CAN_LEAD_TO (one failure mode causing another) and the protection chain, in which a Trigger indicates a failure mode and trips an Interlock that activates an Action. RCA: root cause analysis.

k	All relationships	Process flow only
2	27 / 71	4 / 5
3	118 / 346	7 / 9
4	290 / 685	9 / 12
6	750 / 1,088	15 / 23
8	1,095 / 1,156	25 / 40

Table 2: Nodes reached within k hops of the ten eight-hop source units (mean / max), when the walk follows every relationship in either direction or only downstream process flow (OUTPUTS, FLOWS_TO).

tion in deployment, since an unscoped eight-hop neighbourhood holds most of the graph, far more than can be passed to the model in one call. We therefore evaluate two configurations of the agent: *deployed*, with the tools as described, and *scoped*, which adds one option to both tools that restricts the walk to process flow, downstream or upstream along OUTPUTS/FLOWS_TO, described only in the tool descriptions; model, prompt, graph and defaults are unchanged. The scoped configuration is analysed as an ablation (§4.3).

For "if a failure occurs at A, what happens to B?", the prompt has the agent walk the topology from A to B and end every factual sentence with a typed citation: *graph* for a node it relies on, or *inference* for its own reasoning, also listed in an unverified field. Where the graph has no data or no route, it must say so rather than fill the gap.

3.3 Evaluation

Because the topology is explicit, routes, hop distances, direction and target failure data are all queryable, so answers can be checked against the graph without field labels. We generated 46 probes and ran each three times, each in a fresh conversation: 40 propagation probes, ten at each of two, four, six and eight hops, mixed on whether the target has failure data; and six direction probes asked in reverse, three with no route that way and three on recycle loops, each also attaching a failure mode to a unit that does not carry it. Answers are scored on the metrics in Table 3 in Cypher, with no LLM judge; direction probes are adjudicated by hand. For P3, we use 15 separate protection probes, run once, that name a failure mode and its unit, with no target, and ask which trip, interlock or action would stop it: nine on failure modes wired to an interlock node, and six on unprotected failure modes.

Baselines. We compare the agent with two retrieval-augmented generation (RAG) baselines that answer in a single call without tools, using the same model and a prompt that asks for the same output format. *Document RAG* uses no graph at all: 13 of the source documents the graph was built from are cut into 250-word chunks with a 50-word overlap, and the ten chunks that best match the question under BM25 form the context, about 2,500 words. The remaining sources are mostly diagrams, such as fault trees and logic diagrams, whose content is not ideal for text retrieval. The second RAG (*Node RAG*) keeps the graph's content but not its structure: every node, across all layers, is indexed as a short text document (its name, type and relationships), and the 20 best BM25 matches form the context. Each baseline runs three times on all propagation probes.

4 Results and Discussion

4.1 Comparison with RAG Baselines

Figure 2 compares path recall across the agent and both baselines. Process streams are named only in the graph, so a system answering from documents cannot name them; this comparison therefore scores equipment only, counting an item as named when the answer gives its tag or name, and leaves out two-hop routes, which have no intermediate equipment. These values are therefore not comparable with the full P1 in §4.2.

Up to six hops, document RAG matches the de-

	Metric	Gold
P1	**Path fidelity**: recall of intermediate nodes, for the shortest path and for the best-matching valid route of at most eight hops; precision: the share of equipment and streams an answer names, other than the endpoints and their immediate neighbours, that lie on a valid route; and their F1	`shortestPath` and full route enumeration over `OUTPUTS/FLOWS_TO`
P2	**Direction integrity** on reverse-pair probes	One-way vs. recycle pairs; human-adjudicated
P3	**Intervention recall**: the trip, interlock or action that would interrupt propagation; on unprotected controls, no safeguard may be claimed	`Trigger`→`Interlock`→`Action` wiring; controls human-adjudicated
P4	**Inference boundary**, two-sided. *Recall*: if the target has no failure data, consequences there must be declared as inference. *Precision*: if it has failure data, at least one recorded failure mode must be asserted as graph fact	`HAS_FAILMODE` at target
P5	**Entity grounding**: citation keys declared graph-backed that resolve to no node; and failure-mode attributions ("equipment E has failure mode F") that the graph does not record	Node resolution; `HAS_FAILMODE`
P6	**Termination**: cited equipment and streams stay within the source–target span	Path membership
—	**False absence claim**: the answer states that no route exists when the graph records one. Not automatic; counted by reading answers	`shortestPath` over `OUTPUTS/FLOWS_TO`

Table 3: Evaluation metrics. P1, P4 and P5 are scored automatically in Cypher; P2 and the P3 controls are human-adjudicated; P6 raises flags for review.

