WHAT DOES BINARY ACCURACY PROVE? A DIAGNOSTIC DECOMPOSITION OF VISUAL GRAPH REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models are increasingly used to read structured diagrams and reason over the relations they encode. Benchmarks typically score only the final answer. Ask a model "Is there a cycle?" and it almost always says yes. Ask "Which nodes form it?" and the cycle it names is often not in the graph. A correct answer thus cannot show whether the model recovered the diagram's structure. We use cycle detection as a diagnostic, since a named cycle can be checked against ground truth. We score four stages: which edges are extracted, whether a real cycle survives among them, whether the named cycle comes from the model's own list, and whether it is real. We test four models on cyclic graphs and trees, each drawn in a standard layout and a stress-test equal-angle ring layout. On ring drawings, models say "cycle" 84–100% of the time, yet the cycle they name is real only 4.8–44.0% of the time. Degree probes and human judgments suggest the drawings are not simply unreadable. Wrong answers follow two patterns: the real cycle never enters the extracted edge list (most common for three models), or it does and a fabricated cycle is named instead (one model). A clean text edge list restores two models fully but not the others, so extraction explains the gap only in part. A "yes" to "Is there a cycle?" does not show that the cycle a model names is real, so such errors can pass unnoticed and cause silent failures in graph-based workflows. The code and dataset are available at `https://anonymous.4open.science/r/What-Does-Binary-Accuracy-Prove-31DB`.

1 INTRODUCTION

Vision-language models (VLMs) are increasingly used to read diagrams, such as software call graphs, circuit schematics, and dependency charts, and to answer questions about how the parts connect. Most benchmarks check only the final answer: is there a cycle in this graph, or can node A reach node B? But a right answer does not show that the model read the diagram correctly. A model can say "yes, there is a cycle" and then name a cycle that is not in the drawing. Scoring only the yes/no cannot tell these cases apart. So we ask the model for the node sequence of the cycle it means, its *named cycle*, and check it against the true graph: a named cycle is correct only if it is a *gold cycle*, a cycle that is truly present in the graph notation.

This paper uses visual cycle detection as a diagnostic. A named cycle can be checked against the gold graph, so we can score whether the model listed the real edges, kept a gold cycle among them, named a gold cycle, and said "yes" for the right reason. We test four models on 500 cyclic graphs and 50 trees, each drawn twice: Base (Graphviz) and an equal-angle ring of the same graph (Ring). Ring is a stress test, not a claim about ordinary figures.

Findings and contributions.

1. **A four-stage score for diagram recovery** (Section 3). We check which edges the model lists (*extraction*); whether a real cycle remains once we delete its false edges, i.e. whether the right answer was available (*gold-cycle retention*); whether the named cycle comes from its own list (*list faithfulness*); and whether it is real (*verification*). A correct yes/no, or a self-reported edge list, is not enough.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

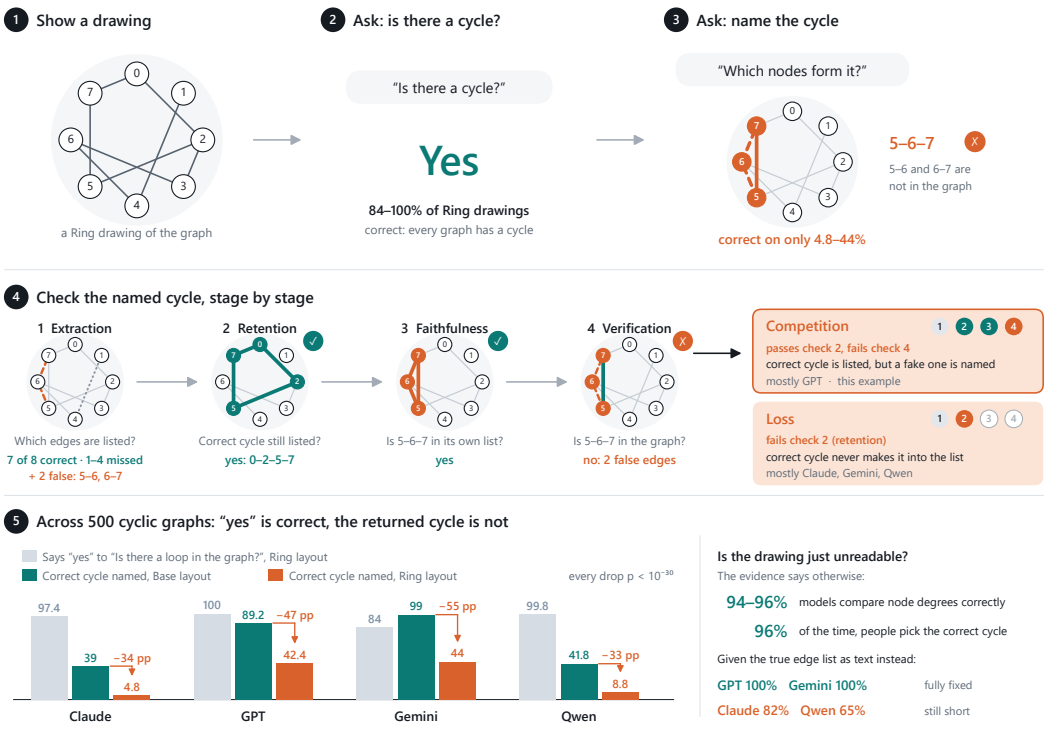


Figure 1: Overview of our work. (1)–(3) We show a Ring drawing of a graph, ask whether it contains a cycle, and then ask the model to name the cycle. Models almost always say "yes", which is correct because every graph is cyclic, but the named cycle is a gold cycle on only 4.8–44% of Ring drawings; here the named cycle 5–6–7 uses two false edges. (4) The four-stage diagnostic on the same example: extraction lists 7 of 8 gold edges (1–4 missed) plus 2 false ones; a gold cycle is retained; the named cycle is faithful to the list; verification fails. A wrong named cycle is either *competition* (check 2 passes, check 4 fails: a gold cycle is listed but a fake cycle is named; mostly GPT, and this example) or *loss* (check 2 fails: no gold cycle survives among the listed gold edges; mostly Claude, Gemini, and Qwen). (5) Per model on the 500 cyclic graphs (ASK): how often it says "yes" on Ring (grey) and names a gold cycle on Base (teal) and on Ring (orange); all four Base→Ring drops have $p<10^{-30}$ (Appendix B). Right: controls show that the picture is not simply unreadable, and a clean text edge list restores GPT and Gemini fully but not Claude or Qwen. The graph is an illustrative toy ring; Appendix Figure 5 shows a real case. Numbers in the figure are rounded; exact values are in Tables 2 and 7.

2. **"Yes" is not a verified cycle** (Section 5). Models answers "yes" on almost every cyclic graph and also return a node list, but Ring drops the named-cycle accuracy 33–54 points for every model. On Ring trees (no cycle), three of four models still say "cycle" about 95% of the time, so balanced accuracy is near chance for Claude, GPT, and Qwen (Table 2).

3. **The failure is not simply an unreadable picture** (Section 7.1). Degree comparison is 94–96% correct on the same Ring drawings, and people pick the printed gold cycle against a printed fake on 96.2% of the 500 drawings.

4. **Two failure types, scored on the same call** (Section 6). *Loss*: no gold cycle survives among the listed gold edges (Claude, Gemini, Qwen). *Competition*: a gold cycle is listed, but a non-gold cycle, usually a fake one, is named (GPT). Losing the gold cycle from the list explains 15% (GPT) to 54% (Gemini) of the Base→Ring accuracy drop; the rest comes from naming a wrong cycle while a gold one is still listed (Figure 2; Appendix I).

5. **A clean text edge list splits the models in two** (Section 7.3). GPT and Gemini reach 100%; Claude and Qwen reach only 81.8% and 65.2%.

6. **Models prefer the shortest cycle** (Section 7.4). Cycle detection does not ask for the shortest cycle, yet on their own picture-based edge lists the named cycle is a shortest cycle of the list 68% of the time (Appendix G). On typed lists, one added false edge redirects the answer in 61–100% of cases when it makes a shorter cycle, versus 2–14% when it makes a longer one (Appendix R). The validity of the shorter cycle reveals an inherent preference without a concrete cause.
7. **What geometry does and does not explain** (Section 6.3). A score from the crossings, nodes, and edges visible on the drawing, D_{fin}, orders Base above Ring (held-out AUC 0.889; Appendix K). That is a ranking of conditions in this particular setting.

2 Related Work

To our knowledge, prior work scores visual graph reasoning, but not whether a gold cycle entered the model's own list and then lost to a fake one. VisionGraph scores edge recognition as a separate sub-task and, for its commercial models, also grades a named path or cycle, not just yes/no (Li et al., 2024). GITQA supplies the graphs and Graphviz drawings we use (Wei et al., 2024; Wei, 2024). VGCure, the Visual Graph Arena, and TraversalBench cover more tasks or localize visual-continuation errors (Zhu et al., 2025; Babaiee et al., 2025; Petrova et al., 2026). Layout and encoding are not interchangeable: Graphviz style, circular versus spring drawings, and image versus text all change reported answers (Wei et al., 2026; Brand et al., 2025; Das et al., 2024). Fluent traces can be unfaithful to the computation that produced them (Turpin et al., 2023; Lanham et al., 2023), and MLLMs hallucinate structure in rendered graphs (Fan et al., 2026). The loss/competition split is built to make that list-then-choice check. A longer discussion is in Appendix X.

3 Diagnosing Visual Graph Reasoning

The gold graph $G^\star = (V^\star, E^\star)$ is the drawing's ground truth. The model's listed edges are a graph $\hat{G}$. All metric definitions are in Appendix A.

3.1 A four-stage diagnostic pipeline

We score four stages (Figure 1, step 4): **extraction** (the picture becomes a listed edge set), **gold-cycle retention** (we keep only listed edges that are also gold; this filter is ours, not the model's), **list faithfulness** (the named cycle is a cycle of the listed graph), and **verification** (the named cycle is a cycle of the gold graph). Writing $\hat{E}$ for the listed edges and $\hat{c}$ for the named cycle,

$$\begin{aligned} \mathrm{Gnd} &= \mathbf{1}[\hat{E} \cap E^\star \text{ still contains a gold cycle}], \\ F &= \mathbf{1}[\hat{y} = \text{no}] \ \vee \ \mathbf{1}[\hat{c} \text{ is a simple cycle of } \hat{E}], \\ V_{\text{strict}} &= \mathbf{1}[\hat{y} = \text{yes}] \ \wedge \ \mathbf{1}[\hat{c} \text{ is a simple cycle of } E^\star]. \end{aligned}$$

These are the forms for cyclic gold graphs. On trees, V_{strict} is a correct "no", and extraction is scored by cycle-status agreement Sym_X, because a tree has no gold cycle (Appendix A). A wrong named cycle is then one of two things. *Loss:* after we drop the fake edges, no gold cycle is left (Gnd_C=0). *Competition:* a gold cycle is still there, but the named cycle is not gold, usually a fake cycle of the list (Gnd_C=1, V_{strict}=0). Loss is a recall failure: no gold cycle made it into the listed gold edges. Competition is mainly a precision failure: invented edges close a competing cycle and that cycle is named. (Section 6).

3.2 Why binary accuracy is insufficient

Yes/no accuracy V_{pol} collapses all four stages to one bit. Two systems can share that bit and differ on whether a gold cycle was listed, whether the named cycle matches the list, and whether it matches the gold graph. Faithfulness F checks the model's own list, not $G^\star$ (Appendix A). On this all-cyclic set, yes/no is almost always "yes," so it cannot tell you whether the named cycle is real. The useful evidence is the gap in Table 2: said yes vs. named a gold cycle. **Therefore, a confident "yes" plus a named cycle is not by itself evidence that the named cycle is real; unchecked, such answers can cause silent failures in graph-related workflows.**

Table 1: Conditions. The first three are model calls on the picture; the next three reuse text lists; the last three are checks. Ring probes use the same Ring drawings.

Condition	Input	Output	Question it answers
ASK	Picture	Yes/no + named cycle	Does the model name a gold cycle?
EXTRACT	Picture (separate call)	Edge list	How well are edges read?
LIST+DECIDE	Picture, one call	Edge list, then yes/no + named cycle	Is the named cycle drawn from its own list? Loss or competition?
Frozen list	Own Ring EXTRACT list as text, no picture	Yes/no + named cycle	Does removing the picture change the answer?
Shuffled list	Frozen list, edge order permuted	Yes/no + named cycle	Does edge order matter?
Gold list (R_G)	Gold edge list as text, no picture	Yes/no + named cycle	Can the model decide once the list is correct?
Gold-blind DFS	Stored lists (no model call)	A cycle of the list	Would a search program that never sees the gold graph do better?
Degree probe (n=140)	Ring picture + two nodes	Higher-degree node	Can the model read local structure?
Shortest-path probe (n=210)	Ring picture + two nodes	Path length	Can the model trace paths?

4 EXPERIMENTAL SETUP

We test labelled, undirected drawings of 500 cyclic graphs (250 from the GITQA-Aug-Legacy cycle/hard split, 250 generated) and 50 connected hard trees (acyclic). Cyclic and tree results are reported separately; they are combined only in balanced accuracy (Table 2). Node ranges, the generator, decoding, sample sizes, and subset coverage are in Appendix B and Appendix C.

Each gold graph is drawn twice. **Base** is the Graphviz `dot` drawing (Ellson et al., 2002). **Ring** puts the same vertices on a regular polygon, either in node-id order (410 graphs) or in *cycle order*, walking around one gold cycle (90 graphs). Figure 4 shows Base, Ring, and `circo` on the same graph. Mean edge crossings rise from 1.9 to 53.4. Failure is not confined to that mean: cycle-order Ring drawings with only 2 crossings still collapse (Table 36, Figure 23; Appendix U). Graphviz `circo` is drawn for comparison and is not used for scoring; its crossings were not recounted on the stored image. Ring is a complete drawing of the same graph (every edge drawn, every label present). Geometry property checks for graphs are in Appendix C.

4.1 MODELS AND CONDITIONS

We evaluate four widely used frontier models: Claude Sonnet 5, GPT-5.6 Luna, Gemini 3.8-Flash, and Qwen 3.8-Flash. All conditions use structured JSON; one snapshot per model (Appendix B). Table 1 lists the conditions. Gnd_X and Gnd_C are the same measure: after the false edges are deleted from a model's edge list, is a gold cycle still in it? They differ only in which list they check. Gnd_X checks the list from the separate EXTRACT call. Gnd_C checks the list the model writes in the LIST+DECIDE call, just before it names a cycle.

Paired yes/no outcomes use two-sided exact McNemar tests, Wilson score intervals, and Holm correction. Benjamini–Hochberg control agrees with Holm on the headline tests (Appendix B).

Table 2: Cycle detection on $N{=}500$ cyclic graphs. Says "yes": sensitivity on this all-cyclic set. Names a gold cycle: the named cycle is a cycle of the gold graph. Gap: yes-rate minus named-cycle accuracy, in percentage points (pp), from exact counts (Appendix B). Balanced acc.: $(\text{sensitivity} + \text{specificity})/2$ using the 50 trees in Table 3. Wilson 95% intervals for named-cycle accuracy: Table 7.

Model	Says "yes"		Names a gold cycle		Gap (pp)		Balanced Acc.	
	Base	Ring	Base	Ring	Base	Ring	Base	Ring
Claude	100.0%	97.4%	39.0%	4.8%	↓ **61.0**	↓ **92.6**	50.0%	50.7%
GPT	99.8%	100.0%	89.2%	42.4%	↓ **10.6**	↓ **57.6**	88.9%	53.0%
Gemini	99.8%	84.0%	99.0%	44.0%	↓ **0.8**	↓ **40.0**	99.9%	74.0%
Qwen	96.4%	99.8%	41.8%	8.8%	↓ **54.6**	↓ **91.0**	54.2%	50.9%

5 Main Results

5.1 Task accuracy vs. named-cycle accuracy

The yes/no answer to whether a cycle exists stays high, while the named cycle does not. The main findings are:

- **Across models:** All four models lose 33.0–55.0 points of named-cycle accuracy when the layout changes from Base to Ring (all $p{<}10^{-30}$; Appendix B).
- **Claude:** Says "yes" on 100.0% of Base drawings and names a gold cycle on 39.0%. On Ring drawings, it says "yes" on 97.4% and names a gold cycle (cycle actually present in the graph) on 4.8% (Table 2).
- **GPT:** Says "yes" on almost every drawing in both layouts, but names a gold cycle on 89.2% of Base drawings and only 42.4% of Ring drawings.
- **Gemini:** Names a gold cycle on 99.0% of Base drawings and 44.0% of Ring drawings. It is the only model that often answers "no" on Ring drawings that contain a cycle (80 of 500).
- **Qwen:** Names a gold cycle on 41.8% of Base drawings and 8.8% of Ring drawings. It says "yes" slightly more often on Ring drawings than on Base ones.
- **Balanced accuracy:** Once the 50 acyclic graphs are counted, balanced accuracy on Ring drawings is near chance for Claude, GPT, and Qwen (Table 2).

5.2 Acyclic graphs: a high yes-rate hides three behaviors

A model that always says "cycle" would look strong on Table 2, because every graph there has a cycle. We conduct another experiment on acyclic graphs. The 50 trees have no cycle, so the right answer is always "no." They show three different behaviors (Table 3):

- **GPT** correctly says "no" on most Base trees (78.0%), but says "cycle" on 94% of trees using Ring layout.
- **Gemini** is right on every Base tree and on about two thirds of trees using Ring layout (64.0%), far ahead of the others.
- **Claude and Qwen** say "cycle" on almost every tree, even on Base drawings (100% and 88%). They still do when given only a text edge list with no picture (Claude 50/50, Qwen 45/50; Appendix S).

On Ring drawings, three of the four models say "cycle" on 94–98% of trees. So GPT's "yes" on Ring drawings tells us no more than Claude's; the difference is that GPT's bias appears only on the Ring layout. The Ring layout also leads models to invent edges that create a cycle in their edge list. This happens on 100% of trees for Claude, 96% for GPT, 72% for Qwen, and 24% for Gemini (Table 3, Ring Sym_X).

Table 3: Evaluation on 50 connected hard trees. Correct decision is no, so V_{pol}=V_{strict}. Correct "no": share of trees on which the model answers "no cycle." Sym_X: extracted graph correctly remains acyclic. Gold-cycle retention is not defined on trees, which have no gold cycle.

Model	**Base correct "no"**	**Ring correct "no"**	**Base** Sym_X	**Ring** Sym_X
Claude	0.0%	4.0%	78.0%	0.0%
GPT	78.0%	6.0%	82.0%	4.0%
Gemini	100.0%	64.0%	100.0%	76.0%
Qwen	12.0%	2.0%	72.0%	28.0%

Table 4: Visual extraction performance (EXTRACT) on separate image calls, 500 cyclic graphs. Sym_X (cycle-status agreement): listed graph has the correct cyclic/acyclic status; it is weak on Ring, where most lists have at least n edges and are therefore cyclic. Gnd_X: listed edges that match the gold graph retain a gold cycle after evaluation-time filtering.

Model	**Edge Precision / Recall**		**Cycle-Status Agr.** (Sym_X)		**Gold-Cycle Retention** (Gnd_X)	
	Base	Ring	Base	Ring	Base	Ring
Claude	0.952/0.964	0.504/0.590	99.6%	100.0%	93.2%	33.5%
GPT	0.992/0.993	0.873/0.936	99.4%	99.6%	98.3%	87.6%
Gemini	0.999/0.999	0.852/0.815	99.7%	84.4%	99.7%	61.8%
Qwen	0.963/0.946	0.684/0.695	95.8%	94.7%	89.2%	42.8%

6 WHERE VISUAL GRAPH REASONING FAILS

6.1 EXTRACTION QUALITY

Ring drawings make the listed edges hard for the models (Table 4). Exact recovery of the full edge set is rare (Table 5). The sharper question is whether a gold cycle is still among the listed *real* edges. It often is for GPT, less often for Gemini, and less often still for Qwen and Claude. A surviving gold cycle is not a clean list; it only says a gold cycle is present among the true-positive edges. In other words, the four models often produce polluted lists, which may contain a real cycle sometimes.

6.2 TWO WAYS A NAMED CYCLE GOES WRONG

Models fail in two different ways:

- **Loss:** no gold cycle made it into the model's edge list.
- **Competition:** a gold cycle is in the list, but the model names a cycle that is not gold, usually a fake one.

Loss is the main cause for Claude, Gemini, and Qwen (65–78% of their wrong answers). Competition is the main cause for GPT (81%) (Table 5; Figure 2(a)). Losing the gold cycle is mostly about how many gold edges survive, not a special failure (Appendix J; Figure 11).

What a fix would have to do. The split says where a fix would have to act: list more real edges (for loss), or stop inventing edges that close fake cycles (for competition). When a gold cycle is in the list but the answer is still wrong, GPT and Gemini usually name a fake cycle that is valid in their own list and no longer than the gold one. Claude and Qwen more often name something that is not even a cycle in their own list (Table 5). In simulation, removing the invented edges lifts GPT's predicted accuracy from 37.8% to 81.3% and Gemini's from 30.2% to 59.6% (Appendix I; Figure 10), so their competition is driven by false edges in the list. The clean gold list of Section 7.3 cannot test this, because it contains no fake cycle.