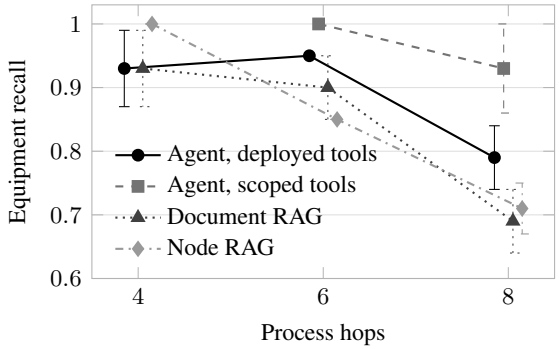


Figure 2: Path recall on equipment only, by hop distance, mean ± s.d. over three runs (n=10 probes per band).

ployed agent (0.93 and 0.90, against 0.93 and 0.95). At eight hops it falls to 0.69 against 0.79, while the scoped agent (§3.2) reaches 0.93. Despite the high recall, document RAG denies the route in 6, 12 and 16 of 30 answers at four, six and eight hops, against 6, 7 and 12 for the deployed agent (Table 5). Node RAG traces the route in all two-hop answers and 27 of 30 four-hop answers but in none of the 60 six- and eight-hop answers, so its recall there (0.85 and 0.71) comes entirely from units named while denying the route. The graph's advantage is thus clearest at depth with scoped traversal.

4.2 Agent Results

These results describe the deployed agent with the tools of §3.2. Figure 3 shows path recall, precision and F1 by hop distance as the mean ± standard deviation of the three per-run band means.

Check	Run 1	Run 2	Run 3
No failure data at target → inference	16/17	17/17	17/17
Failure data at target → stated as fact	23/23	23/23	21/23
Boundary correct (both)	**39/40**	**40/40**	**38/40**
Answer hedges something	39/40	39/40	40/40

Table 4: Inference boundary (P4) on the 40 propagation probes.

Path fidelity (P1). Recall against any valid route stays within 0.03 of recall against the shortest path in every band. Precision stays between 0.75 and 0.88 in every band, so the loss at depth is missed route, not padding. It is lowest at two hops, where the route has a single intermediate stream, so every extra unit an answer mentions as context counts against it. Band means are stable across runs (s.d. ≤ 0.08), but in terms of stability, a probe's recall is identical in all three runs for only 21 of the 40 propagation probes.

Inference boundary (P4). The boundary is placed correctly in nearly every answer (Table 4). Almost every answer marks some claim as inference, so passing the first row alone would prove little. The second row rules this out: when the graph records the target's failure modes, the agent states them as fact.

Entity grounding (P5). Each direction probe attaches a failure mode to a unit that does not carry it, and all 18 answers (six probes, three runs) re-

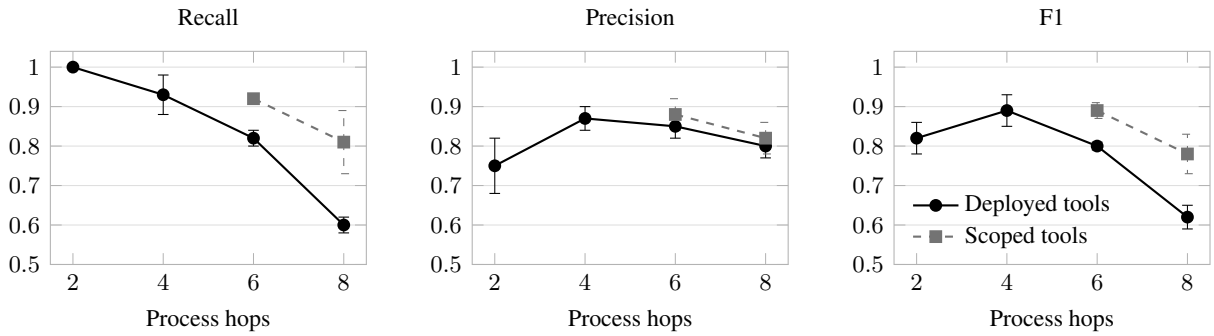


Figure 3: Path fidelity (P1) by hop distance, mean ± s.d. over three runs per condition (n=10 probes per band).