How much of the layout drop is loss? Two questions: *What kind of error is a wrong Ring answer?* That is the loss share shown in (Figure 2(a)). *Why did accuracy fall from Base to Ring?* Part of the

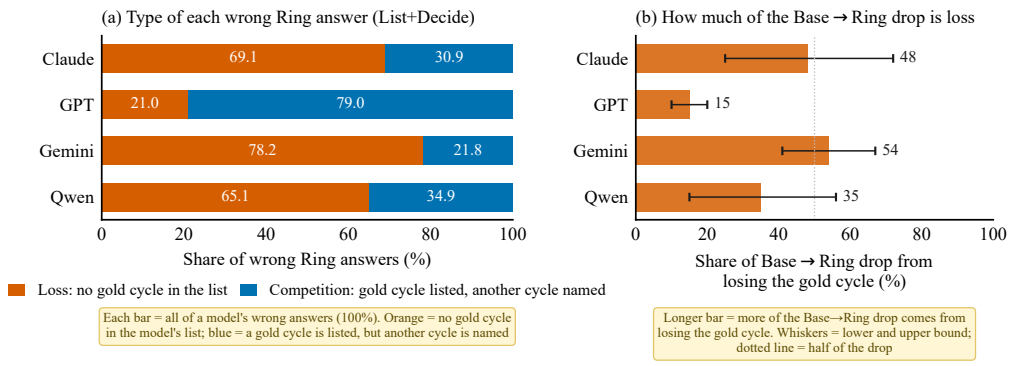


Figure 2: Two views of the same failures. (a) Composition of wrong LIST+DECIDE answers on Ring: loss (no gold cycle among the listed gold edges) vs. competition (a gold cycle is listed, a non-gold cycle is named). (b) Share of the Base→Ring drop in named-cycle accuracy that is due to the gold cycle no longer surviving in the list; whiskers are the bounds from Ring-only and Base-only weights (Table 17).

Table 5: One-call list-then-decide (LIST+DECIDE) on Ring (n=500). Exact: listed and gold edge sets match. Gnd_C: after we drop listed edges that are not gold, a gold cycle remains (this filter is ours, not the model's). $F|$yes: the named cycle is a cycle of the listed graph, among "yes" answers. Named cycle: it is a gold cycle. Competition: share of *wrong* outputs with Gnd_C=1. Named $\leq g$: among competition answers, the named cycle is a list-faithful fake cycle no longer than g, the length of the shortest gold cycle retained in the list. F=0: share of competition answers whose named cycle is not a cycle of the model's own list. Wrong-output n: Claude 470, GPT 281, Gemini 284, Qwen 441.

Model	Exact	Gnd_C	$F\|$yes	Named cycle	Competition	Named $\leq g$	F=0
Claude	0.0%	35.0%	88.5%	6.0%	30.9%	60.0%	13.1%
GPT	17.2%	87.8%	98.6%	43.8%	79.0%	79.3%	1.8%
Gemini	18.8%	55.4%	99.3%	43.2%	21.8%	100.0%	0.0%
Qwen	2.0%	42.4%	69.2%	11.8%	34.9%	48.1%	29.2%

fall is the gold cycle dropping out of the list; the rest is answers that go wrong although it is still there. The *drop share* is shown in (Figure 2(b); Appendix I):

- **GPT** (15%) and **Qwen** (35%): most of the fall happens *after* the gold cycle survived.
- **Claude** (48%) and **Gemini** (54%): the two parts are about equal.

So Qwen is mostly loss in the first view and mostly post-survival in the second: its wrong Ring answers usually lack the gold cycle, but even when the cycle survives, its accuracy falls from 43% to 14% (Table 17). Neither number is the picture-versus-clean-list gap of Section 7.3.

Available is not the same as named. A fake cycle no longer than the shortest surviving gold cycle is often *available* in the model's list, but availability is not the same as naming it (Appendix H; Figure 9). Appendix Figure 5 shows a real case: Qwen keeps the gold 4-cycle among its gold edges, also lists a false chord, and names the triangle that chord closes.

Where the errors come from. Missed gold edges carry more crossings than kept ones, and invented edges land on ring neighbours more often than chance. One reading is a *ring-neighbour prior*, a tendency to invent edges between vertices that sit next to each other on the ring (Figure 3 in Appendix A illustrates ring neighbours and this confound). For GPT and Gemini, the data are also largely consistent with a *skip-chord clearance confound*: a gold chord that skips one ring vertex passes so close to it that it can look like two neighbor edges through that vertex (Appendix G; Figure 8). Though adding tiny gaps did not improve results. Loss depends mainly on how many gold edges are listed (Appendix J); which edges are missed tracks their crossings (Appendix G).

Gemini names a shortest cycle of its own list on most of its named cycles that are valid in that list, not only the wrong ones. GPT and Qwen do so less often, and Claude less often still (Appendix G; Tables 11–13).

The four models do not share one failure.

- **Claude and Qwen** mostly lose the gold cycle on Ring. On their Base lists, a gold-blind finder (a search program that never sees the true graph) names a gold cycle more often than they do in their separate ASK call (Claude 70.4% vs. 39.0%, Qwen 74.6% vs. 41.8%; Table 28). The two numbers come from different calls, so this is a hint, not proof.
- **Gemini** also mostly loses the gold cycle on Ring, but it is the closest to an ideal shortest-cycle selector. (Appendices F, L; Figure 7.)
- **GPT** fails mainly through competition, and follows shortness only partly.

6.3 WHICH DRAWINGS FAIL

The difficulty score D_{fin} uses only what is visible on the page: the total crossing count $N_\times$, the node count n, and the edge count m (Appendix K).

$$D_{\mathrm{fin}} = \ln(1 + N_\times) + 0.35\, n - 0.25\, m. \tag{1}$$

One intercept per model turns D_{fin} into a probability of naming a gold cycle. On a held-out 20% of graphs, set aside before any fitting, the pooled AUC is 0.889 (0.78 / 0.82 / 0.93 / 0.79). GPT and Gemini stay together and above Claude and Qwen (Figure 12). The score does not identify a mechanism, but it establishes a pattern in the current setting to give an overview of when and which models may fail or be reliable.

One thing to note is that crossings alone do not account for all failures. Cycle-order rings still collapse relative to Base at a mean of only 1.9 crossings, while crossings cause missed edges in every model (Table 36; Figure 23).

7 EXPLORING ALTERNATIVE EXPLANATIONS

The collapse in Section 5 could still be unreadability, a text-level artifact, a decision failure after a correct list, or a non-causal link with length. Therefore, we test each alternative and attack our findings from different angles. Detailed writeups are in Appendices M–U.

7.1 IS RING LAYOUT AN UNREADABLE SCRIBBLE?

The Ring drawings are not simply unreadable. On the same Ring drawings, all four models answer the degree comparison (V_{deg}) correctly on 94–96% of trials, and people pick the gold cycle out of two options 96.2% of the time (Table 23). Claude and Qwen are weaker at tracing paths (V_{SP}, 45–49%), so these probes do not show that they can reliably read individual edges. However, their path-tracing rates are still far above their named-cycle rates. (Appendices M, N, O; Figures 13–16).

7.2 IS THE FAILURE A TEXT-LEVEL ARTIFACT?

Four text-level checks leave the gap in place: (i) **freezing the list as text** (the model's own listed edges fed back with the picture removed) keeps the named-cycle accuracy within 10 points of the picture call; (ii) **shuffling the edge order** does not help; (iii) **a gold-blind finder** (a search program that never sees the true graph) clearly lifts only Qwen (Gemini slightly), not Claude or GPT (Table 28; Appendix P; Figure 17); and (iv) **an explicit DFS prompt** did not differ from the frozen list.

7.3 IS THE PROBLEM READING THE PICTURE, OR REASONING ON THE LIST?

To separate the two, we hand each model the *correct* edge list as plain text, with no image, and ask the same cycle question (Appendix Q; Figure 18). **GPT and Gemini solve the task once the list**

is correct: both reach 100% named-cycle accuracy, while Claude reaches 81.8% and Qwen reaches 65.2%. **For GPT and Gemini, the picture is therefore the problem:** the picture-to-text gap is an extraction gap, reflecting the difficulty of reading the edges from the image. **Claude and Qwen retain a deficit even on clean text:** they still miss 18.2 and 34.8 points, respectively, in a single call with no image, so extraction cannot explain these residual errors.

7.4 What does one inserted edge do on a typed list?

When one added edge creates a shorter cycle, models name it. We added one false edge to a clean text list. If it made a shorter cycle, models used it in 58–100% of cases, versus 4–15% when it made a longer one. This was tested on text lists, not images, and the shorter cycle is technically valid there, so it shows how models choose, not that they are wrong (Appendix R; Figures 6, 19). Restricting to graphs that have both a shorter and a longer arm leaves the same sign.

7.5 Trees, image probes, crossing density, and artifact checks

On typed trees with one added chord, Claude and Qwen lose exact reconstruction of the longer cycle (Appendix S; Figure 20). On the unmodified picture, a per-edge present/absent probe shows a crossing effect for Claude and Qwen after correction; GPT and Gemini show none (Appendix T; Figure 21). At 41 or more crossings on the least-crossed shortest gold cycle, no model names a gold cycle (Appendix U; Figure 22; binned counts, not a manipulated sweep). Gold-cycle retention is within 10 points of a chance baseline with the same number of listed gold edges, so loss mostly tracks edge recall; looser scoring changes the size of the Base→Ring gap (smaller for GPT and Gemini, larger for Claude and Qwen) but removes it for no model (Appendix J).

8 Discussion and Limitations

A right yes/no hides two different failures: no gold cycle entered the listed gold edges, or one did and a fake cycle was named instead. Loss mostly tracks how many gold edges are listed; which specific edges are missed is a separate question (Appendix G). On messy picture lists, shortest-cycle selection describes Gemini well, GPT and Qwen partly, and Claude least. On typed lists all four follow it when a shorter cycle is planted, as the prompt licenses. Better recall would shrink loss but not GPT-style competition, which needs fewer invented edges. A pipeline that needs a correct parse should verify the named structure independently, for example against a second extraction or a symbolic parse of the source, **not against the model's own list.** The main findings remain grounded by rigorous statistical tests (Appendix B.4).

Limitations. (i) Ring is artificial. Cycle-order Ring drawings, with a mean of 2 crossings, still lose 20–35 points (Table 36), so crossings alone do not explain the collapse (Figure 3) (Appendix U). (ii) The insertion test used text; a per-edge image probe shows a crossing effect for Claude and Qwen (Appendix T). (iii) People did an easier task than the models (Appendix N). (iv) Drawings are labelled node-link graphs; GITQA size confounds any training-exposure check (Appendix W).

9 Conclusion

In this paper, we show that a "yes" to "Is there a cycle?" is not evidence that a vision-language model recovered the graph. We found that on a harder but complete layout, every model we tested keeps saying "yes" far more often than the cycle it names is real. The errors split into two kinds, a gold cycle that never reaches the model's edge list and a fake cycle that is named over a gold one, and they call for different fixes. We do not claim these results describe all vision-language models, all diagrams, or internal representations. Within this setting, the practical remedy is to verify the named structure independently (a second extraction, a symbolic parse, or the source data when available), not against the model's own answer. In conclusion, the tested VLMs cannot be fully trusted for graph based workflows.

Reproducibility Statement

Graphs, Ring layouts, stored lists, and scoring code are included in the anonymized supplementary material. Identifiers, prompts, snapshots, and recomputation details are in the appendices.

Ethics Statement

This work evaluates public models on synthetic visual graph benchmarks. The human reference (a two-option forced choice, 2AFC) used five volunteer raters and recorded only an anonymous rater code, the chosen option, a readability rating, and response time; no personal data were collected. No IRB was available to review this protocol; it is a low-risk, anonymous, non-interventional perception task on synthetic graphs with voluntary participation.

AI Use Statement

Generative AI tools were used for coding assistance, for re-checking statistics against stored outputs, for literature identification and citation checking, and for manuscript editing. All statistics were computed by the authors' analysis scripts, and all reported counts, tables, and tests were checked against the stored experimental outputs. Every cited claim was checked against the primary source. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Qihang Ai, Jiafan Li, Jincheng Dai, Jianwu Zhou, Lemao Liu, Haiyun Jiang, and Shuming Shi. Advancement in graph understanding: A multimodal benchmark and fine-tuning of vision-language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7485–7501, 2024. doi: 10.18653/v1/2024.acl-long.404. URL https://aclanthology.org/2024.acl-long.404/.

Qihang Ai, Ruizhou Li, Menghui Wang, and Haiyun Jiang. Graph-to-vision: Multi-graph understanding and reasoning using vision-language models. *arXiv preprint arXiv:2503.21435*, 2025. doi: 10.48550/arXiv.2503.21435. URL https://arxiv.org/abs/2503.21435.

Zahra Babaiee, Peyman Kiasari, Daniela Rus, and Radu Grosu. Visual graph arena: Evaluating visual conceptualization of vision and multimodal large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2081–2113, 2025. URL https://proceedings.mlr.press/v267/babaiee25a.html.

Tianyi Bai, Yuxuan Fan, Jiantao Qiu, Fupeng Sun, Jiayi Song, Junlin Han, Zichen Liu, Conghui He, Wentao Zhang, and Binhang Yuan. Hallucination at a glance: Controlled visual edits and fine-grained multimodal learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://papers.nips.cc/paper_files/paper/2025/hash/c518f504ad5894ccb264a9890f0f5544-Abstract-Conference.html.

Timo Brand, Henry Förster, Stephen G. Kobourov, and Jacob Miller. Visualization biases MLLM's decision making in network data tasks. In *VIS x GenAI Workshop at IEEE VIS*, 2025. doi: 10.48550/arXiv.2511.03617. URL https://arxiv.org/abs/2511.03617.

Yukun Cao, Shuo Han, Zengyi Gao, Zezhong Ding, Xike Xie, and S. Kevin Zhou. GraphInsight: Unlocking insights in large language models for graph structure understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12096–12134, 2025. doi: 10.18653/v1/2025.acl-long.591. URL https://aclanthology.org/2025.acl-long.591/.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 37665–37691, 2024a. doi: 10.52202/079017-1189. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/4254e856d01a5e7b7ea050477c3ef9b9-Abstract.html.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. LLaGA: Large language and graph assistant. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7809–7823, 2024b. URL https://proceedings.mlr.press/v235/chen24bh.html.

Debarati Das, Ishaan Gupta, Jaideep Srivastava, and Dongyeop Kang. Which modality should I use – text, motif, or image?: Understanding graphs with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 503–519, 2024. doi: 10.18653/v1/2024.findings-naacl.34. URL https://aclanthology.org/2024.findings-naacl.34/.

Andreea-Ioana Deac, Petar Veličković, Ognjen Milinkovic, Pierre-Luc Bacon, Jian Tang, and Mladen Nikolic. Neural algorithmic reasoners are implicit planners. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15529–15542, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/82e9e7a12665240d13d0b928be28f230-Abstract.html.

Yihe Deng, Chenchen Ye, Zijie Huang, Mingyu Derek Ma, Yiwen Kou, and Wei Wang. GraphVis: Boosting LLMs with visual knowledge graph integration. In *Advances in Neural Information Processing Systems*, volume 37, pp. 67511–67534, 2024. doi: 10.52202/079017-2155. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/7cb04f510593c9ba30da398f5e0a7e7b-Abstract-Conference.html.

Walter Didimo, Fabrizio Montecchiani, and Tommaso Piselli. Graph drawing for LLMs: An empirical evaluation. *arXiv preprint arXiv:2505.03678*, 2025. URL https://arxiv.org/abs/2505.03678.

Bowen Dong, Minheng Ni, Zitong Huang, Guanglei Yang, Wangmeng Zuo, and Lei Zhang. MIRAGE: Assessing hallucination in multimodal reasoning chains of MLLM. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b238324b309da12c7446d92c14db9f7e-Abstract-Conference.html.

Yuntao Du, Yiming Wang, Renshuo Yuan, Jincheng Yue, Yijing Chen, Yue Fan, Bo Zhang, Qian Li, and Lizhen Cui. VKG-QA: Visual knowledge graph-based question answer for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41213–41223, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Du_VKG-QA_Visual_Knowledge_Graph-based_Question_Answer_for_Large_Multimodal_Models_CVPR_2026_paper.html.

John Ellson, Emden R. Gansner, Lefteris Koutsofios, Stephen C. North, and Gordon Woodhull. Graphviz—open source graph drawing tools. In *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pp. 483–484. Springer, 2002. doi: 10.1007/3-540-45848-4_57. URL https://doi.org/10.1007/3-540-45848-4_57.

Yixin Fan, Zhao He, Yuxin Hou, Changhua Zhou, Zihao Liu, Peng Wang, Chenglong Lu, Xu Zhang, and Wei Wang. DiGraphHal-Bench: Evaluating multimodal large language models on complex directed graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30885–30894, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Fan_DiGraphHal-Bench_Evaluating_Multimodal_Large_Language_Models_on_Complex_Directed_Graphs_CVPR_2026_paper.html.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IuXR1CCrSi.

Bangwei Guo, Xujiang Zhao, Yanchi Liu, Wei Cheng, Shengyu Chen, Dongyue Li, Masaharu Morimoto, Takayuki Kuroda, Dimitris Metaxas, and Haifeng Chen. TopoAgent: A structure-aware perception-to-reasoning framework for diagram-to-graph topology extraction with large vision-language models. *arXiv preprint arXiv:2608.28701*, 2026. URL https://arxiv.org/abs/2608.28701.

Shuo Han, Yukun Cao, Zezhong Ding, Zengyi Gao, S. Kevin Zhou, and Xike Xie. See or say graphs: Agent-driven scalable graph understanding with vision-language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 41565–41589, 2026. doi: 10.18653/v1/2026.findings-acl.2066. URL https://aclanthology.org/2026.findings-acl.2066/.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 132876–132907, 2024. doi: 10.52202/079017-4224. URL https://papers.nips.cc/paper_files/paper/2024/hash/efaf1c9726648c8ba363a5c927440529-Abstract-Conference.html.

Cheng-Yu Hsieh, Jieyu Zhang, Zixian Ma, Aniruddha Kembhavi, and Ranjay Krishna. SUGARCREPE: Fixing hackable benchmarks for vision-language compositionality. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/63461de0b4cb760fc498e85b18a7fe81-Paper-Datasets_and_Benchmarks.pdf.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. URL https://arxiv.org/abs/2307.13702.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 292–305, 2023. doi: 10.18653/v1/2023.emnlp-main.20. URL https://aclanthology.org/2023.emnlp-main.20/.

Yunxin Li, Baotian Hu, Haoyuan Shi, Wei Wang, Longyue Wang, and Min Zhang. VisionGraph: Leveraging large multimodal models for graph theory problems in visual context. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 27903–27919, 2024. URL https://proceedings.mlr.press/v235/li24ab.html.

Zhuowei Li, Haizhou Shi, Yunhe Gao, Di Liu, Zhenting Wang, Yuxiao Chen, Ting Liu, Long Zhao, Hao Wang, and Dimitris N. Metaxas. The hidden life of tokens: Reducing hallucination of large vision-language models via visual information steering. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35799–35819, 2025. URL https://proceedings.mlr.press/v267/li25ca.html.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Yuan-Fang Li, Chen Gong, and Shirui Pan. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41540–41565, 2025. URL https://proceedings.mlr.press/v267/luo25t.html.

Huitong Pan, Qi Zhang, Cornelia Caragea, Eduard Dragut, and Longin Jan Latecki. FlowLearn: Evaluating large vision-language models on flowchart understanding. In *Proceedings of the 27th European Conference on Artificial Intelligence (ECAI)*, 2024. URL https://arxiv.org/abs/2407.05183.

Soham Petkar, Hari Aakash K, Anirudh Vempati, Akshit Sinha, Ponnurangam Kumaraguru, and Chirag Agarwal. A graph talks, but who's listening? rethinking evaluations for graph-language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32441–32462, 2026. doi: 10.18653/v1/2026.findings-acl.1624. URL https://aclanthology.org/2026.findings-acl.1624/.

Clara Petrova, Zhuo Chen, and Marin Soljačić. TraversalBench: Challenging paths to follow for vision language models. *arXiv preprint arXiv:2604.10999*, 2026. doi: 10.48550/arXiv.2604.10999. URL https://arxiv.org/abs/2604.10999.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4035–4045, 2018. doi: 10.18653/v1/D18-1437. URL https://aclanthology.org/D18-1437/.

Candace Ross, Florian Bordes, Adina Williams, Polina Kirichenko, and Mark Ibrahim. What's in common? multimodal models hallucinate when reasoning across scenes. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3e5b0db387078ac4968fd536d3c3a019-Abstract-Datasets_and_Benchmarks_Track.html.

Yuanfu Sun, Yuanhang Ren, Kang Li, Chuanhao Ji, Jiaxi Li, Jiajin Liu, Ninghao Liu, and Qiaoyu Tan. GraphVerse: A comprehensive visual graph reasoning benchmark for multimodal large language models. *arXiv preprint arXiv:2608.06769*, 2026. doi: 10.48550/arXiv.2608.06769. URL https://arxiv.org/abs/2608.06769.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 74952–74965, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ed3fea9033a80fea1376299fa7863f4a-Paper-Conference.pdf.

Petar Veličković, Lars Buesing, Matthew C. Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. Pointer graph networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2232–2244, 2020a. URL https://proceedings.nips.cc/paper_files/paper/2020/hash/176bf6219855a6eb1f3a30903e34b6fb-Abstract.html.

Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=SkgKO0EtvS.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The CLRS algorithmic reasoning benchmark. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22084–22102, 2022. URL https://proceedings.mlr.press/v162/velickovic22a.html.

Rohan Wadhawan, Hritik Bansal, Kai-Wei Chang, and Nanyun Peng. ConTextual: Evaluating context-sensitive text-rich visual reasoning in large multimodal models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 49733–49787, 2024. URL https://proceedings.mlr.press/v235/wadhawan24a.html.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*, volume 36, pp. 30840–30861, 2023a. doi: 10.52202/075280-1345. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/622afc4edf2824a1b6aaf5afe153fa93-Abstract-Conference.html.

Rongzheng Wang, Zhe Wang, Ke Qin, Rongwei Wang, Muquan Li, Yizhuo Ma, Yihong Huang, Jielei Wang, and Shuang Liang. Visual graph reasoning via knowledge compilation. *arXiv preprint arXiv:2609.22327*, 2026. URL https://arxiv.org/abs/2609.22327.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Yanbin Wei. GITQA-Aug-Legacy. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/Yanbin99/GITQA-Aug-Legacy. Legacy predecessor of the reorganized GVLQA datasets.

Yanbin Wei, Shuai Fu, Weisen Jiang, Zejian Zhang, Zhixiong Zeng, Qi Wu, James T. Kwok, and Yu Zhang. GITA: Graph to visual and textual integration for vision-language graph reasoning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 44–72, 2024. doi: 10.52202/079017-0002. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/00295cede6e1600d344b5cd6d9fd4640-Abstract.html.

Yanbin Wei, Jiangyue Yan, Chun Kang, Yang Chen, Hua Liu, James Kwok, and Yu Zhang. DynamicGTR: Leveraging graph topology representation preferences to boost VLM capabilities on graph QAs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40822–40832, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wei_DynamicGTR_Leveraging_Graph_Topology_Representation_Preferences_to_Boost_VLM_Capabilities_CVPR_2026_paper.html.

Mingrui Wu, Jiayi Ji, Oucheng Huang, Jiale Li, Yuhang Wu, Xiaoshuai Sun, and Rongrong Ji. Evaluating and analyzing relationship hallucinations in large vision-language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53553–53570, 2024. URL https://proceedings.mlr.press/v235/wu24l.html.

Zhongxing Xu, Chengzhi Liu, Qingyue Wei, Juncheng Wu, James Y. Zou, Xin Eric Wang, Yuyin Zhou, and Sheng Liu. More thinking, less seeing? assessing amplified hallucination in multimodal reasoning models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/777db387a5ccb131ba8c7cd155166b85-Abstract-Conference.html.

Sibei Yang, Ge Zheng, Jiajin Tang, Jiaye Qian, Hanzhuo Huang, and Cheng Shi. Discovering compositional hallucinations in LVLMs. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/299a08ee712d4752c890938da99a77c6-Abstract-Conference.html.

Zhiqiang Zhong and Davide Mottin. Exploring graph structure comprehension ability of multimodal large language models: Case studies. In *The Third Learning on Graphs Conference*, 2024. doi: 10.48550/arXiv.2409.08864. URL https://arxiv.org/abs/2409.08864. Extended abstract.

Yingjie Zhu, Xuefeng Bai, Kehai Chen, Yang Xiang, Jun Yu, and Min Zhang. Benchmarking and improving large vision-language models for fundamental visual graph understanding and reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 30678–30701, 2025. doi: 10.18653/v1/2025.acl-long.1482. URL https://aclanthology.org/2025.acl-long.1482/.

Appendix Contents

Part I — Foundations

A Metric and Notation Glossary **16**
A.1 Core objects . . . 16
A.2 Decision metrics . . . 16
A.3 Extraction metrics . . . 17
A.4 Faithfulness . . . 17
A.5 Independent visual-test metrics . . . 17
A.6 Clean-input comparison metrics . . . 17
A.7 Selection and geometry . . . 18
A.8 Complete glossary table . . . 18

B Models, Endpoints, Decoding Configuration, and Statistics **20**
B.1 Model identifiers and providers . . . 20
B.2 Structured output and decoding . . . 20
B.3 Snapshot bookkeeping . . . 20
B.4 Statistical testing . . . 20

Part II — Materials and Construction

C Dataset and Rendering Construction **21**
C.1 Ring construction . . . 21
C.2 Geometric validation . . . 21
C.3 Graphs and layouts . . . 21

D Prompts and Exact Task Wording **23**

E Follow-up stimulus construction (text-list insertions and unmodified-image probes) **25**

Part III — Extended Analysis of Main Results

F Error-Shape Taxonomy **27**

G Geometric Locus of Loss and Hallucination **29**

H Cycle-Competition Measurement Details **31**

I Selection Fidelity and Decomposition of the Collapse **32**

J Ruling Out Chance, Scoring, and Generative Artifacts **35**

K Which Drawings Fail: The Difficulty Score **38**

L Full Model-by-Model Failure Profiles **39**

Part IV — Robustness checks, text-list insertion, and image probes

M Readability Probes: Degree and Path Length **40**

N Human Reference on the Ring Drawings **43**

O Path-Level Replication of the Cycle Findings **44**
O.1 Does the correct path length survive extraction? . . . 44
O.2 Named path, not just length . . . 45

P Text-Level Interventions on Frozen Lists **46**

Q Localizing the Deficit: Gold-List Substitution **48**

R Single-Edge Insertion on Typed Lists **49**

S Chord Insertion on Trees **51**

T Per-edge picture questions (unmodified image; no edge added or deleted) **52**

U Observational Crossing Bins **54**

V Full Cross-Condition Results **57**

Part V — Scope and Related Work

W Scope, Limitations, and What Would Change the Conclusions **58**

X Extended Related Work **58**
X.1 Visual graph reasoning with VLMs . . . 59
X.2 Layout, modality, and encoding . . . 59
X.3 Hallucination and faithfulness . . . 60
X.4 The unresolved gap . . . 61

Reading guide. Readability probes: Appendices M and N. Loss vs. competition: Appendices H–J. Text-list insertions: Appendices R–S. Unmodified-image probes: Appendix T. Which drawings fail: Appendices K and U. Definitions and statistics: Appendices A and B. Appendix W lists the scope limits in full.

PART I — FOUNDATIONS

A METRIC AND NOTATION GLOSSARY

This appendix collects the formal objects and equations used throughout the paper. Table 6 is the one-page reference.

A.1 CORE OBJECTS

We define the objects in the order they appear in the pipeline: the true graph, what the model outputs, and how that output can go wrong.

Gold graph. $G^\star = (V^\star, E^\star)$ is the *gold graph*: the ground-truth vertices and edges of the diagram shown to the model. A *gold cycle* is any cycle of $G^\star$.

What the model outputs. Shown a rendering of $G^\star$, the model produces two things:

- a *listed graph* $\hat{G} = (\hat{V}, \hat{E})$, built from the edges it extracts;
- when asked whether $G^\star$ contains a cycle, a *named cycle* $\hat{c} = (v_1, \ldots, v_k)$: the ordered sequence of nodes it gives as its cycle.

When a named cycle is correct. The named cycle is a gold cycle only if every consecutive edge, including the closing edge (v_k, v_1), belongs to $E^\star$. Otherwise it is not a gold cycle.

False edges and fake cycles. A *false edge* is a listed edge that is not in $E^\star$. The listed graph $\hat{G}$ is *polluted* if $\hat{E}$ contains at least one false edge, whether or not it also misses true edges. A *fake cycle* is any cycle of $\hat{G}$ that uses at least one false edge.

Evaluation only. Every comparison with $G^\star$ is made at evaluation time. The model never sees $G^\star$.

Why pollution matters. A polluted list creates two different risks for anything that uses $\hat{G}$ downstream:

- *The correct structure is gone.* Once false edges are set aside, missing edges can leave no gold cycle at all.
- *The correct structure is present but not chosen.* A gold cycle survives, but false edges also create fake cycles that compete with it for selection.

A model, or a symbolic component after it, can be fully consistent and faithful to its own extracted list and still fall into either failure. Two common checks cannot tell these cases apart. Binary accuracy does not separate "right for the right reason" from "right for no verifiable reason." List faithfulness alone does not separate "faithful to a correct list" from "faithful to a polluted one."

This is why conventional accuracy is insufficient for structural tasks. It is a single bit computed from a pipeline with at least four points of failure, and those points fail differently across models (Section 6).

A.2 DECISION METRICS

Let $\mathrm{Cyc}(G) = \mathbf{1}[G \text{ contains a cycle}]$. Independent-test and clean-input-comparison metrics used later (V_{deg}, V_{SP}, R_G, R_E) are defined in Table 6.

Yes/no accuracy V_{pol} is whether the yes/no call matches gold cyclicity. Named-cycle accuracy V_{strict} additionally requires that, when the model says yes, the named cycle $\hat{c} = (v_1, \ldots, v_k)$ is a cycle of the gold graph $G^\star$ with no repeated vertices:

$$V_{\mathrm{pol}} = \mathbf{1}\big[\hat{y} = \mathrm{Cyc}(G^\star)\big], \tag{2}$$

$$V_{\mathrm{strict}} = \mathbf{1}\big[\hat{y} = \mathrm{Cyc}(G^\star)\big] \wedge \mathbf{1}\big[\hat{y} = 0 \vee \mathrm{SimpleCycle}(\hat{c}; G^\star)\big]. \tag{3}$$

Here $\mathrm{SimpleCycle}(\hat{c}; G^\star)$ equals 1 if and only if $k \geq 3$, the vertices of $\hat{c}$ are distinct after stripping an optional closing repeat, and every consecutive pair (v_i, v_{i+1}) together with the closing pair (v_k, v_1)

is an edge in $E^\star$. "Simple" means no repeated vertices; the edge check is against gold $E^\star$, so a cycle that exists only in the model's listed graph (via a hallucinated chord) does not count. On trees, V_{strict}=1 if and only if $\hat{y}$=0.

A.3 Extraction metrics

These metrics score the edge list a model reads off the picture. Write $\hat{E}_X$ for the listed edges and $E^\star$ for the true edges.

Precision asks how much of the list is correct. It is the share of listed edges that are true edges: $\text{Prec} = |\hat{E}_X \cap E^\star|/|\hat{E}_X|$.

Recall asks how much of the graph was found. It is the share of true edges that appear in the list: $\text{Rec} = |\hat{E}_X \cap E^\star|/|E^\star|$.

Cycle-status agreement Sym_X asks a simpler question. Do the list and the true graph agree on whether any cycle exists? On its own this says little. A graph with n nodes and no cycle has at most $n-1$ edges. So any list with n or more edges must contain a cycle, right or wrong. Most Ring lists are that long: 100.0% of Claude's, 99.4% of GPT's, 81.6% of Qwen's, and 66.7% of Gemini's.

Gold-cycle retention asks the sharper question. We delete every false edge from the list and keep only its true edges. Does a gold cycle remain?

$$\text{Gnd}_X = \mathbf{1}\big[\hat{G}^\star_X \text{ contains a cycle}\big], \qquad \hat{G}^\star_X = (\hat{V},\, \hat{E}_X \cap E^\star). \tag{4}$$

Only true edges remain in $\hat{G}^\star_X$, so any cycle in it is a gold cycle. Gnd is defined only for graphs that have a cycle. The subscript X marks the separate EXTRACT call. Gnd_C is the same measure on the list from the one-call LIST+DECIDE condition. The false edges are deleted by us at scoring time; the model never sees this filtered list.

Gnd=1 means the correct cycle could still be recovered from what the model listed. It does not mean the list is clean. The full list may still hold false edges, and those can form fake cycles that compete with the gold one.

Trees have no cycle, and neither does any part of a tree. So for trees we use Sym_X instead.

This measure splits every wrong LIST+DECIDE answer into exactly one of two kinds. If no gold cycle remains (Gnd_C=0), the answer is a *loss*. If a gold cycle remains (Gnd_C=1), the answer is a *competition* error. One more property follows. If the named cycle is in the model's own list and is also a gold cycle, all its edges are true edges, so Gnd_C=1. We checked both properties on every output.

A.4 Faithfulness

Faithfulness F checks only internal consistency — whether the named cycle $\hat{c}$ is a simple cycle of the model's own listed edges $\hat{E}_{\text{list}}$, independent of whether those edges are gold. We strip an optional repeated start node before this check:

$$F = \mathbf{1}\big[\text{SimpleCycle}(\hat{c}; \hat{E}_{\text{list}})\big] \quad (\text{if } \hat{y}=1;\, F{=}1 \text{ if } \hat{y}=0). \tag{5}$$

F and V_{strict} are the pairing that exposes the competition branch: F=1 with V_{strict}=0 means the model was fully consistent with a list that was itself wrong.

A.5 Independent visual-test metrics

Degree comparison V_{deg} and shortest-path length V_{SP} (with extracted-graph counterparts Sym_{deg}, Sym_{SP}, and gold-filtered Gnd_{SP}) are scored on the same images as the cycle task but do not require a named cycle. Named-path $V_{\text{SP}}^{\text{named}}$ additionally requires a gold s–t walk of gold length. Operational definitions are in Table 6.

A.6 Clean-input comparison metrics

R_G is V_{strict} given the clean gold edge list as text with no image. R_E is whether the named cycle is a valid cycle of the model's own listed graph, independent of $G^\star$.

A.7 SELECTION AND GEOMETRY

On a retained list (Gnd_C=1), write $\hat{E}_{\mathrm{list}}$ for the listed edges in LIST+DECIDE. Let g_{gold} be the length of the shortest surviving gold cycle in $\hat{E}_{\mathrm{list}} \cap E^\star$ and g_{fake} the length of the shortest fake cycle in $\hat{E}_{\mathrm{list}}$. The list falls in cell **A** (gold shorter: $g_{\mathrm{fake}} > g_{\mathrm{gold}}$), **T** (tie), or **B** (fake shorter: $g_{\mathrm{fake}} < g_{\mathrm{gold}}$); s_{loss} is V_{strict} on loss lists. Selection fidelity s_A, s_T, s_B is V_{strict} within each cell; an ideal shortest-cycle selector has $(s_A, s_T, s_B) = (1, \frac{1}{2}, 0)$. Then

$$P(V_{\mathrm{strict}}{=}1) = P(\mathrm{Gnd}{=}1)\big[P(A)\, s_A + P(T)\, s_T + P(B)\, s_B\big] + P(\mathrm{Gnd}{=}0)\, s_{\mathrm{loss}}. \tag{6}$$

For Ring geometry, a *crossing* is a pair of gold edges with four distinct endpoints whose straight segments strictly intersect. K is the minimum, over all gold cycles of length equal to the girth, of the summed crossings on that cycle's edges. S_{adj} counts non-edges (u, v) whose endpoints are consecutive on the ring and whose gold distance satisfies $d(u, v) + 1 \leq$ girth. We call such a pair a *neighbour shortcut*. Figure 3 illustrates the definition. The difficulty score D_{fin} is Equation 1: visible crossings, nodes, and edges, with no gold cycle required.

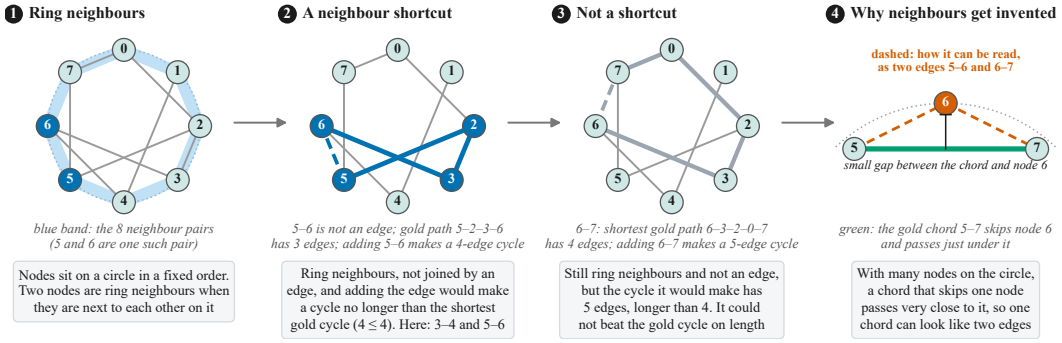


Figure 3: Ring neighbours and neighbour shortcuts, on the toy graph used throughout the appendices. (1) On a Ring drawing the nodes sit on a circle in a fixed order. Two nodes are *ring neighbours* when they are next to each other on that circle (blue band). (2) A *neighbour shortcut* is a pair of ring neighbours that is not an edge of the true graph, and whose edge, if added, would create a cycle no longer than the shortest gold cycle. Here 5–6 qualifies: the gold path 5–2–3–6 has 3 edges, so adding 5–6 makes a 4-edge cycle, and the shortest gold cycle also has 4 edges. S_{adj} counts such pairs; this toy graph has two (3–4 and 5–6). (3) The pair 6–7 is also a non-edge between ring neighbours, but adding it makes a 5-edge cycle, longer than the gold one, so it is not counted. (4) One reason a model may list neighbour edges that are not there (schematic, not to scale): a gold chord that skips one node passes close to it, and can look like two neighbour edges through that node. With many nodes on the circle, this gap can be very small (Appendix G).

A.8 COMPLETE GLOSSARY TABLE

Table 6 provides formal definitions for all evaluation metrics.

Table 6: Glossary of graph terms, metrics, and experimental indicators.

Category	Metric	Operational Definition
Core Terms	Gold graph $G^\star$	Benchmark ground truth: the vertices and edges actually present in the source graph.
	Listed graph $\hat{G}$	Graph formed by the edges extracted and reported by the model.
	Named cycle $\hat{c}$ / gold cycle	The node sequence the model gives as its cycle / a cycle of $G^\star$. The named cycle is correct when it is a gold cycle.
	False edge / fake cycle	A false edge is listed but absent from $G^\star$; a fake cycle uses at least one false edge.
	Polluted extraction	A listed edge set containing at least one false edge; it may also omit gold edges.
Conditions	ASK / EXTRACT / LIST+DECIDE	Picture $\rightarrow$ yes/no + named cycle / picture $\rightarrow$ edge list / picture $\rightarrow$ edge list, then yes/no + named cycle, in one call. Subscripts X and C (Gnd_X, Gnd_C) mark the EXTRACT and LIST+DECIDE calls.
Cycle Decision	V_{pol} (Yes/no accuracy)	Binary classification correctness: $\mathbf{1}[\hat{y} = \mathrm{Cyc}(G^\star)]$.
	V_{strict} (Named-cycle accuracy)	Correct yes/no answer and, if cyclic, $\hat{c}$ is a true gold cycle of $G^\star$ with no repeated vertices. On trees: strictly no cycle.
Extraction Quality	Prec (Edge Precision)	Fraction of listed edges belonging to $G^\star$: $\lvert\hat{E} \cap E^\star\rvert / \lvert\hat{E}\rvert$.
	Rec (Edge Recall)	Fraction of gold edges recovered in extraction: $\lvert\hat{E} \cap E^\star\rvert / \lvert E^\star\rvert$.
Gold-Cycle Retention	Sym_X (Cycle-status agreement)	Extracted graph cyclicity matches gold: $\mathbf{1}[\mathrm{Cyc}(\hat{G}_X) = \mathrm{Cyc}(G^\star)]$.
	Gnd_X / Gnd_C (Gold-cycle retention)	After false listed edges are removed at evaluation time, the remaining true positives retain a gold cycle (cyclic graphs only). This does not imply a clean list.
	Loss / Competition	Two kinds of wrong named cycle: Gnd=0 (no gold cycle among the listed gold edges) / Gnd=1 (a gold cycle is listed, but the named cycle is not gold, usually a fake cycle of the list).
Output Faithfulness	F (Faithfulness)	The named cycle $\hat{c}$ is a simple cycle of the model's own listed edges, after stripping an optional repeated start node.
Independent Visual Tests	V_{deg} (Visual Degree)	Correctly identifying higher-degree vertex (n=140 items, 2 pairs/graph).
	$\mathrm{Sym}_{\mathrm{deg}}$ (Extracted Degree)	Extracted graph yields the correct higher-degree vertex (strictly higher).
	V_{SP} (Visual Shortest Path)	Correctly reporting gold shortest-path integer length (n=195 items, 3 queries/graph).
	$V_{\mathrm{SP}}^{\mathrm{named}}$ (Named path)	Named s–t walk is a simple path in $G^\star$ of gold shortest-path length.
	$\mathrm{Sym}_{\mathrm{SP}}$ (Extracted Shortest Path)	Extracted graph yields the correct shortest-path length.
	$\mathrm{Gnd}_{\mathrm{SP}}$ (Gold-path retention)	After extracted edges not in $E^\star$ are removed at evaluation time, the remaining true-positive edges still yield the correct shortest-path length. Recoverability only unless a named walk is also scored.
Clean-Input Comparisons	R_G (Gold-List Reasoning)	V_{strict} on gold edge list $E^\star$ as plain text with no image.
	R_E (Own-List Reasoning)	Named cycle $\hat{c}$ forms a valid cycle in the model's own extracted graph.
	Finder V_{strict}	V_{strict} of a gold-blind depth-first cycle finder run on a stored list.
Selection	Cells A / T / B	Gold shorter / tie / fake shorter. On a retained list: the shortest fake cycle is longer than / equal to / shorter than the shortest surviving gold cycle.
	s_A, s_T, s_B (Selection fidelity)	V_{strict} within cell A / T / B; ideal shortest-cycle selector $(1, \frac{1}{2}, 0)$.
	$I(V_{\mathrm{pol}}; V_{\mathrm{strict}})$	Mutual information $I(V_{\mathrm{pol}}; V_{\mathrm{strict}})/H(V_{\mathrm{strict}})$ over the 500 cyclic graphs only. On an all-cyclic set this is near zero whenever the model almost always says yes; GPT's 0 is exactly that.
Ring Geometry	K (Gold-cycle crossings)	Crossings on the least-crossed shortest gold cycle: the minimum over shortest gold cycles of the summed Ring crossings on the cycle's edges.
	S_{adj} (Neighbour shortcuts)	Ring-consecutive non-edges (u, v) with $d(u, v) + 1 \leq$ girth.
	$N_\times$ (Visible crossings)	Total number of edge crossings drawn on the picture (Base: visible strokes; Ring: straight-chord intersections). Unlike K, it needs no gold cycle.
	D_{fin} (Difficulty score)	$\ln(1 + N_\times) + 0.35\,n - 0.25\,m$ (Equation 1).

B MODELS, ENDPOINTS, DECODING CONFIGURATION, AND STATISTICS

B.1 MODEL IDENTIFIERS AND PROVIDERS

Evaluations used `claude-sonnet-5` (Anthropic Vertex), `gpt-5.6-luna` (OpenAI API), `gemini-3.8-flash` (Google Vertex), and `qwen/qwen3.8-flash` (OpenRouter). Throughout the paper we refer to them as Claude, GPT, Gemini, and Qwen.

B.2 STRUCTURED OUTPUT AND DECODING

All calls enforce structured JSON output schemas with a maximum output budget of $100{,}000$ tokens. Claude, Gemini, GPT, and Qwen used default thinking provided by the provider. A response is parsed only if it is a JSON object with the required fields; any other response is a parse failure and is re-queried.