Hops	Run 1	Run 2	Run 3	Scoped (3 runs)
2	0/10	0/10	0/10	—
4	2/10	2/10	2/10	—
6	3/10	3/10	1/10	3/30
8	5/10	4/10	3/10	6/30
All	**10/40**	**9/40**	**6/40**	**9/60**

Table 5: Answers that deny a route the graph records. Runs 1–3 use the deployed tools; the scoped column pools three runs of the ablation (§4.3).

ject this false premise and name the correct owner. Every sentence tying a failure mode to a unit was matched against `HAS_FAILMODE`, and all 565 such attributions(across 198 answers: three deployed runs of all 46 probes and three scoped runs of the 20 six- and eight-hop probes) are correct. All 561 graph-backed citation keys also resolve to a node in the graph

False absence claims. We read every propagation answer and counted those that deny a process route the graph records; saying only that no *failure* link (`CAN_LEAD_TO`) exists is true and not counted. Denials grow with distance (Table 5) and recur on the same probes across runs. Most denials fall on routes that pass through a single node joining separate parts of a unit, such as a furnace's process and convection sections. Document RAG (§4.1) denies the same routes, so some denials may reflect how the graph models these units rather than an agent error.

Direction (P2), intervention (P3) and termination (P6). All 18 direction answers (six probes, three runs) are correct: the agent denies a route on the three one-way pairs and traces the loop on the three recycle pairs in every run. On the protection probes, the agent named the wired interlock, its trigger and its actions in all nine protected cases, and for the six unprotected failure modes stated that no protection is recorded rather than offering a nearby one. One control answer links the failure to protection on a related failure mode without marking the link as inference. P6 flags one eight-hop probe in the first run for citing a node outside the source–target span.

4.3 Why Long Routes Are Lost

Grounding holds, paths do not. From two to eight hops, path recall falls by 40 points while failure-mode attributions stay correct. The agent does not invent equipment as it loses the route; it makes false claims *about* the graph, such as denying a route that exists.

The layers bury the route. Because the tools follow every relationship, a neighbourhood floods into the knowledge layers (Table 2), and for eight of the ten eight-hop pairs even the smallest one that holds the whole route exceeds the 150-node cap. The agent is therefore told when a result is partial, but the prescribed recovery, a smaller k, moves a distant target further out of reach.

Ablation. With the scoped configuration (§3.2), over three reruns of the six- and eight-hop probes (Figure 3), eight-hop recall rose from 0.60 to 0.81 ± 0.08, with every scoped run (0.73–0.89) above every deployed run (0.57–0.61), and six-hop recall from 0.82 to 0.92. The gap between six and eight hops halves, from 0.22 to 0.11. Precision holds (0.80 to 0.82 at eight hops), so eight-hop F1 rises from 0.62 to 0.78 without answers padding the route. False absence claims at eight hops fall from 12 of 30 answers to 6 of 30 (Table 5).

Recycle loops. In six of the ten eight-hop pairs, a heat-integration or recycle loop also links the target back to the source, and these pairs lose the most recall (Table 6). On one-way pairs alone, recall

Hops	Pair connected	Deployed	Scoped
8	both ways (n=6)	0.51	0.75
8	one way (n=4)	0.74	0.90
6	both ways (n=5)	0.80	0.85
6	one way (n=5)	0.84	0.99
4	both ways (n=4)	0.86	—
4	one way (n=6)	0.98	—
2	both ways (n=5)	1.00	—
2	one way (n=5)	1.00	—

Table 6: Path recall for pairs that a recycle loop also connects in reverse versus one-way pairs, pooled over three runs per condition. The scoped ablation covers six and eight hops only.