B.3 SNAPSHOT BOOKKEEPING

One snapshot per model is used for every condition, including the text-list insertions and the unmodified-image probes (Appendices R–T). No condition mixes snapshots.

B.4 STATISTICAL TESTING

This section describes the statistical procedure used throughout the paper, from single comparisons to the full family of tests.

Paired comparisons. Most comparisons run two conditions on the same graphs, such as Base vs. Ring V_{strict} or the frozen list vs. the shuffled list. Each graph then yields a pair of binary outcomes (Y_i, Z_i). We compare these pairs with an exact two-sided McNemar test. The test depends only on the discordant pairs, $b = \#\{i : Y_i{=}1, Z_i{=}0\}$ and $c = \#\{i : Y_i{=}0, Z_i{=}1\}$.

Intervals and rounding. Proportions are reported with Wilson 95% intervals. Paired differences are reported with Newcombe intervals and a graph-level bootstrap (2,000 resamples). Gaps and paired differences in the text are computed from exact per-item counts before rounding. They may therefore differ by up to ± 0.1 points from a difference of the one-decimal percentages shown in the tables.

Multiple testing. The paper has 36 primary tests, corrected together with Holm–Bonferroni. For each of the four models, these are:

- eight paired McNemar tests: Base vs. Ring V_{strict}; Base vs. Ring V_{pol}; shuffled vs. frozen list; frozen list vs. ASK; R_G vs. LIST+DECIDE; T1 shorter vs. longer; T2 short vs. long; and T3 acceptance at low vs. high crossings;
- one unpaired Fisher test: Base V_{strict} on GITQA vs. generated graphs.

This gives 32 McNemar tests and 4 Fisher tests in total.

Claims of no difference. A non-significant result alone does not establish that two conditions are equivalent. Where we claim no difference, we use two one-sided tests with a ± 10-point equivalence margin, paired by graph.

Comparing models. A Cochran's Q test compares the four models on the same graphs. A generalized estimating equation, with graphs as clusters, models V_{strict} as a function of layout $\times$ model.

Retention vs. selection. One pairing recurs across Sections 5–6. On the same one-call response, it compares $Y = \text{Gnd}_C$ (a gold cycle is retained in the listed edges) with $Z = V_{\text{strict}}$ (the named cycle is gold). Gnd_C relies on filtering applied at evaluation time, which the model cannot perform. The gap between the two therefore counts answers in which a gold cycle was *retained but not selected*.

Results after correction. Of the 36 primary tests, 20 remain significant after Holm correction:

- Base vs. Ring V_{strict}, for all four models (Table 7);
- R_G vs. LIST+DECIDE, for all four models;

Table 7: Headline V_{strict} with Wilson 95% intervals, Base−Ring difference with Newcombe 95% interval, exact McNemar test, and discordant-pair odds ratio b/c with exact 95% interval (500 cyclic graphs).

Model	Base	Ring	Difference (pp)	McNemar b, c; p; b/c
Claude	39.0 [34.8, 43.3]	4.8 [3.2, 7.0]	34.2 [29.6, 38.7]	181, 10; 10^{-41}; 18.1 [9.6, 38.4]
GPT	89.2 [86.2, 91.6]	42.4 [38.1, 46.8]	46.8 [41.7, 51.5]	249, 15; 10^{-55}; 16.6 [9.9, 30.1]
Gemini	99.0 [97.7, 99.6]	44.0 [39.7, 48.4]	55.0 [50.5, 59.3]	276, 1; 10^{-80}; 276 [49, 10940]
Qwen	41.8 [37.6, 46.2]	8.8 [6.6, 11.6]	33.0 [27.9, 37.9]	191, 26; 10^{-31}; 7.3 [4.9, 11.5]

- T1 shorter vs. longer, for all four models;
- T2 short vs. long, for Claude and Qwen (GPT and Gemini are at ceiling);
- the T3 per-edge crossing probe, for Claude and Qwen;
- Base vs. Ring V_{pol}, for Claude ($100.0\% \rightarrow 97.4\%$), Gemini ($99.8\% \rightarrow 84.0\%$), and Qwen ($96.4\% \rightarrow 99.8\%$);
- GITQA vs. generated Base V_{strict}.

Model comparisons. Cochran's Q rejects equal accuracy across the four models on both Ring (Q=411) and Base (Q=635). In pairwise comparisons, GPT and Gemini do not differ on Ring (Holm p=0.55), and Claude and Qwen do not differ on Base (p=0.35). In the graph-clustered model, Ring multiplies Claude's odds of naming a gold cycle by 0.079 [0.051, 0.122], with a within-graph working correlation of 0.09.

PART II — MATERIALS AND CONSTRUCTION

C DATASET AND RENDERING CONSTRUCTION

C.1 RING CONSTRUCTION

Ring drawings position all n vertices at equal angular increments $\theta_k = 2\pi k/n$. Vertex ordering follows either node-id order (82.0%) or cycle order (18.0%). Automated checks verify minimum vertex clearance and edge–node distance. Furthermore, every graph was checked by 5 independent reviewers to ensure the graphs are readable and correct.

C.2 GEOMETRIC VALIDATION

Table 8 reports crossing, chord, spacing, and clearance statistics. Base crossings are visible strokes on the stored drawing: exact Bézier intersections for the 250 generated graphs, and a count read from the PNG for the GITQA graphs and the trees that were in the earlier audited subset. Ring crossings are exact intersections of the stored straight chords.

C.3 GRAPHS AND LAYOUTS

The test set has two parts. The first part is 500 graphs that each contain a cycle. Half of them (250) come from the GITQA-Aug-Legacy cycle/hard split, from `graph1070` onward. These have 26 to 29 nodes. The other 250 are new graphs that we made for this study. They are connected, have 11 to 22 nodes, and their shortest cycle has 3 to 6 edges. They have not been released before. The second part is 50 trees: connected graphs with no cycle, each with 26 nodes and 25 edges. All drawings are labelled and undirected.

The trees check the yes/no answer. A model that always says "cycle" would score perfectly on the 500 cyclic graphs and fail every tree. So we report cyclic and tree results separately, and combine them only in balanced accuracy.

The new graphs come from our own generator. It builds one short cycle and attaches trees to it. The code is in the supplementary material.

Table 8: Geometric properties (means) across the 500 cyclic gold graphs, with the GITQA / generated split and the two Ring vertex orders. Cross.: visible-stroke edge crossings for Base and exact straight-chord crossings for Ring. Chord: mean edge length. Min sp.: minimum vertex spacing. Clear.: minimum edge–node clearance. Lengths in % of canvas diagonal.

Set	**Layout**	n	m	Cross.	Cross./edge	Chord	Min sp.	Clear.
All cyclic (500)	Base (`dot`)	21.9	23.3	1.9	0.09	19.3	7.73	2.19
	Ring	21.9	23.3	53.4	2.22	33.2	8.88	1.63
GITQA (250)	Ring	27.4	28.8	71.0	2.46	28.6	6.47	0.77
Generated (250)	Ring	16.4	17.8	35.9	1.98	37.8	11.29	2.50
Node-id rings (410)	Ring	20.7	22.1	64.8	2.69	37.2	9.41	1.83
Cycle-order rings (90)	Ring	27.5	29.0	1.9	0.06	14.9	6.46	0.74
Trees (50)	Base / Ring	26.0	25.0	0.5 / 86.4	0.02 / 3.45	14.9 / 37.4	6.12 / 6.82	3.57 / 0.95

Each graph is drawn twice. The **Base** drawing is the Graphviz `dot` layout of that graph (Ellson et al., 2002). The **Ring** drawing is our own: it places all nodes at equal angles around a circle. The nodes go around the circle in one of two orders. In *node-id order*, they follow their numbers (410 graphs: 160 from GITQA and all 250 new graphs). In *cycle order*, the order follows a walk around one gold cycle (90 GITQA graphs). Trees keep the order stored in their layout files. Ring images are 12×12 inches at 150 dpi (1800×1800 pixels).

Base and Ring use the same edge list. The graph does not change; only the drawing does. Automated checks found no missing edges, no overlapping nodes, and no rendering errors in either drawing. Every edge is drawn and every node label is present, so a drop on Ring cannot come from a missing edge or label. Because the task is identical, Section 5 can attribute the gap between yes/no accuracy and named-cycle accuracy to the drawing.

The drawings differ. Mean edge crossings rise from 1.9 on Base to 53.4 on Ring, about 28 times as many (Table 8). Figure 4 shows the drawings side by side on `gitqa_229`.

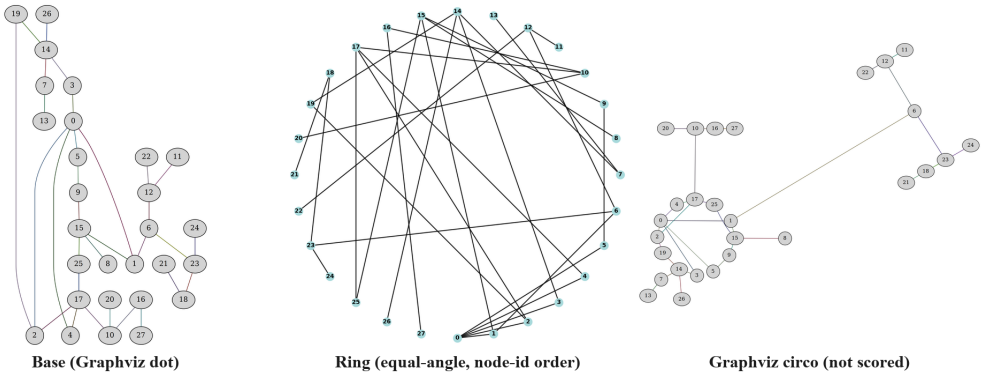
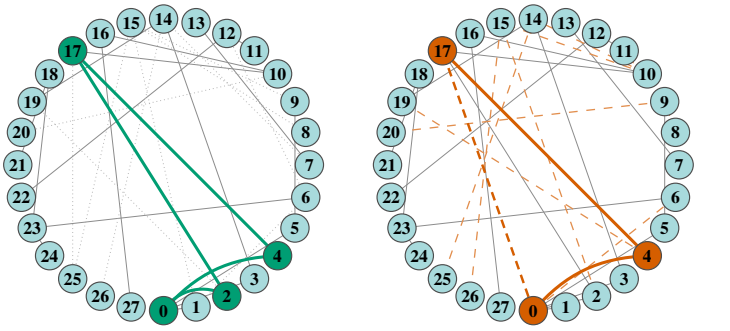


Figure 4: Drawing comparison on GITQA `graph1299` (`gitqa_229`). Left: official Base (`dot`). Centre: our regular equal-angle ring (node-id order). Right: official `circo`—a radial Graphviz render, not a ring. Cycle detection is evaluated on Base and the ring only. The gold 4-cycle is $(0, 4, 17, 2)$.



Qwen's listed gold edges
(20 of 31; missed gold edges dotted)
gold cycle $(0, 4, 17, 2)$ survives

Qwen's full list
(11 false edges dashed)
names $(0, 4, 17)$, closed by false edge $(0, 17)$

Figure 5: Polluted-list failure on `gitqa_229`, drawn from Qwen's stored LIST+DECIDE output. Left: Qwen's listed gold edges (20 of 31; missed gold edges dotted); the gold cycle $(0, 4, 17, 2)$ survives (Gnd_C=1). Right: Qwen's full list, with its 11 false edges dashed; it names $(0, 4, 17)$, closed by the false edge $(0, 17)$. The named cycle is list-faithful but not a gold cycle. Highlighted edges are drawn curved where a straight chord would pass under a node; the model saw straight chords, as in the real Ring drawing of this graph (Figure 4, centre).

D PROMPTS AND EXACT TASK WORDING

ASK: visual cycle decision

```
Undirected graph. Does it contain a cycle?
Return ONLY the JSON object fields: has_cycle (bool) and cycle_nodes
(ordered simple cycle without repeating the start node; empty if no cycle).
```

EXTRACT: visual extraction

```
Undirected graph with nodes 0..n-1.
Return ONLY a compact JSON object with:
  "num_nodes": n,
  "edges": [[u,v], ...]  // each edge once, with u < v
Do not pretty-print; keep JSON minified. No adjacency list.
```

LIST+DECIDE: list, then decide

```
Undirected graph with nodes 0..n-1.
First list every edge you observe in the image as pairs (node, node).
Then, using only that edge list, determine whether the graph contains a cycle.
If yes, also list the cycle nodes.
Return ONLY JSON fields: num_nodes, edges [[u,v] with u<v],
has_cycle (bool), cycle_nodes (ordered simple cycle without repeating start;
empty if no cycle).
```

Frozen List and Shuffled List

```
Here is the edge list extracted from the image: <JSON edges>.
Using ONLY this list, answer: does the graph contain a cycle?
If yes, list the cycle nodes. Do not add or remove edges.
Return ONLY JSON fields: has_cycle (bool), cycle_nodes
(ordered simple cycle without repeating start; empty if no cycle).
```

Algorithmic DFS Prompt

```
Here is an undirected edge list: <JSON edges>.
Using ONLY this list, decide whether the graph contains a cycle.
Do not add or remove edges.
Follow this procedure internally (do not print the trace):
1. Build an adjacency list from the given edges (treat each edge as undirected).
2. Run DFS from every unvisited node. Keep parent pointers.
3. If a back edge to a visited non-parent is found, reconstruct that cycle.
4. Before claiming a cycle, verify that every consecutive pair in cycle_nodes
(including last to first) appears in the given edge list.
If any pair is missing, do not claim a cycle.
Return ONLY JSON fields: has_cycle (bool), cycle_nodes
(ordered simple cycle without repeating start; empty if no cycle).
```

Gold-List Reasoning (R_G)

```
You are given an undirected graph as an edge list. Does it contain a cycle?
Graph edges:
<JSON edges>
Return ONLY the JSON object fields: has_cycle (bool) and cycle_nodes
(ordered simple cycle without repeating the start node; empty if no cycle).
```

Degree Comparison (V_{deg})

```
The image is an undirected graph. Node labels are integers.
Degree means the number of neighbors (incident edges).
Which of the two nodes has the higher degree: a or b?
Ties are not possible in this item.
Return ONLY a JSON object: {"higher_degree_node": a}
or {"higher_degree_node": b}.
```

Shortest-Path Length (V_{SP})

```
The image is an undirected graph. Node labels are integers.
What is the length of the shortest path between node s and node t?
Length means the number of edges in a shortest path.
Return ONLY a JSON object: {"shortest_path_length": <non-negative integer>}.
```

Named Shortest Path (C_{SP}, Section O.2)

```
Undirected graph with nodes 0..n-1. Node labels are integers.
First list every edge you observe as pairs (node, node).
Then, using ONLY that edge list, give the length of the shortest path
between node s and node t, and list the nodes of one such path.
Return ONLY JSON fields: num_nodes, edges [[u,v] with u<v],
shortest_path_length (non-negative integer, number of edges),
path_nodes (ordered from s to t, including both endpoints).
```

Per-edge image probe ($T3$, Appendix E)

```
The image is an undirected node-link drawing. Node labels are integers 0, 1, 2, ...
For each numbered pair, answer whether that exact undirected edge is drawn.
Answer true only if you see a black line connecting those two labelled nodes.
Answer false if the two nodes are nearby but no edge is drawn between them.

1. nodes u1 and v1
...
8. nodes u8 and v8

Return ONLY JSON with eight booleans pair_1 ... pair_8, in this same order.
```

Edge Insertion ($T1$, $T2$, Appendix E). T1 and T2 reuse the Gold-List Reasoning (R_G) prompt above verbatim, substituting the modified edge list (gold plus one inserted edge) for `<JSON edges>`.

E FOLLOW-UP STIMULUS CONSTRUCTION (TEXT-LIST INSERTIONS AND UNMODIFIED-IMAGE PROBES)

This appendix details the three follow-ups of Appendices R–T, labelled T1 (edge insertion), T2 (tree chord), and T3 (edge-presence probe). The stored-output analyses of Appendix G need no construction beyond the stored lists and Ring positions used in Sections 5–6. T1 and T2 insert one edge into a text list and render no images. T3 is a present/absent probe on the existing Ring PNGs: it does not insert edges and does not render new drawings. All three reuse the cyclic and tree gold graphs and the equal-angle Ring positions of Appendix C. This appendix describes only how the stimuli are built. The results are in Appendix R for T1 (Figure 19), Appendix S for T2 (Figure 20), and Appendix T for T3.

T1: single-edge insertion (arm selection). For each cyclic gold graph $G^\star$ with girth g, we enumerate every non-gold pair (u, v), compute its shortest-path distance d in $G^\star$, and group it by the length $d{+}1$ of the cycle it would create: *shorter* if $d{+}1{<}g$, *equal* if $d{+}1{=}g$, *longer* otherwise. Each non-empty group is one *arm*. Within an arm we take the pair with the most extreme length (the shortest for the shorter arm and the longest for the longer arm; every pair in the equal arm has length g; ties are broken by the smallest vertex pair (u, v), or the largest for the longer arm), so each arm tests the strongest available manipulation rather than an arbitrary pair. A graph contributes to the shorter arm only if a strictly shorter alternative exists. Cross-model comparisons use the 60 graphs evaluated for all four models.

T2: tree chord selection. For each gold tree, we compute all-pairs shortest-path distances and select a *short* chord at the smallest available distance $d \in \{2, 3\}$ (creating a 3- or 4-edge cycle) and a *long* chord at the smallest available distance $d \geq 6$ (creating a 7-edge cycle), each with ties broken by smallest (u, v). The baseline arm re-runs R_G on the unmodified tree edge list; the short/long arms add exactly one chord.

T3: image probe-pair selection. For each cyclic gold graph, we partition non-gold pairs into *ring-adjacent* (consecutive on the equal-angle Ring) and *far* (ring hop distance ≥ 8), and partition gold edges into the top and bottom quartile by Ring crossing count (ties broken by edge identity), taking at least 2 edges per quartile. We sample 2 pairs of each of the four kinds per graph (seeded by graph id for reproducibility), filling with other pairs on graphs where a category has fewer than 2 candidates, for exactly 8 probes per graph. All 8 probes for a graph are asked in one structured call against that graph's unmodified Ring image (Appendix D). Cross-model comparisons use the 75 graphs evaluated for all four models (150 probes of each kind).

Execution. T2 uses the 50 trees. Shared insertion and per-edge probe sets are the 60 and 75 graphs evaluated for all four models. All use the snapshots and decoding settings of Appendix B.

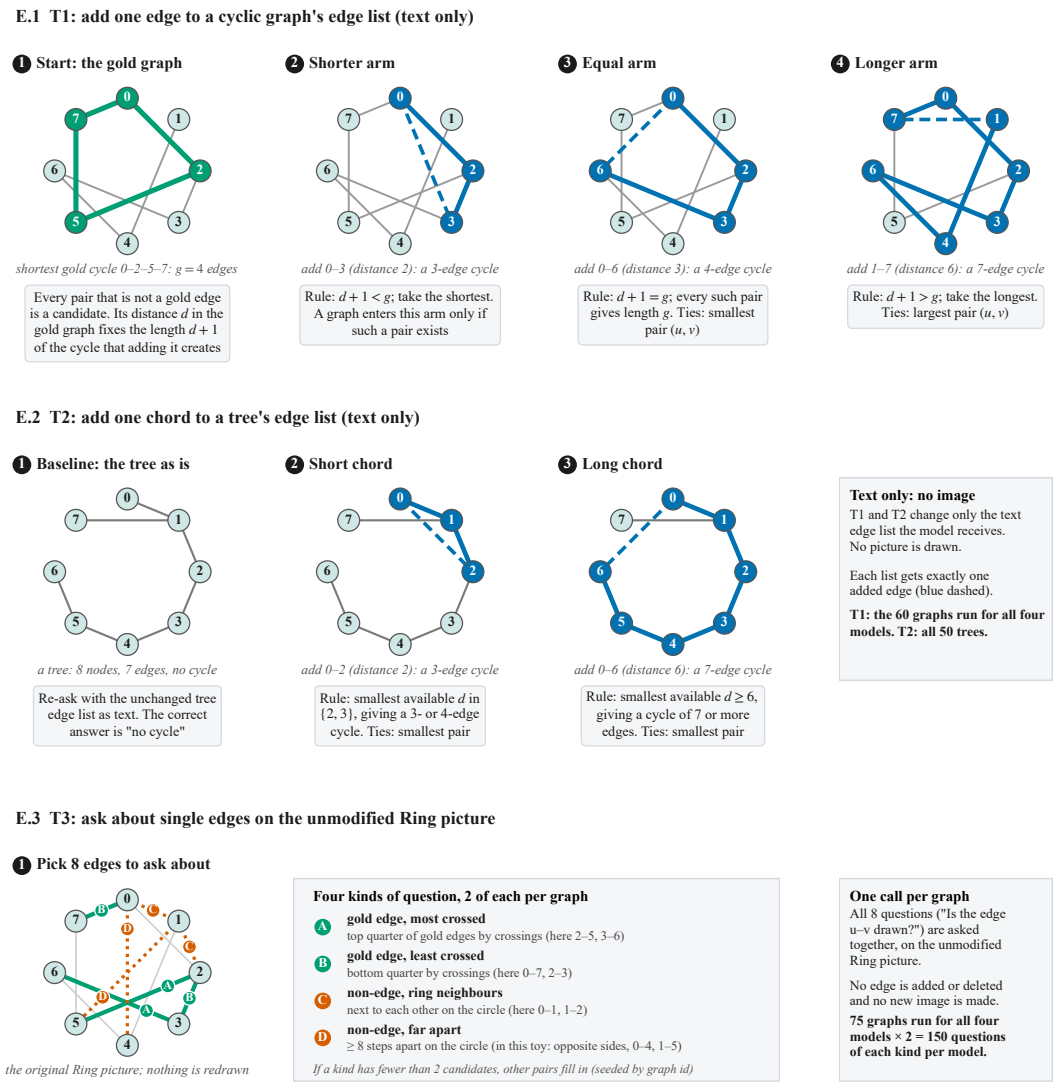


Figure 6: Walkthrough of this appendix on toy graphs. It shows how each stimulus is built, not results; the results are in Appendices R (T1), S (T2), and T (T3). **E.1 (T1).** (1) The gold graph's shortest cycle has g=4 edges. Adding a non-gold pair at distance d creates a cycle of d+1 edges. (2)–(4) The three arms: a pair that creates a cycle shorter than g (the shortest available), equal to g, or longer than g (the longest available). Blue dashed: the one added edge; blue solid: the gold path that closes the new cycle. **E.2 (T2).** (1) A tree has no cycle; the baseline re-asks its unchanged edge list. (2) A short chord at distance 2 or 3 creates a 3- or 4-edge cycle. (3) A long chord at distance 6 or more creates a cycle of at least 7 edges. T1 and T2 change only the text edge list; no image is drawn. **E.3 (T3).** On the unmodified Ring picture we ask about 8 single edges per graph, 2 of each kind: gold edges with the most (A) and fewest (B) crossings, and non-edges between ring neighbours (C) or far apart on the circle (D). All 8 are asked in one call.

PART III — EXTENDED ANALYSIS OF MAIN RESULTS

This part gives the full evidence behind the main results. Table 9 lists the question each appendix answers.

Terms. The *named cycle* is the cycle a model names. A *gold* edge or cycle is one that exists in the true graph. A *fake* or *invented* edge is one the model lists that is not in the true graph. *Loss* means

a wrong answer where no gold cycle survives among the model's listed gold edges. *Competition* means a wrong answer where a gold cycle is still in the list, but the named cycle is not gold (usually a fake cycle of the list). Appendix A gives the full definitions.

Table 9: Roadmap of Part III: the question each appendix asks and its short answer.

Appendix	Question	Short answer
F: Error-shape taxonomy	What kinds of wrong answers does each model give?	Claude and Qwen name cycles built on invented edges, GPT is usually one edge from a gold cycle, Gemini splits between one-edge errors and "no cycle".
G: Geometric locus	Which edges get dropped or invented?	Heavily crossed edges are dropped; invented edges tend to join ring neighbours.
H: Cycle competition	Are fake cycles no longer than the gold one available in the list?	Yes, in most lists of Claude, Qwen, and GPT; a simple finder fails for the same reason.
I: Selection fidelity	How much of the drop is losing the cycle, and how much is naming the wrong one?	It depends on the model; invented edges are the largest single lever.
J: Chance, scoring, and simulation	Could chance, the scorer, or a simple generative story explain the results?	Loss is mostly how many gold edges remain; scoring is not the cause; simulated lists match the observed ones.
K: Difficulty score	Which drawings fail?	D_{fin}, from visible crossings, nodes, and edges, separates Base from Ring and keeps the two model pairs apart. It is weak inside one layout.
L: Failure profiles	Do the four models fail at the same stage?	No; a single accuracy number hides the differences.

F Error-Shape Taxonomy

Takeaway. The four models make different kinds of wrong answers. Claude and Qwen mostly name cycles built on several invented edges, GPT is usually one closing edge away from a gold cycle, and Gemini splits between one-edge errors and "no cycle" answers.

Hypothesis. If wrong named cycles on Ring come from different stages for different models, their shapes should differ. A model that is close to a correct named cycle but picks the wrong closing edge should make mostly premature closes. A model working from a heavily polluted list should make mostly edge hallucinations.

Experiment. When a model answers a cyclic Ring drawing wrongly on the visual cycle task (ASK), we compare its reported named cycle with the gold graph $G^\star$, up to cyclic rotation and reversal, and place it in one of four types:

- **False No:** the model says there is no cycle.
- **Repeated vertex:** after the usual closing copy of the start node is dropped, some node still appears twice. The answer walks back through a node instead of naming a cycle.
- **Premature Close:** exactly one named cycle edge is missing from $G^\star$, wherever it sits on the cycle.
- **Edge Hallucination:** two or more named cycle edges are missing from $G^\star$.

Table 10 reports the breakdown on all 500 graphs, and Figure 7 shows one example of each type.

Observation. The dominant error shape differs by model. Claude's wrong answers (476 in total) are mostly edge hallucinations, and Qwen's (456) lean the same way. GPT's (288) are mostly premature closes. Gemini's (280) are split between premature closes and "no cycle" answers. The repeated-vertex counts sit inside those wrong-answer totals.

Interpretation. The shapes follow the per-model profiles of Section 6. Claude and Qwen name cycles built on several invented edges, which is consistent with polluted, loss-dominated lists. GPT

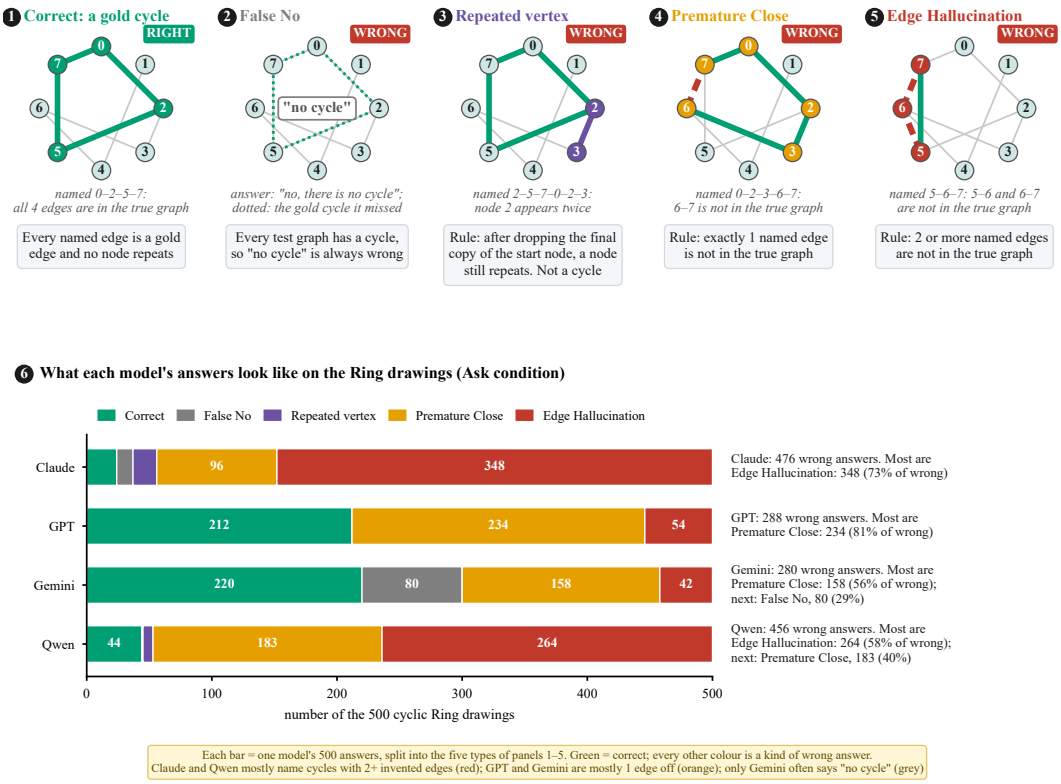


Figure 7: Walkthrough of this appendix on the toy graph. (1)–(5) The five answer types, one example each. Green solid: a named edge that is in the true graph; red dashed: a named edge that is not. (1) Correct: the named cycle 0–2–5–7 uses only gold edges. (2) False No: the model says there is no cycle, although every test graph has one. (3) Repeated vertex: the named sequence visits node 2 twice, so it is not a cycle. (4) Premature Close: exactly one named edge (6–7) is not in the true graph. (5) Edge Hallucination: two or more named edges (5–6 and 6–7) are not in the true graph. (6) How the 500 cyclic Ring answers of each model split over the five types (Table 10); the text beside each bar names the most common wrong type.

Table 10: Error shapes on the equal-angle visual cycle task (N=500), classified up to cyclic rotation. False No: says no cycle. Repeated: after dropping a repeated start node, some node still appears twice. Premature Close: exactly one named cycle edge missing from gold. Edge Hallucination: two or more missing edges. Rows sum to 500.

Model	Named-cycle correct	False No	Repeated	Premature Close	Edge Hall.
Claude	24	13	19	96	348
GPT	212	0	0	234	54
Gemini	220	80	0	158	42
Qwen	44	1	8	183	264

is usually one closing edge away from a gold cycle, which is consistent with a model that reads most of the drawing while invented edges close a fake cycle that it then names. Gemini's "no cycle" answers are a failure mode that yes/no scoring would register directly, unlike the other three.

Limitation. The taxonomy classifies the named cycle only. It does not say which listed edges produced it; the LIST+DECIDE-list analyses of Appendices G–I address that.

G GEOMETRIC LOCUS OF LOSS AND HALLUCINATION

Takeaway. Edges dropped during extraction are the heavily crossed ones. Invented edges tend to join ring neighbours (nodes next to each other on the circle; Figure 3). Most models then name the shortest cycle in their own list.

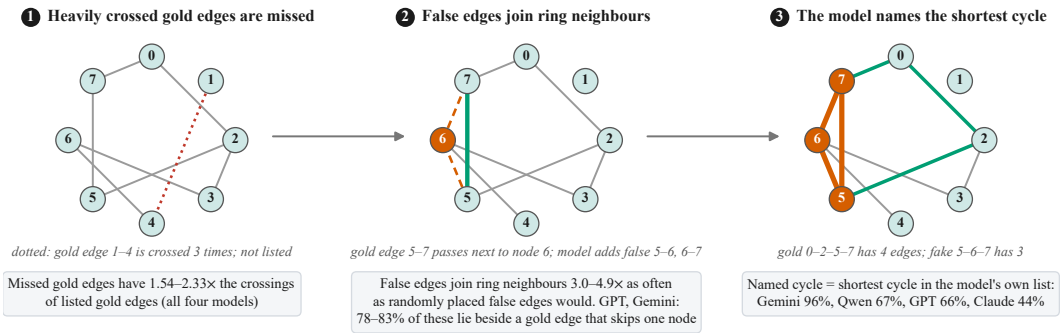


Figure 8: Walkthrough of this appendix on a toy ring with one gold 4-cycle $(0, 2, 5, 7)$. (1) The most-crossed gold edge (1–4) is the one dropped from the list. (2) The gold chord 5–7 passes close to node 6, and the model lists two ring-neighbour edges (5–6, 6–7) instead. (3) The list now holds the gold 4-cycle and a fake triangle $(5, 6, 7)$; the model names the shortest cycle of its list, which is the fake. Boxed numbers are the per-model results of Tables 11–13.

Hypothesis. If loss and competition have a geometric source, missed gold edges should be the more heavily crossed ones. Invented edges should concentrate on one visually salient pattern instead of falling evenly among non-edges.

Experiment. We run three analyses on stored Ring outputs:

- *Missed vs. kept edges.* For every gold edge missing from a model's Ring extraction (a *missed* edge) and every gold edge it retained (a *kept* edge), we compute that edge's Ring crossing count, pooled over all 500 cyclic graphs (Table 11). A per-edge logistic model with crossings, chord length, clearance, and endpoint degree, cross-validated by graph, gives each factor's odds ratio.
- *Ring-adjacent hallucinations.* We ask whether hallucinated (false, non-gold) edges connect two vertices that are consecutive on the ring (ring neighbours, Figure 3, panel 1), and compare with the exact per-graph chance rate (the share of non-edges that are ring-consecutive).
- *Shortest-cycle selection.* For every one-call (LIST+DECIDE) Ring "yes" answer whose named cycle is list-faithful, we check whether that named cycle is a shortest cycle of the model's own listed graph, regardless of whether it is correct (Table 13).

Table 11: Per-edge crossing geometry on Ring, pooled over the 500 cyclic graphs. Missed/Kept: mean crossing count of gold edges dropped vs. retained during extraction. Ring-adjacent: share of hallucinated (false, non-gold) edges connecting two ring-consecutive vertices; Chance is the exact per-graph ring-adjacent share of non-edges, weighted by each graph's hallucination count.

Model	Missed Crossings	Kept Crossings	Ratio	Hallucinations Ring-Adjacent	Chance
Claude	6.85	2.94	2.33×	1959/7452 (26.3%)	8.6%
GPT	6.79	4.42	1.54×	440/1935 (22.7%)	7.5%
Gemini	6.54	4.05	1.61×	676/1828 (37.0%)	7.6%
Qwen	6.34	3.72	1.70×	1340/5191 (25.8%)	8.0%

Observation.

1. Missed edges are the more crossed ones. Missed edges carry more crossings than kept edges for every model, by a factor of 1.54 to 2.28 (Table 11). Crossings remain the strongest per-edge

Table 12: Further locus results per model. Odds ratio: per standard deviation of crossings, from the per-edge logistic model with chord length, clearance, and degree controlled. Skip-one share: among ring-adjacent hallucinations, the share that co-occurs with a gold skip-one chord (a gold chord that skips one ring vertex) at an endpoint, on all graphs and on GITQA alone.

Model	Odds ratio (crossings)	Skip-one share, all	Skip-one share, GITQA
Claude	2.05	43.2%	37.1%
GPT	2.61	78.0%	76.7%
Gemini	4.37	83.4%	81.8%
Qwen	2.26	49.0%	44.3%

Table 13: Shortest-cycle selection rule on one-call (LIST+DECIDE) Ring lists. Rule Match: the named cycle (valid in the model's own list) is a shortest cycle of that list, regardless of correctness. DFS Match: the named cycle equals the cycle a gold-blind depth-first finder returns on the same list. s_A: V_{strict} when every fake cycle is strictly longer than the shortest surviving gold cycle; s_B: when a fake cycle is strictly shorter.

Model	Rule Match	DFS Match	s_A	s_B
Claude	192/437 (43.9%)	63/437 (14.4%)	3/9 (33.3%)	8/90 (8.9%)
GPT	323/492 (65.7%)	178/492 (36.2%)	124/134 (92.5%)	48/207 (23.2%)
Gemini	391/405 (96.5%)	243/405 (60.0%)	193/193 (100.0%)	5/61 (8.2%)
Qwen	197/295 (66.8%)	160/295 (54.2%)	25/53 (47.2%)	12/96 (12.5%)

predictor of a miss after chord length, clearance, and degree are controlled. The odds ratios are in Table 12, and the out-of-fold AUC is 0.69–0.80.

2. Invented edges favor ring neighbours. Hallucinated edges are ring-adjacent 3.0–4.9× more often than chance (Table 11). Most ring-adjacent hallucinations for GPT and Gemini co-occur with a gold skip-one chord (Table 12), and for those the skipped vertex sits at a median distance of 1.0% of the canvas diagonal from the chord (Figure 3, panel 4).

3. Selection follows the shortest cycle of the model's own list. A named cycle is *list-faithful* (F=1) when it is a simple cycle of the model's own list. Among list-faithful named cycles, the share that is a shortest cycle of the model's own list is highest for Gemini (96.5%) and lowest for Claude (43.9%), with GPT (65.7%) and Qwen (66.8%) in between (Table 13). For every model, the shortest-cycle rule matches the named cycle more often than depth-first order does.

Interpretation. The results are consistent with a specific reading of each branch:

- **Loss:** busy, highly crossed chords are disproportionately the edges dropped during extraction.
- **Competition:** invented edges default to the one visually salient pattern that a high-crossing rendering does not otherwise offer, a straight local hop along the Ring. These edges feed the fake cycles of the competition branch. For GPT and Gemini, most adjacent extras are also consistent with a clearance effect: a skip-one chord that passes close to the skipped vertex can look like two edges through it. So a ring-neighbour prior is not the only explanation.
- **Selection:** "report the shortest cycle in my list" is close to Gemini's general procedure, not a pattern seen only in its failures. It also describes GPT and Qwen better than depth-first order does.

Limitation. The analysis re-scores stored outputs and is observational. It does not change crossing count directly; the per-edge probe in Appendix T does that on the same images. Chord length, clearance, and crossings are correlated on a ring, so only the crossing coefficient is interpreted. Adjacent-hallucination rates are therefore not a pure neighbour prior, and whether a close chord is legible depends on node radius, which we did not measure.

H CYCLE-COMPETITION MEASUREMENT DETAILS

Takeaway. Suppose a model's edge list still contains a gold cycle. Does it also contain a fake cycle that is just as short, or shorter? For three models, it usually does. This holds for 94.9% of Claude's lists, 75.0% of Qwen's, and 69.5% of GPT's. It holds for only 30.3% of Gemini's. A simple search program that never sees the true graph is misled by the same fake cycles.

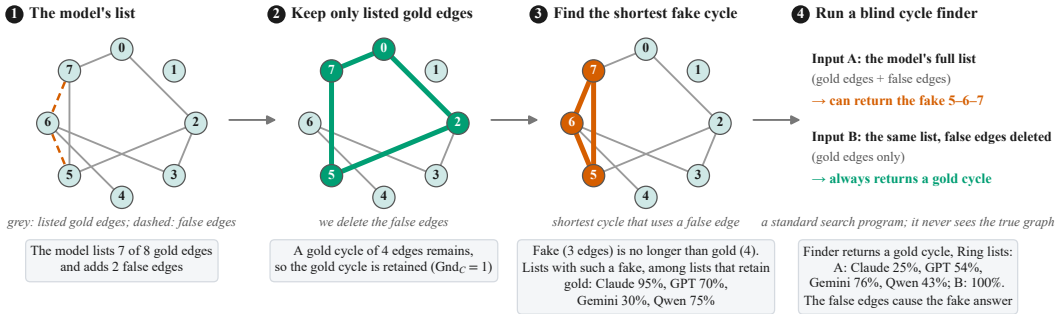


Figure 9: How cycle competition is measured, on the same toy list as Figure 8. (1) The model's list mixes gold and false edges. (2) Keeping only its gold edges gives the shortest gold cycle, g_{gold}=4, so Gnd_C=1. (3) The shortest cycle through any false edge gives g_{fake}=3; since $g_{\text{fake}} \leq g_{\text{gold}}$, a competitor exists. (4) A gold-blind finder, a plain search program that never sees the gold graph, is run twice. On the model's extracted list (gold plus false edges) it can return the fake cycle. On the same list with the false edges removed (gold edges only) it always returns a gold cycle. The false edges, not the search, produce the fake answer. Numbers are from Table 14.

Hypothesis. A *competitor* is a fake cycle in the model's list that is no longer than the gold cycle in that list. Competition should not be a quirk of how one model writes its answer. If it is a property of the lists themselves, two things should follow. First, competitors should be common in the lists. Second, a search program given the same lists should fail where competitors exist.

Experiment. We use only *retained-cycle* lists: lists that still contain a gold cycle. On each one we make three measurements.

- *Shortest gold cycle.* We keep only the listed edges that are in the true graph. We then find the shortest cycle among them exactly. Its length is g_{gold}.
- *Shortest fake cycle.* We take each false edge in the list, one at a time. We find the shortest path between its two ends that does not use it. The edge plus that path is the shortest fake cycle through the edge. The shortest over all false edges has length g_{fake}. A competitor exists when $g_{\text{fake}} \leq g_{\text{gold}}$.
- *Finder.* We run a fixed depth-first search program on the full list. It never sees the true graph. The finder uses the lists from the separate EXTRACT call; the two measurements above use the one-call LIST+DECIDE lists. As a check, we also run it on the list's true edges alone.

The true graph is used only to score results afterwards. Neither the model nor the finder ever sees it.

Table 14: Cycle competition in one-call LIST+DECIDE lists and finder behaviour on EXTRACT lists. Fake≤gold is measured among retained-cycle LIST+DECIDE items; Wrong+comp. among wrong retained-cycle LIST+DECIDE items. Finder columns: gold-blind depth-first finder on retained-cycle EXTRACT lists.

Model	Gnd_C	Fake≤gold	Wrong+comp.	Finder, Ring	Finder, Base
Claude	35.0%	94.9%	95.9%	25.1%	75.9%
GPT	87.8%	69.5%	95.5%	53.7%	93.7%
Gemini	55.4%	30.3%	100.0%	76.5%	99.5%
Qwen	42.4%	75.0%	81.8%	42.9%	83.8%

Observation. Table 14 shows three results.

1. Competitors are common. Claude has one in 94.9% of its retained-cycle lists, the most of any model. Gemini has one in 30.3%, the fewest. When the answer on a retained-cycle list is wrong, a competitor is almost always there: in 81.8–100.0% of cases.

2. The finder often fails on Ring lists. On Ring lists it returns a gold cycle in 25.1–76.5% of cases. On the cleaner Base lists it does so in 75.9–99.5%. Ring is lower for every model.

3. The finder itself works. Given only the true edges of each list, it returns a gold cycle every time (100%), for every model and layout.

Interpretation. For Claude, Qwen, and GPT, a list that keeps the gold cycle usually also offers a fake cycle that is at least as short. The finder's results follow the same order. The more often a model's lists contain a competitor, the less often the finder finds a gold cycle in them. The finder works on clean edges, so its failures come from the false edges, not from the search.

Limitation. We compare cycle lengths only. We do not measure which cycle looks more prominent in the picture. The insertion experiment (Appendix R) tests the effect of length directly, on text.

I Selection Fidelity and Decomposition of the Collapse

Takeaway. We ask two questions about each model's Ring lists. First, is a gold cycle still in the list? Second, if it is, does the model name it? The second question separates the models. Gemini names the gold cycle almost every time it is the shortest cycle in its list, and GPT comes close. Claude and Qwen often miss it, even when no fake cycle competes with it. In simulation, removing the invented edges raises accuracy more than any other single change.

Hypothesis. Suppose a model works in two steps. It first reads a list of edges from the picture. It then names the shortest cycle in that list. Its accuracy should then depend on the list, in four cases:

- **Loss:** no gold cycle is left in the list. Accuracy should be near 0%.
- **Case A:** the shortest gold cycle is shorter than every fake cycle. Accuracy should be near 100%.
- **Case T (tie):** the shortest gold cycle and the shortest fake cycle have the same length. Accuracy should be near 50%.
- **Case B:** a fake cycle is shorter than every gold cycle. Accuracy should be near 0%.

A model that departs from these numbers is not simply naming the shortest cycle. We call how closely it follows them its *selection fidelity*. If the two-step account is right, a simulation built on it should also reproduce each model's results.