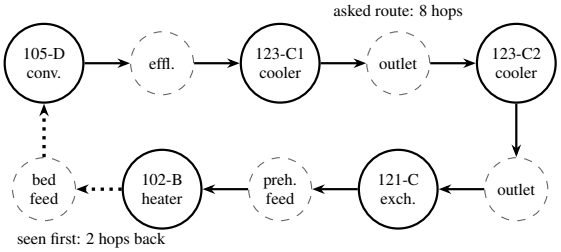


Figure 4: P-047, where a failure at converter 105-D is asked to reach start-up heater 102-B. The route runs eight hops through the effluent coolers and feed/effluent exchanger (solid), but 102-B also feeds 105-D directly (dotted). The agent found the short reverse edge, concluded that 102-B lies upstream, and reported no route.

drops from 1.00 at two hops to 0.74 at eight. Loops make it worse, pulling loop pairs down to 0.51. A typical failing answer finds the short link from target back to source first, decides that the target is upstream, and stops searching (Figure 4). Scoping improves both groups but does not close the gap.

5 Conclusion

We presented an LLM agent for an operating ammonia plant whose domain memory is a knowledge graph built around process topology, with a benchmark that checks propagation reasoning against the graph itself, without field labels. The agent's claims about entities hold: every failure-mode attribution is correct, and it rejects questions that place a failure mode on the wrong unit. Its claims about routes do not: recall falls from 1.00 at two hops to 0.60 at eight, and 40% of eight-hop answers deny a route the graph records. Because grounding and path fidelity are scored separately, this failure is visible even though no entity is invented. Restricting the same tools to the topology backbone recovers about half of the lost recall and halves the false denials, which points to the tools' traversal rather than the model; recycle loops remain the main source of error. Retrieval over the source documents matches the agent up to six hops, so the graph's advantage lies in long routes with scoped traversal and in answers that can be checked against it. Two practical lessons follow: model process topology explicitly so that propagation reasoning can be checked against the graph, and let traversal tools stay on that backbone instead of walking every layer at once.

Limitations

We measure consistency with the graph, not diagnostic correctness: a graph-consistent answer can still be wrong where the graph is. The baseline comparison scores equipment only, and grounding (P5) and the inference boundary (P4) apply to graph-backed answers alone. Results cover one plant, one graph and one model with limited probes and no confidence intervals. Probes are generated from the graph rather than collected from users. For a failure mode the graph does not record, the agent falls back on the model's general knowledge and marks it as inference, and such answers carry no graph guarantee.

Declaration on Generative AI

The system evaluated in this paper is itself an LLM agent (§3.2). Generative AI assistants were also used in building the knowledge graph, where they helped the author compare the content already ingested into Neo4j with the parts of the process topology not yet added. They were further used in preparing the paper: drafting and editing text, checking bibliographic details, writing the probe generation and scoring scripts, and reviewing agent responses. All numbers, adjudications and claims were checked by the authors, who take full responsibility for the graph and the paper.

References

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Can knowledge graphs reduce hallucinations in LLMs? : A survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3947–3960, Mexico City, Mexico.

Z. Ji, N. Lee, R. M. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. 2023. Survey of

hallucination in natural language generation. *ACM Computing Surveys*, 55(12).

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations (ICLR)*.

Y. Ma, S. Zheng, Z. Yang, H. Pan, and J. Hong. 2025. A knowledge-graph enhanced large language model-based fault diagnostic reasoning and maintenance decision support pipeline towards industry 5.0. *International Journal of Production Research*.

Claire Palmer and Paul W. H. Chung. 2000. Creating signed directed graph models for process plants. *Industrial & Engineering Chemistry Research*, 39(7):2548–2558.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Wan Barmawi Muhammad Razaq, Hao Chen, Marcos R. Machado, and João Luiz Rebelo Moreira. 2026. How can the integration of AI large language models and knowledge graph enhance fault diagnosis? A systematic literature review. *Applied Soft Computing*, 194:114908.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations (ICLR)*.

V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri. 2003. A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. *Computers & Chemical Engineering*, 27(3):313–326.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.