Experiment. We run four analyses on the Ring answers.

- *Sort the lists into cases.* Every List+Decide Ring list goes into one of the four cases above (Appendix A). Within each case we measure named-cycle accuracy (Equation 6).
- *Split the drop from Base to Ring.* In the Ask condition, accuracy falls from Base to Ring. We split this fall into two parts. One part comes from the gold cycle surviving in fewer lists. The other part comes from lower accuracy on the graphs where it does survive. Survival is read from the separate Extract lists. Each part is weighted by the average of its Base and Ring values (midpoint weights).
- *Measure what the yes/no answer adds.* On the 500 cyclic graphs, we measure how much the yes/no answer tells us about whether the named cycle is correct. We report it as a share of the total uncertainty about named-cycle correctness: the mutual information divided by the entropy, $I/H(V_{\text{strict}})$.
- *Simulate lists, then remove causes.* From the Extract Ring lists we fit two rates for every pair of nodes: the chance that a true edge is missed, and the chance that a false edge is added. The rates depend on the edge's crossings, chord length, clearance from nodes, degree, whether the two nodes are ring neighbours, and their distance in the gold graph. The fit is cross-validated by graph. We draw 80 simulated lists per graph. Each simulated list is scored with the model's own accuracy

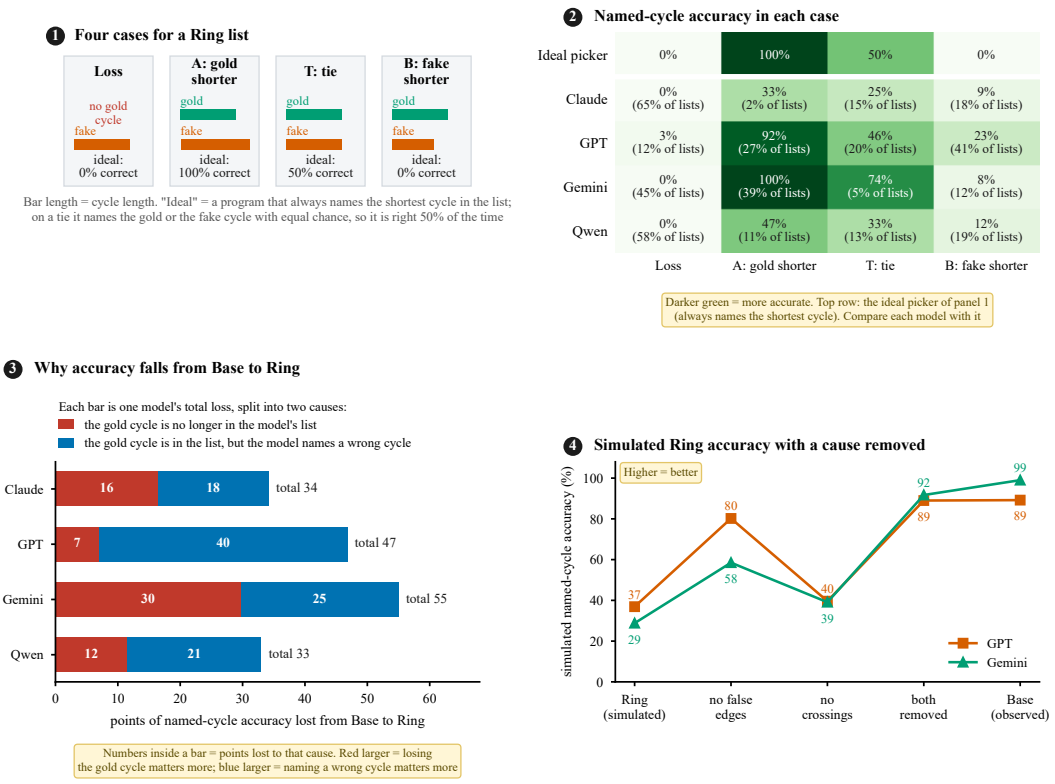


Figure 10: Walkthrough of this appendix. (1) Each LIST+DECIDE Ring list falls into one of four cases: no gold cycle is left (loss), the gold cycle is shorter than every fake cycle (A), they tie (T), or a fake cycle is shorter (B). The *ideal picker* is a program that always names the shortest cycle in the list; on a tie it names the gold or the fake cycle with equal chance. It therefore scores $0/100/50/0\%$ in the four cases. (2) Each model's named-cycle accuracy in each case, and the share of its lists that fall in that case (Table 15). (3) How many points of named-cycle accuracy each model loses from Base to Ring, split into two causes: the gold cycle is no longer in the list (red), or it is in the list but the model names a wrong cycle (blue) (Table 17). (4) Simulated Ring accuracy when one cause is removed at a time (Table 18).

in each case, estimated on other graphs (cross-fitted). We then rerun the simulation three more times: with every edge's crossings set to zero, with all invented edges removed, and with both.

Table 15: Selection outcomes by cell on one-call (LIST+DECIDE) Ring lists, 500 graphs. Each cell: number of lists (share of all lists) and V_{strict} within the cell. Loss: no gold cycle survives. A: every fake cycle strictly longer than the shortest surviving gold cycle; T: tie; B: a fake cycle strictly shorter. Ideal shortest-cycle selector: 0 / 100 / 50 / 0%.

Model	Loss	A	T	B
Claude	325 (65%): 0.0%	9 (2%): 33.3%	76 (15%): 25.0%	90 (18%): 8.9%
GPT	61 (12%): 3.3%	134 (27%): 92.5%	98 (20%): 45.9%	207 (41%): 23.2%
Gemini	223 (45%): 0.4%	193 (39%): 100.0%	23 (5%): 73.9%	61 (12%): 8.2%
Qwen	288 (58%): 0.3%	53 (11%): 47.2%	63 (13%): 33.3%	96 (19%): 12.5%

Observation. We report six results.

1. Accuracy in each case (Table 15). Gemini follows the shortest-cycle pattern closely: 100.0% in case A, 73.9% on ties, and 8.2% in case B. GPT is close to it: 92.5, 45.9, and 23.2%. Claude (33.3, 25.0, 8.9%) and Qwen (47.2, 33.3, 12.5%) are far below it. They often miss the gold cycle even

Table 16: Ideal, observed, and simulated results on Ring. Ideal selector: accuracy an ideal shortest-cycle selector would score on the model's own lists. Observed and Simulated: V_{strict} of the model and of the independent-edge simulation with cross-fitted selection fidelity. Gnd_C: share of lists that still contain a gold cycle, observed and simulated.

Model	Ideal selector	Observed	Simulated	Gnd_C observed	Gnd_C simulated
Claude	9.4%	6.0%	5.2%	35.0%	31.9%
GPT	36.6%	43.8%	37.4%	87.8%	88.1%
Gemini	40.9%	43.2%	28.9%	55.4%	59.1%
Qwen	16.9%	11.8%	9.0%	42.4%	39.6%

Table 17: Decomposition of the Base–Ring drop in ASK. This is a cross-call comparison of EXTRACT-list survival with ASK accuracy (Cohen's κ=0.48–0.73 with LIST+DECIDE). Share: part of the drop due to falling survival, with midpoint weights; the remainder is due to falling accuracy given survival. Bounds: Ring-only or Base-only weights for the survived cell. Midpoint, both: the Base not-survived cell is midpointed too (Gemini's value then rests on a single graph). Acc. given survival: accuracy among graphs where a gold cycle survives, Base → Ring. Base n: Base graphs in the not-survived cell, whose accuracy is therefore taken from Ring.

Model	Share	Bounds	Midpoint, both	Acc. given survival	Base n
Claude	48%	25–72%	41%	41 → 14%	36
GPT	15%	10–20%	10%	90 → 47%	8
Gemini	54%	41–67%	20%	99 → 65%	1
Qwen	35%	15–56%	21%	44 → 15%	55

Table 18: Counterfactual predicted accuracy on Ring for all four models. Predicted: accuracy simulated with a per-image error factor and cross-fitted selection fidelity, the model used for the counterfactuals (Table 16 uses independent edges, hence 36.9 vs. 37.4% for GPT). No invented edges / No crossings / Neither: the counterfactual named in the column. Base: observed Base accuracy.

Model	Predicted	No invented edges	No crossings	Neither	Base
Claude	5.3%	11.5%	9.5%	22.4%	39.0%
GPT	36.9%	80.2%	39.6%	89.0%	89.2%
Gemini	28.8%	58.5%	39.1%	91.7%	99.0%
Qwen	9.0%	19.3%	14.8%	34.3%	41.8%

in case A, where it is the shortest cycle in the list. Cases A and B repeat the last two columns of Table 13.

2. Comparison with a perfect shortest-cycle picker (Table 16). We compute the accuracy that a program naming the shortest cycle of each list would reach on the model's own lists. GPT (43.8% vs. 36.6%) and Gemini (43.2% vs. 40.9%) score above that program. So neither model simply names the shortest cycle. For example, GPT names a gold cycle in 23.2% of the lists where a fake cycle is strictly shorter; the program scores 0% there. Claude (6.0% vs. 9.4%) and Qwen (11.8% vs. 16.9%) score below it.

3. Where the Base-to-Ring drop comes from (Table 17). The *Share* column gives the part of the drop that comes from the gold cycle surviving less often. The rest comes from lower accuracy when it survives. Accuracy given survival falls for every model; for GPT it falls from 90% to 47%. One cell needs a convention. On Base, a few graphs have no surviving gold cycle; for Gemini there is only one. For all four models we therefore take that cell's accuracy from Ring.

4. What the yes/no answer adds. It removes 0.7% (Claude), 0.0% (GPT), 15.3% (Gemini), and 0.1% (Qwen) of the uncertainty about named-cycle correctness on the 500 cyclic Ring items. GPT says "yes" on every graph, so its answer carries no information and its value is exactly 0. Only Gemini says "no" often enough for its value to mean anything.

5. Simulation (Table 16). The simulation reproduces how often the gold cycle survives, for all four models. It also reproduces accuracy for Claude and Qwen. For GPT and Gemini it predicts lower accuracy than observed. It ranks graphs from easy to hard reasonably well (AUC 0.76–0.79). Errors do cluster by picture. The spread of error counts per graph is 1.1–$1.9\times$ what independent edges would give, after trimming outliers. But adding a per-picture term to the simulation changes its predictions little.

6. Removing causes (Table 18). For every model, removing invented edges raises predicted accuracy more than removing crossings. With both removed, GPT and Gemini come close to their Base accuracy (89.0 vs. 89.2% and 91.7 vs. 99.0%). Claude and Qwen stay below Base even with both removed (22.4 vs. 39.0% and 34.3 vs. 41.8%). This fits their weaker selection even on clean lists (Appendix Q).

Interpretation. How the Base-to-Ring drop splits differs by model. (This split describes the drop from Base to Ring. It answers a different question from the gold-list test of Appendix Q.) For GPT (15% from survival) and Qwen (35%), most of the drop comes *after* survival: the gold cycle is in the list, but the model names something else. For Claude (48%) and Gemini (54%), losing the gold cycle and naming the wrong cycle contribute about equally. Once the gold cycle has survived, errors have two sources. The first is fake cycles that compete with the gold one. The second, for Claude and Qwen, is imperfect selection even when nothing competes. Invented edges matter most in the simulation because each one can create a short fake cycle. So competition is a failure of extraction precision, meaning edges are listed that are not there. It is not a separate weakness in telling cycles apart. The near-zero values in result 4 mostly reflect the test set: every graph has a cycle, and most models almost always say yes. The count gaps in Table 2 are the real evidence about yes/no answers.

Limitation. The four cases use the LIST+DECIDE list, but the Base-to-Ring split uses separate EXTRACT calls. The two calls agree on survival only moderately (Cohen's κ=0.48–0.73). The simulation treats edges as independent once their geometry is known. It does not model errors that cluster, or invented edges that follow a pattern. That is where it under-predicts GPT and Gemini. Several cases in Table 15 are small; for example, Claude's case A holds only 9 lists. Accuracy in a case with fewer than 30 lists is exploratory, not a precise estimate.

J RULING OUT CHANCE, SCORING, AND GENERATIVE ARTIFACTS

Takeaway. We checked three ways in which the Ring results could be misleading. None of them explains the collapse. First, whether a gold cycle survives in a list depends mostly on how many gold edges the model listed, not on which ones. Second, the drop from Base to Ring is not caused by a strict scoring rule: it remains under looser rules. Third, lists simulated from simple per-edge error rates look like the real lists.

Two terms. Gnd_C is the share of a model's LIST+DECIDE Ring lists that still contain a gold cycle after the false edges are deleted (Figure 11, panel 2). The *Base-to-Ring drop* is the model's named-cycle accuracy on Base minus its accuracy on Ring, on the same graphs.

Hypothesis. All three checks reuse stored answers; no model is called again.

- *Chance.* Ring graphs are dense, and a model lists only some of the gold edges. Whether a gold cycle survives might then depend only on how many edges are listed. If so, picking the same number of gold edges at random should give the same survival rate as the model's lists.
- *Scoring.* Named-cycle scoring is strict. If the Base-to-Ring drop came from that strictness, looser rules should shrink or remove it.
- *Simulation.* Appendix G argues that crossings and invented neighbour edges shape the lists. If so, lists simulated from per-edge rates, combined with each model's selection behaviour, should reproduce how often the gold cycle survives. The rates depend on crossings, chord length, clearance, degree, ring adjacency, and gold distance.

Experiment. All three checks re-score the stored Ring LIST+DECIDE and ASK outputs.

- *Random-miss baseline.* Take one of the model's lists and count its gold edges; call the count k. Now imagine picking k gold edges at random instead. We compute exactly how likely such a

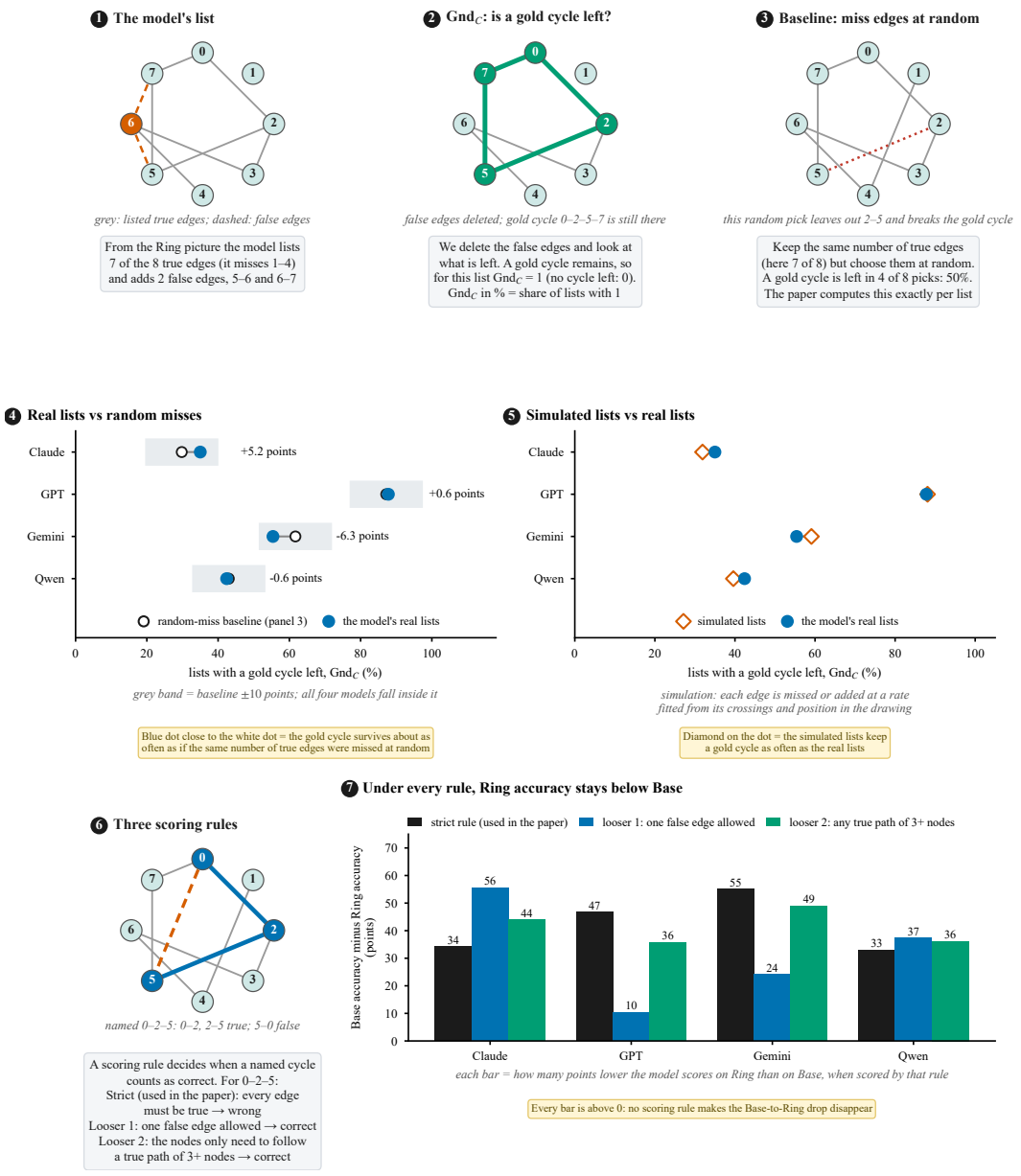


Figure 11: Walkthrough of this appendix on the toy ring. (1) A model's edge list: 7 of the 8 true edges plus 2 false edges. (2) What Gnd_C measures: delete the false edges and check whether a gold cycle is left. Here one is, so this list scores 1; Gnd_C in % is the share of lists that score 1. (3) The random-miss baseline: keep the same number of true edges, but choose them at random, and ask how often a gold cycle is left. (4) For every model, the real lists keep a gold cycle about as often as the random-miss baseline, within ± 10 points (Table 19). (5) Lists simulated from per-edge error rates keep a gold cycle about as often as the real lists. (6) A scoring rule decides when a named cycle counts as correct. The strict rule used in the paper needs every edge to be true; looser rule 1 allows one false edge; looser rule 2 only needs the named nodes to follow a true path of 3 or more nodes. (7) Under every rule, each model still scores lower on Ring than on Base (Table 20).

random pick is to contain a gold cycle, using inclusion–exclusion over the gold graph's simple cycles. We compare this with the observed survival rate using a paired equivalence test (two one-sided tests) with a margin of ± 10 points.

- *Looser scoring.* A *scoring rule* decides when a named cycle counts as correct. The *strict rule* is the one used in the rest of the paper: every edge of the named cycle, including the closing edge, must be in the true graph, and no node may repeat. We re-score every answer under two looser rules. *Looser rule 1* also counts a named cycle as correct if at most one of its edges is false. *Looser rule 2* counts it as correct if its nodes, in order, follow a path of true edges through at least three nodes; the closing edge is not checked. For example, take the named cycle 0–2–5, where 0–2 and 2–5 are true edges and the closing edge 5–0 is false. The strict rule marks it wrong. Both looser rules mark it correct (Figure 11, panel 6). For each rule we then recompute the Base-to-Ring drop.
- *Simulation.* We use the simulation of Appendix I. Its rates are fit on EXTRACT Ring lists only. They never see named-cycle accuracy or the split into loss and competition.

Table 19: Observed Gnd_C on Ring compared with the chance baseline and with simulated lists. Difference: observed minus chance, with 90% CI from the paired test.

Model	Observed	Chance	Difference [90% CI]	Simulated
Claude	35.0%	29.8%	+5.2 [+2.4, +8.0]	31.9%
GPT	87.8%	87.2%	+0.6 [−1.2, +2.3]	88.1%
Gemini	55.4%	61.7%	−6.3 [−8.4, −4.1]	59.1%
Qwen	42.4%	43.0%	−0.6 [−3.1, +2.0]	39.6%

Table 20: Base–Ring gap in V_{strict} (in points) under looser scoring rules. All gaps remain significant ($p<10^{-12}$).

Model	One wrong edge allowed	Any gold path accepted
Claude	55.6	44.0
GPT	10.4	35.8
Gemini	24.2	49.0
Qwen	37.4	36.2

Observation. We report three results.

1. Survival matches the random-miss baseline within ± 10 points for all four models (Table 19). Claude keeps a gold cycle slightly more often than the baseline ($+5.2$ points), and Gemini slightly less often (-6.3 points).

2. Looser scoring changes the size of the Base-to-Ring gap, but removes it for no model (Table 20). With one wrong edge allowed, the gap is still 10.4 points (GPT) to 55.6 points (Claude). Every gap remains significant ($p<10^{-12}$).

3. Simulated lists reproduce survival closely (Table 19, last column).

Interpretation. In these dense graphs, whether a gold cycle survives depends mostly on how many gold edges remain. We see no sign of a rule that deletes cycle edges in particular. This is the reading used in Sections 6 and 8. The match is not tight enough to call it "the same as chance" at a stricter ± 5-point margin for Claude and Gemini, because their 90% intervals extend beyond ± 5 points. *Which* edges are dropped is a separate question, answered in Appendix G: the missed edges carry more crossings. The scoring rule does not cause the Base-to-Ring collapse. Geometry explains the lists, and each model's selection behaviour on those lists explains the named cycle, to the extent shown in Appendix I.

Limitation. These checks re-score the same Ring lists; they are not a new sample. The simulation tests our account of how the lists are formed.

K WHICH DRAWINGS FAIL: THE DIFFICULTY SCORE

Takeaway. D_{fin} is built from the crossing count, the node count, and the edge count on the drawing. It separates Base from Ring and keeps GPT with Gemini, Claude with Qwen. Inside one layout the ranking is weak.

Experiment. $N_\times$ is an integer: visible strokes on the stored Base PNG, and exact straight-chord intersections on the Ring drawing. Equation 1 is the score used in Section 6.3. On the 500 cyclic graphs and both layouts (4,000 model outcomes), pooled logistic regression with one intercept per model gives

$$P_m(V_{\text{strict}}{=}1) = \big(1 + \exp(-(a_m - 0.86\, D_{\text{fin}}))\big)^{-1}, \tag{7}$$

with $(a_{\text{Claude}}, a_{\text{GPT}}, a_{\text{Gemini}}, a_{\text{Qwen}}) = (1.70, 4.33, 4.67, 1.77)$. The unrounded weights are $+0.349$ and -0.264; rounding them to $+0.35$ and -0.25 changes cross-validated AUC by less than 0.001, so D_{fin} can be computed by hand. The edge term carries a negative weight: at fixed n and $N_\times$, more edges means more independent cycles and therefore more admissible cycles to name, which makes the item easier.

Protocol. Because the weights are chosen on the same data, we separate selection from evaluation. Twenty percent of the *graphs* (94 of 500) are set aside before any fitting. On the remaining 380 we run five rounds of 70/30 train/test splits, always splitting by graph so that no drawing appears on both sides. The held-out set is scored once, at the end. Cross-validated (test-fold) AUC is 0.875 ± 0.006 across the five rounds against a training-set AUC of 0.875 — a train-minus-test gap of 0.0002, so the fit is not memorising. On the held-out graphs the pooled AUC is 0.889, and by model 0.78 / 0.82 / 0.93 / 0.79. A two-term variant without the edge term, $\ln(1 + N_\times) + 0.09\, n$, scores 0.871 and 0.882 under the same protocol, so the edge term adds about 0.005 AUC.

What the pooled number does and does not mean. The pooled AUC is inflated by two separations that have nothing to do with ranking drawings within a condition: model identity, and the Base/Ring layout gap in total crossings. Within a single model *and* a single layout the score is much weaker — Base 0.66 / 0.71 / 0.68 / 0.58 and Ring 0.53 / 0.64 / 0.80 / 0.68, with Claude on Ring at chance. Equation 1 is therefore a cross-condition difficulty ordering. It does not rank which Ring drawing fails.

Reading the score. D_{fin} runs from 0.35 to 8.0 on this collection. The crossover at which a model is more likely wrong than right is $D_{\text{fin}} \approx 2.0$ for Claude and 2.0 for Qwen against 5.0 for GPT and 5.4 for Gemini: the stronger pair tolerates roughly two and a half times the difficulty before falling below even odds. Figure 12 shows that split.

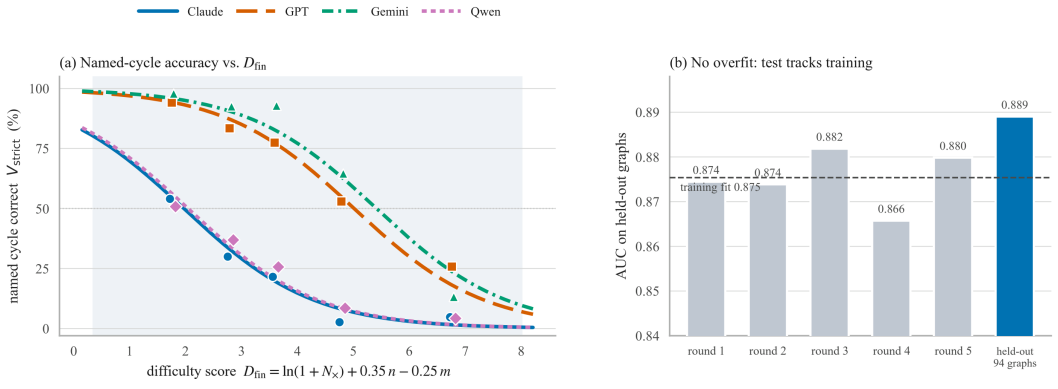


Figure 12: (a) Named-cycle accuracy versus D_{fin} (Equation 1), four models on one axis. Curves: $P_m = (1 + \exp(-(a_m - 0.86\, D_{\text{fin}})))^{-1}$. Points: observed V_{strict} by D_{fin} quintile, Base and Ring pooled. (b) Held-out AUC across the five 70/30 rounds and on the held-out 94 graphs, against the training-set fit (dashed). Test tracks training, so the three-term fit is not overfitting. The left of the axis is mostly Base and the right mostly Ring, so part of the pooled drop tracks the layout gap in total crossings.

L FULL MODEL-BY-MODEL FAILURE PROFILES

Takeaway. The four models fail at different stages. Gemini mainly loses the gold cycle but selects well, GPT keeps the gold cycle but is pulled off it by fake cycles, and Claude and Qwen show a mix of extraction loss and a residual error on the clean gold list.

Hypothesis. If the branch decomposition, extraction quality, selection fidelity, and the clean-input ablations measure different stages, combining them should give distinct per-model profiles instead of one shared "models fail under stress" story.

Experiment. For each model we combine Tables 4–5, Table 13, the gold-list ablation (Appendix Q), and the deterministic finder on clean-layout lists (Appendix P).

Table 21: Headline metrics behind the per-model profiles. Loss: share of wrong LIST+DECIDE outputs on Ring in which the gold cycle was lost. Ring precision is as in Table 4. F, all outputs: faithfulness over all LIST+DECIDE Ring outputs (Table 37); Table 5 reports it among "yes" answers only. Gold-list: accuracy when the clean gold list is given as text.

Model and main pattern	Loss	Ring precision	F, all outputs	Gold-list
Claude: extraction loss plus a residual error on the gold list	69.1%	0.50	88.6%	81.8%
GPT: competition from a retained cycle	21.0%	0.87	98.6%	100.0%
Gemini: near-ideal selection, loss under stress	78.2%	0.85	99.4%	100.0%
Qwen: mixed extraction and decision errors	65.1%	0.68	73.8%	65.2%

Observation.

Claude. Ring precision is the lowest of the four models. Even given the clean gold list as text, Claude reaches only 81.8%. On Base drawings, its listed real edges still contain a gold cycle 92.8% of the time, but it names one on only 39.0% (a different call), so the Base miss is not "the gold cycle never entered the list". On its own Base lists, a gold-blind finder names a gold cycle on 70.4% of graphs against Claude's 39.0% on the separate ASK call (a cross-call comparison). Its high faithfulness over all LIST+DECIDE outputs (88.6%) therefore reflects consistency with a heavily polluted list, not successful grounding.

GPT. GPT has the highest Ring precision (0.87) and retains a gold cycle in 79.0% of its wrong outputs. In 95.5% of that branch, a fake cycle no longer than the retained gold cycle was available to compete with it. GPT resists a strictly shorter fake more often than the other models (22.8%). Its competition is driven by false edges: removing invented edges in the simulation of Appendix I lifts its predicted accuracy from 37.8% to 81.3%. On the clean gold list, which contains no fake cycle, it reaches 100.0%.

Gemini. Gemini reaches 99.0% named-cycle accuracy on Base. On Ring its loss branch is 78.2% of wrong outputs and it answers "no cycle" on 80 drawings in ASK. Even so, it names a shortest cycle of its own list on 96.5% of list-faithful named cycles (391/405).

Qwen. Loss accounts for 65.1% of wrong outputs, alongside low Ring precision (0.68), the lowest faithfulness over all LIST+DECIDE outputs (73.8%), and a gold-list score of 65.2%. A gold-blind finder beats Qwen on its own Ring lists (18.1% vs. frozen 6.6%) and, as a cross-call comparison against the separate ASK call, on Base lists (74.6% vs. 41.8%).

Interpretation. The profiles separate three patterns: missing the gold cycle (Gemini, and the loss branch in general), fake edges in the list (GPT), and a residual error on the clean typed list for Claude and Qwen (a direct measurement whose cause is unresolved). A single accuracy number would place Gemini and GPT together and Claude and Qwen together on Ring, while hiding that the reasons differ within each pair.

Limitation. The profiles are syntheses of stored-output analyses and one clean-input ablation per model. They describe these snapshots on this graph family and are not claims about the model lines.

PART IV — ROBUSTNESS CHECKS, TEXT-LIST INSERTION, AND IMAGE PROBES

This part tests the main findings from other angles. Table 22 is the roadmap.

Reminder of terms. Ring is the equal-angle drawing. Base is the Graphviz drawing. ASK asks for yes/no and a named cycle from the picture. EXTRACT asks for the edge list only. LIST+DECIDE lists the edges and then decides, in one call. R_G is a text-only decision on the clean gold list. Gnd means a gold cycle survives in the list. Appendix A gives the full definitions.

Table 22: Roadmap of Part IV: the question each appendix asks and its short answer.

Appendix	Question	Short answer
M: Readability probes	Is the Ring picture simply unreadable?	No. Degree comparison stays high. Path length falls for Claude and Qwen.
N: Human reference	Can people pick the gold cycle from two options?	Yes, on 96.2% of the 500 drawings.
O: Path replication	Do the cycle findings repeat for shortest paths?	Partly. Loss dominates for Claude, Gemini, and Qwen. GPT makes few errors $(42/210)$, and most of those are also loss.
P: Text-only fixes	Does repairing the frozen list as text help?	Mostly no. Shuffling changes nothing. A finder helps clearly only for Qwen.
Q: Gold-list substitution	Does the deficit sit in extraction?	For GPT and Gemini, yes. Claude and Qwen keep a gap on the clean list.
R: Edge insertion	Does edge length steer selection on text?	Yes. The shorter cycle is named far more often.
S: Chord on trees	Does this hold with only one possible cycle?	GPT and Gemini stay exact. Claude and Qwen drop on the long cycle.
T: Per-edge probes	Are crossed gold edges accepted less often?	Clearly for Qwen, weakly for Claude, not for GPT or Gemini.
U: Crossing bins	Does total clutter explain the collapse?	No. Crossings on the gold cycle track accuracy better.
V: Full table	Where are all results collected?	Table 37, by model, layout, and condition.

M READABILITY PROBES: DEGREE AND PATH LENGTH

Takeaway. The Ring pictures are not unreadable. All four models still compare degrees well. Claude and Qwen are weak at shortest-path length, though.

Hypothesis. If the equal-angle drawing is hard to read in general, models should also fail other graph questions on the same images. They should not fail only on named cycles.

Experiment. We ask two questions on the same Ring drawings. Neither needs a named cycle. The degree probe ($V_{\deg}$) asks which of two labelled vertices has the higher degree (chance 50%, $n{=}140$ items). The path probe (V_{SP}) asks how many edges lie on the shortest path between two vertices ($n{=}210$). Figures 13–14 show examples. Table 23 adds the human result of Appendix N for context.

Observation. Table 23 and Figure 15 give the headline numbers.

Degree. Every model scores 93.6–96.4%. On hard pairs ($|\Delta \deg| {=} 1$), scores are 87.1–92.9%.

Shortest path. Path accuracy stays well above cycle V_{strict} for every model. Claude scores 45.2% vs. 4.8%, GPT 86.7% vs. 42.4%, Gemini 80.0% vs. 44.0%, and Qwen 49.0% vs. 8.8%. Claude and Qwen are also weaker on Base paths (70.0% and 58.6%).

By distance. Table 24 splits path accuracy by distance. Claude is weak at every distance (57.1, 38.6, and 40.0%). Qwen is weak at distances 1 and 3 (52.9% and 20.0%) but reaches 74.3% at distance 2. GPT and Gemini stay between 75.7% and 94.3%.

Table 23: Independent readability tests on the same equal-angle drawings (70 cyclic graphs; 140 degree items; 210 shortest-path items, distances 1–3). Human 2AFC (chance 50%) shows the drawings support this discrimination; it is not comparable to V_{strict}. It has n=500 answers.

Model	Degree (V_{deg})	SP length (V_{SP})	Cycle V_{strict}	Human 2AFC
Claude	93.6%	45.2%	4.8%	—
GPT	96.4%	86.7%	42.4%	—
Gemini	95.7%	80.0%	44.0%	—
Qwen	95.0%	49.0%	8.8%	—
Human (2AFC)	—	—	—	96.2%

Table 24: Visual shortest-path length accuracy (V_{SP}) on Ring, by gold distance (70 items per distance).

Model	Distance 1	Distance 2	Distance 3
Claude	57.1%	38.6%	40.0%
GPT	94.3%	87.1%	78.6%
Gemini	78.6%	85.7%	75.7%
Qwen	52.9%	74.3%	20.0%

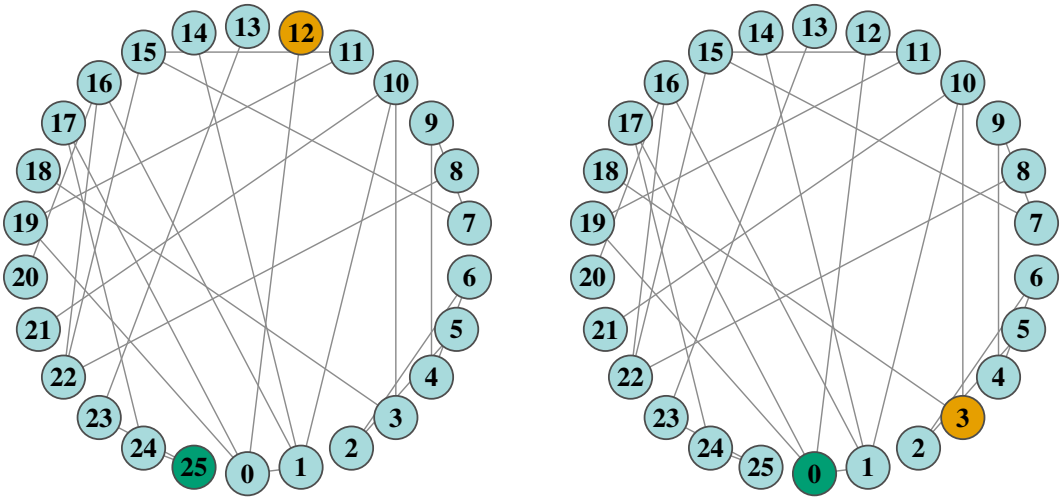


Figure 13: Visual degree comparison queries on an illustrative example ring, g_{00} (not from the scored 70-graph probe set). Green: higher-degree node; yellow: lower-degree node. Left: hard pair $|\Delta \deg|=1$ (nodes 25 vs. 12; degrees 2 vs. 1). Right: easy pair $|\Delta \deg|\geq 2$ (nodes 0 vs. 3; degrees 4 vs. 2).

Interpretation. The pictures are readable at the level of degree. Degree stays high for all four models, and GPT and Gemini also read path length well. Claude and Qwen do not. Both are weak even on single-edge items, and Qwen falls to 20% at distance 3. So the picture is not globally unreadable, but not every model can read its edges. Naming a cycle is still harder than comparing degrees. These results fit earlier reports that circular drawings bias MLLM bridge judgments (Brand et al., 2025), that accuracy transfers poorly across layouts (Babaiee et al., 2025), and that Graphviz layout choice is not neutral for cycle questions (Wei et al., 2026). They do not replicate those studies.

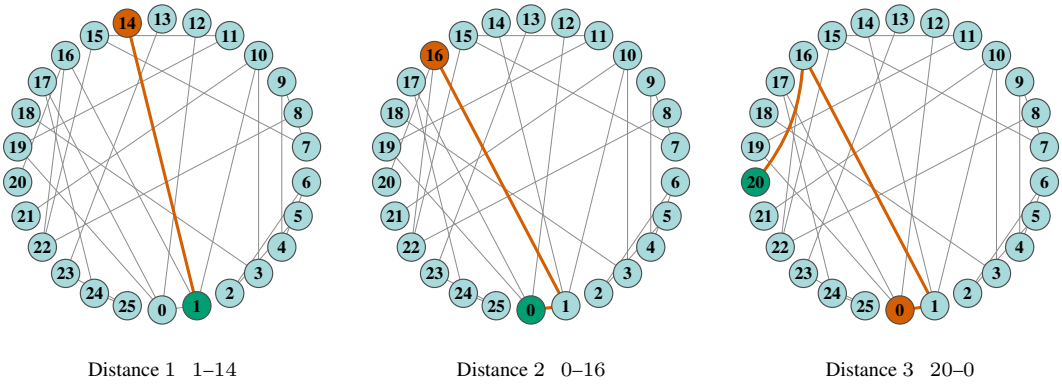


Figure 14: Visual shortest-path length queries on an illustrative example ring, g_{00} (not from the scored 70-graph probe set). Green: source; orange: target and gold shortest path. Left to right: distance 1 (1–14), distance 2 (0–16), and distance 3 (20–0); one path edge is drawn curved where a straight chord would pass under a node. The model reports an integer length.

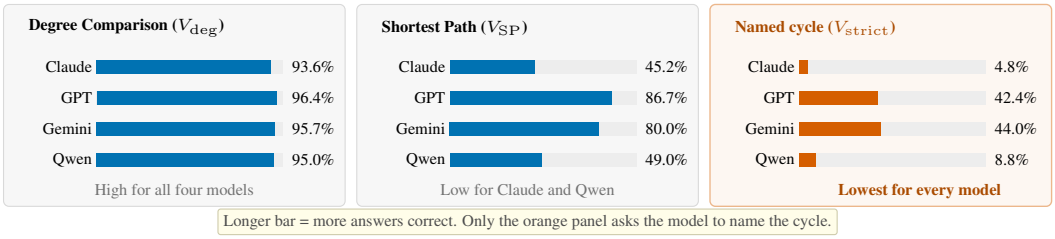


Figure 15: Table 23 at a glance, on one shared 0–100% scale. Degree and shortest-path bars (blue) stay above the named-cycle bar (orange), which collapses for all four models and is lowest for Claude. Only the named cycle requires naming a structure that is checked against the gold graph. Degree stays high for every model. Path length is weak for Claude at every distance and for Qwen at distances 1 and 3, including single-edge items (Section 7.1).

Limitation. Neither probe requires following information around a cycle, as a named cycle does. They cannot show which geometric property drives the cycle-specific deficit. Appendices G, K, and T address that, using per-edge geometry, the difficulty score, and a per-edge image probe.

N HUMAN REFERENCE ON THE RING DRAWINGS

Takeaway. Given two options, people pick the gold cycle on 96.2% of the 500 drawings. So the drawings are legible for a comparison task. This is not a model score, and it does not measure unaided cycle finding.

Hypothesis. If the Ring drawings are legible, people should reliably tell a gold cycle from a close foil on them. If model difficulty came from the drawings themselves, the drawings people miss should also be the ones models miss.

Experiment. Five volunteer raters saw one Ring drawing at a time. They chose which of two listed node sequences is a gold cycle (a two-alternative forced choice, 2AFC; chance 50%). The foil was one of two kinds. A *premature* foil is a gold path missing its closing edge. An *adjacent* foil is completed by a non-edge between two ring neighbours (Figure 3). Option order was counterbalanced. Each rater received a disjoint block of 100 drawings. The 500 study drawings are the 500 graphs of the collection, so every graph has one human answer.

We keep each rater's first answer per drawing and re-score it against the gold graph; the re-score agrees with the recorded answer on every item. The median response time is 25.1 seconds. This gives 500 answers, one on each of the 500 graphs. We compare human accuracy across foil types, ring orders, and sources, and compare per-graph human correctness with model V_{strict} on the same 500 graphs.

Table 25: Human identification of the gold cycle on the Ring drawings (500 graphs of the collection, one answer each), with Wilson 95% intervals.

Subset	Answers	Correct
All graphs	500	96.2% [94.1, 97.6]
Premature foil	345	94.8% [91.9, 96.7]
Adjacent foil	155	99.4% [96.4, 99.9]
Cycle-order rings	90	96.7% [90.7, 98.9]
Node-id rings	410	96.1% [93.8, 97.6]
GITQA graphs	250	97.2% [94.3, 98.6]
Generated graphs	250	95.2% [91.8, 97.2]

Observation. Table 25 shows humans correct on 96.2% [94.1, 97.6] of the 500 graphs ($481/500$). Each of the five raters answered 100 graphs, and their accuracy ranged from 89.0% to 99.0% (heterogeneity p=0.00067).

Foil type. Premature foils are harder than adjacent ones (94.8% vs. 99.4%).

Ring order and source. Both matter little (95.2–97.2%).

Match with models. Per graph, human correctness shows no detectable link to model correctness (Spearman ρ from $+0.00$ to $+0.04$, all $p \geq 0.32$). With only 19 human errors, this comparison has little power.

Interpretation. Raters can pick the gold cycle when shown two candidates. So the drawings are legible enough for that task. This does not show that a person, or a model, could find the cycle unaided. The 96.2% is not a model score and cannot be compared with 4.8%. The 19 errors are too few to test whether the drawings people miss are the ones models miss. The 2AFC rate shows the drawings support this discrimination; it is not comparable to V_{strict}. It does not show that model failures are unrelated to image difficulty.

Limitation. Raters saw candidate sequences, while models produced a cycle unaided. The two tasks are related, not identical. Raters did not share drawings, so inter-rater agreement cannot be computed. Rater accuracy also differs.

O PATH-LEVEL REPLICATION OF THE CYCLE FINDINGS

Takeaway. The cycle findings partly repeat for shortest paths. For Claude, Gemini, and Qwen, wrong named paths are mostly loss: the true path never enters the list. GPT makes few errors (42/210), and most of those are also loss. On Base, Claude's and Qwen's lists support more than they state.

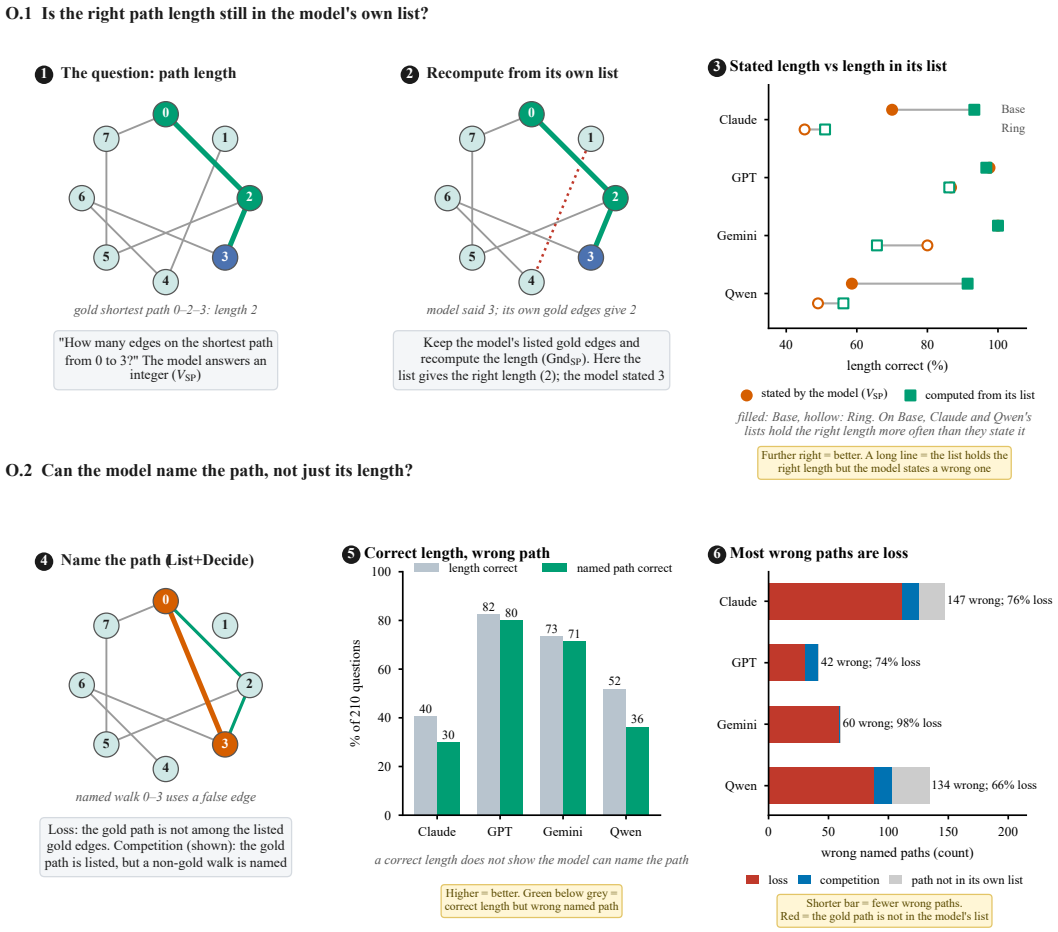


Figure 16: Walkthrough of this appendix on the toy ring, for the query from node 0 to node 3 (gold shortest path 0–2–3, length 2). **O.1** (1) The model states a path length (V_{SP}). (2) We keep the model's own listed gold edges and recompute the length (Gnd_{SP}); in the example the list supports length 2 although the model said 3. (3) Stated vs. recoverable accuracy per model (Table 26); on Base, Claude's and Qwen's lists hold the right length far more often than they state it. **O.2** (4) In one LIST+DECIDE call the model lists edges and names a path. A wrong named path is *loss* if the gold path is not among the listed gold edges, and *competition* if it is listed but a non-gold walk is named (shown: 0–3 through a false edge). (5) A right length is more common than a right named path. (6) Most wrong named paths are loss (Table 27).

O.1 DOES THE CORRECT PATH LENGTH SURVIVE EXTRACTION?

Hypothesis. If the cycle deficit were only a problem of stating the answer, the correct path length should be recoverable from a model's own extracted edges, even when its stated length is wrong. This is the path version of gold-cycle retention (Gnd) for cycles (Section 6).

Experiment. We re-score stored material, with no new model calls. We use the EXTRACT lists behind Table 4 and the same 210 gold source–target pairs as V_{SP}. Gnd_{SP} is the shortest-path length

on the gold-filtered, true-positive-only subgraph, compared with gold (Table 26; Appendix A defines the metric). V_{SP} and its extracted-graph counterpart $\mathrm{Sym}_{\mathrm{SP}}$ score only the reported answer. Neither asks whether the length was recoverable from the extraction.

Table 26: Shortest-path recoverability from stored lists, computed with no new API calls from the extraction lists behind Table 4 and the 210 V_{SP} queries. $\mathrm{Sym}_{\mathrm{SP}}$: length on the full extracted graph matches gold. $\mathrm{Gnd}_{\mathrm{SP}}$: length on the gold-filtered true-positive subgraph matches gold. Gemini Base list-based columns cover 159 items.

Model	**Base**			**Ring (Equal-Angle)**		
	V_{SP}	$\mathrm{Sym}_{\mathrm{SP}}$	$\mathrm{Gnd}_{\mathrm{SP}}$	V_{SP}	$\mathrm{Sym}_{\mathrm{SP}}$	$\mathrm{Gnd}_{\mathrm{SP}}$
Claude	70.0%	92.4%	93.3%	45.2%	55.2%	51.0%
GPT	97.6%	96.7%	96.7%	86.7%	84.8%	86.2%
Gemini	100.0%	100.0%	100.0%	80.0%	66.2%	65.7%
Qwen	58.6%	91.4%	91.4%	49.0%	56.7%	56.2%

Observation. Table 26 gives the numbers.

GPT and Gemini. On Base, $\mathrm{Gnd}_{\mathrm{SP}}$ stays within a few points of V_{SP}. On Ring, GPT's stated length matches what its list supports. Gemini states the correct length more often than its own extraction supports (80.0% vs. 66.2%).

Claude and Qwen on Base. Their lists usually keep the right path length even when the stated answer is wrong. Claude's $\mathrm{Gnd}_{\mathrm{SP}}$ is 93.3% against a stated 70.0% (gap 23.3 points). Qwen's is 91.4% against 58.6% (gap 32.8 points).

Interpretation. GPT's path errors trace back to its extracted lists, because what it states matches what they support. On clean drawings, Claude and Qwen show the reverse: their lists support more than they state. This is the path version of the cross-call residual seen for cycles on Base drawings (a hint, not proof; Appendix P).

Limitation. Recoverable means the right length is present among the true-positive edges. It does not mean the model could point to the walk. The next subsection tests naming directly.

O.2 Named path, not just length

Hypothesis. A correct integer length may not be a verified answer. If the cycle split carries over, a named s–t path should fail in the same two ways. In loss, the gold path never enters the listed true-positive edges. In competition, it does, but the model names a list-faithful walk that is not gold.

Experiment. We repeat the list-then-decide protocol for shortest paths. It uses the same Ring drawings and the same 210 distance-1/2/3 queries as Table 23. The model lists observed edges, then reports the length and names an s–t path (Appendix D). This is C_{SP}, a separate call from the visual probe V_{SP}. A walk counts as named if it is a simple gold s–t path of gold length. $\mathrm{Gnd}_{\mathrm{SP}}$ and F are scored as for cycles.

Table 27: Named-path shortest-path check on the Ring drawings (210 items, one list-then-decide call each; this is C_{SP}, not the visual V_{SP} probe of Table 23). Integer length: reported s–t distance on this call. Named walk: that walk is a simple gold s–t path of gold length. Loss and competition shares are among *wrong* named paths: Loss $\mathrm{Gnd}_{\mathrm{SP}}$=0; Competition $\mathrm{Gnd}_{\mathrm{SP}}$=1, F=1, named walk wrong; the remaining wrong paths are unfaithful walks.

Model	Integer length	Named walk	$\mathrm{Gnd}_{\mathrm{SP}}$	F	**Loss**	**Competition**
Claude	85/210 (40.5%)	63/210 (30.0%)	99/210	152/210	111/147	15/147
GPT	173/210 (82.4%)	168/210 (80.0%)	178/210	208/210	31/42	11/42
Gemini	154/210 (73.3%)	150/210 (71.4%)	151/210	206/210	59/60	1/60
Qwen	109/210 (51.9%)	76/210 (36.2%)	117/210	127/210	88/134	15/134

Observation. Table 27 shows three results.

Length overstates naming. For every model, the integer length is right more often than the named path. Claude scores 40.5% vs. 30.0%, GPT 82.4% vs. 80.0%, Gemini 73.3% vs. 71.4%, and Qwen 51.9% vs. 36.2%.

Loss dominates. Among wrong named paths, most have $\mathrm{Gnd}_{\mathrm{SP}}{=}0$ for Claude (111 of 147), Gemini (59 of 60), and Qwen (88 of 134). GPT has few errors. Most are loss (31 of 42), and some are competition (11 of 42).

List faithfulness. Gemini's named walk is list-faithful on 206 of 210 items. Qwen's is list-faithful on only 127.

Interpretation. A correct length is not the same as pointing to a real shortest path. On this call, a gold path is still present among the true-positive edges on 47.1% (Claude), 71.9% (Gemini), and 55.7% (Qwen) of items. Those three models are loss-dominated among wrong named paths. GPT makes few errors ($42/210$), and most of those are also loss. Degree results still argue against a globally unreadable picture. Integer path length does so only for GPT and Gemini. None of these probes sets a ceiling on naming a cycle.

Limitation. Shortest-path queries use distances 1–3, so the competition branch has few chances to arise. Loss dominance for paths is partly built in by design.

P TEXT-LEVEL INTERVENTIONS ON FROZEN LISTS

Takeaway. Shuffling the frozen list, or running a symbolic finder on it, does not recover named-cycle accuracy for most models. Only Qwen gains clearly from the finder.

Hypothesis. If the polluted list is easy to repair once the image is gone, a text-only fix on the frozen list should restore named-cycle accuracy. The fix might remove order effects, add careful reasoning, or use an algorithm.

Experiment. Table 28 freezes each model's own Ring list as plain text. We test three conditions:

- the original edge order;
- a shuffled edge order, which tests the order sensitivity that Cao et al. (2025) document for textual graph prompts (150 graphs);
- a gold-blind deterministic depth-first finder run on the stored list.

Table 28: Text-only interventions and a symbolic finder on lists extracted from the equal-angle drawings ($N{=}500$). Shuffle is on a 150-graph subset, shown with the frozen list on the same subset. Finder is a gold-blind DFS cycle finder run on the stored EXTRACT lists. R_E: the named cycle is a valid cycle of the frozen list it was given. R_G: decision on clean gold edges.

Experimental Condition	Claude	GPT	Gemini	Qwen
Ring Visual Baseline (V_{strict})	4.8%	42.4%	44.0%	8.8%
List-then-Decide (LIST+DECIDE, Visual)	6.0%	43.8%	43.2%	11.8%
Frozen Extracted List (Text-Only)	7.6%	42.8%	43.0%	6.6%
Frozen List, shuffle subset	9.3%	42.0%	43.3%	8.7%
Frozen List + Edge Shuffle	5.3%	45.3%	43.3%	14.1%
Deterministic Cycle Finder on EXTRACT Ring	8.4%	46.6%	46.8%	18.1%
Deterministic Cycle Finder on EXTRACT Base	70.4%	92.2%	99.2%	74.6%
Own-List Cycle Validity (R_E, Frozen)	93.0%	100.0%	100.0%	43.3%
Gold-List Comparison (R_G)	81.8%	100.0%	100.0%	65.2%

Observation. Table 28 shows that no text-only repair of the extracted list recovers named-cycle accuracy.

Frozen list. Frozen-list V_{strict} is 7.6%, 42.8%, 43.0%, and 6.6% for Claude, GPT, Gemini, and Qwen. Each is equivalent to the visual baseline within ± 10 points (two one-sided tests).

Shuffle. Shuffling does not change the frozen-list result on the same 150 graphs (McNemar $p \geq 0.15$). It is equivalent within ± 10 points for Claude, GPT, and Gemini.

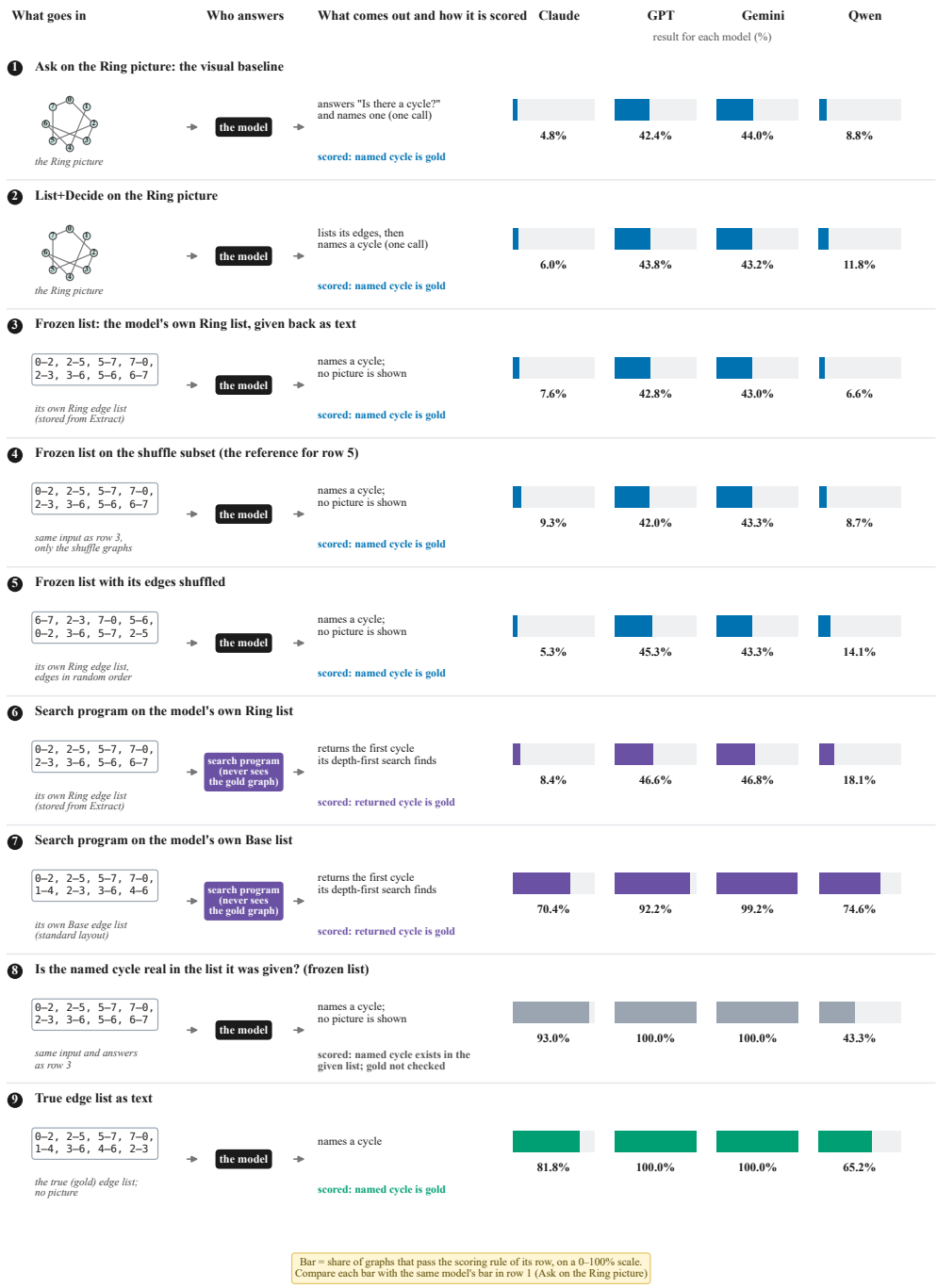


Figure 17: Walkthrough of every condition in Table 28, one row per table row and in the same order. Each row shows what the input is, who answers, what is produced, how it is scored, and each model's result. The edge lists are toy examples. The model's own Ring list (rows 3–6 and 8) misses one gold edge and adds two false ones, while its Base list (row 7) and the true list (row 9) are clean. Rows 1–2 give the model the Ring picture. Rows 3–5 give it only its own stored Ring list as text: unchanged, on the 150-graph shuffle subset, or with the edges in random order. Rows 6–7 replace the model with a depth-first search program that never sees the gold graph. Row 8 re-scores the answers of row 3 against the list the model was given, so it checks list validity, not correctness (R_E). Row 9 gives the model the true edge list as text (R_G). Row 8 uses a different scoring rule from the other rows.

Finder. On the same stored Ring lists, the finder returns a gold cycle on 8.4–46.8% of items. We compare it with each model's own named cycle on the same frozen list (exact McNemar, uncorrected). It does no better than Claude (8.4% vs. 7.6%, p=0.64). It is above GPT, but not significantly (46.6% vs. 42.8%, p=0.050). It is slightly better than Gemini (46.8% vs. 43.0%, p=0.014) and far better than Qwen (18.1% vs. 6.6%, $p=3\times10^{-11}$).

DFS prompt. On the 70-graph pilot set, the explicit prompt did not differ from the frozen list ($p \geq 0.18$).

Interpretation. Edge order does not appear to explain the pollution added during extraction. A correct computation on the wrong graph still gives a wrong answer. The finder gives a second reading through a cross-call comparison: its output on a model's own EXTRACT list against that model's separate ASK or LIST+DECIDE answer. On clean Base lists, Claude and Qwen fail to name cycles that their own lists contain. This is a cross-call hint, not proof of a reasoning deficit.

Limitation. The finder's output is valid in the listed graph by construction. The gold graph is used only afterwards, for scoring. A finder restricted to true-positive edges must return a gold cycle whenever Gnd=1. The shuffle comparison uses a subset. The DFS prompt was not run on the full collection, so an explicit reasoning prompt is untested on these 500 graphs.

Q LOCALIZING THE DEFICIT: GOLD-LIST SUBSTITUTION

Takeaway. Given the clean gold list as text, GPT and Gemini reach 100%. Claude (81.8%) and Qwen (65.2%) do not. For GPT and Gemini, the gap comes from the extracted list. For Claude and Qwen, a gap remains even on a clean list.

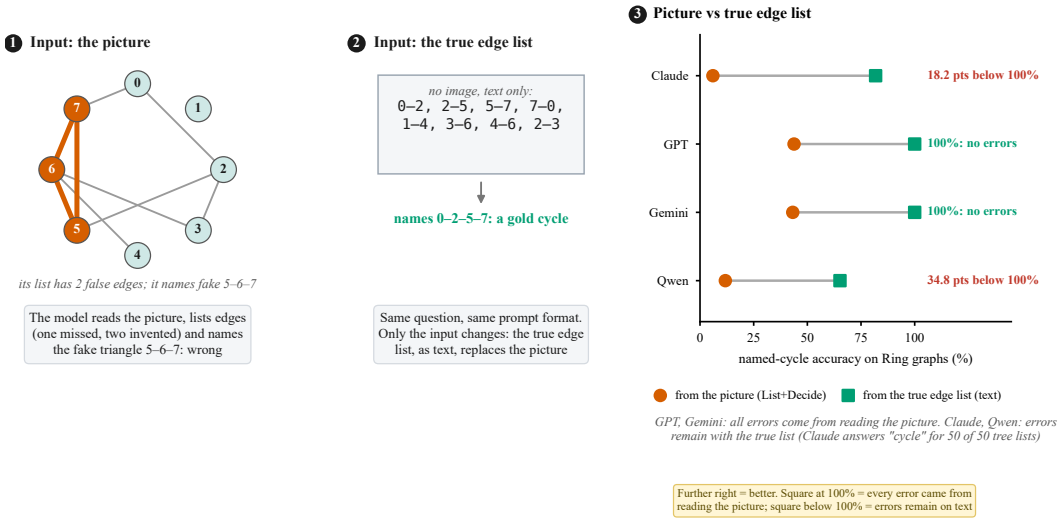


Figure 18: Walkthrough of this appendix on the toy ring. (1) From the picture, the model lists edges with one gold edge missed and two invented, and names the fake triangle. (2) The gold edge list is given as text with no image; the question and prompt shape are unchanged. (3) Named-cycle accuracy from the picture (LIST+DECIDE) and from the gold list (R_G). For GPT and Gemini the gap closes, so the deficit is in reading the picture. Claude and Qwen keep a residual on clean text, which extraction cannot explain.

Hypothesis. If the deficit sits in extraction and not in the later decision, swapping the polluted list for the clean gold list should restore named-cycle accuracy. The decision task and prompt structure stay the same.

Experiment. R_G is a text-only cycle decision given the clean gold edge list $E^\star$. It uses no image and no model-extracted intermediate. It covers the cyclic graphs (500).

Observation. GPT and Gemini reach 100.0% named-cycle accuracy given the clean list. Claude reaches 81.8% and Qwen 65.2%. On the same graphs, R_G beats the one-call LIST+DECIDE condition for every model. The discordant pairs b, c are $382, 3$ (Claude), $281, 0$ (GPT), $220, 0$ (Gemini), and $276, 9$ (Qwen). All differences are significant after Holm correction.

Interpretation. For GPT and Gemini, the gap between picture and clean list closes. It comes from the extracted list, not from an inability to do the typed-list task. Claude (18.2-point residual) and Qwen (34.8-point residual) keep a gap on the clean list. This is a direct single-call measurement on clean text, so extraction cannot explain it. Claude also answers "cycle" on $50/50$ clean tree lists (Appendix S), so part of the gap may be a yes-bias on text. This appendix is not the Base→Ring decomposition (Appendix I).

Limitation. This ablation changes the input directly instead of re-scoring stored outputs, which makes it the main localizing evidence in the paper. It still tests externalized text, not a model's internal representation. A model could hold the right structure and still fail our task, which asks for an explicit edge list. It could also do the reverse. We treat R_G as strong evidence for where the deficit sits, not as a claim about internal representations.

R SINGLE-EDGE INSERTION ON TYPED LISTS

Takeaway. Adding one false edge to a clean gold list redirects the named cycle when the edge makes the cycle shorter. It rarely does so when the edge makes it longer. This holds for all four models. Gemini shows it most cleanly and Qwen least.

Manipulation follow-ups. Section 6 and Appendices M–Q find the loss/competition split, and the length and shortest-cycle pattern within it, from stored model outputs and clean-input checks. The next three appendices change the input in a controlled way. Appendices R and S plant one edge in a *text* list and re-query. Appendix T asks present/absent questions about the existing Ring images, with no inserted edge and no new rendering. Together they ask three things. What does the length rule do on text lists? Does it hold on trees? Does crossing-sensitive acceptance of gold edges appear on unmodified drawings? All four models use the same snapshots as the main tables. Appendix E gives construction details and prompts.

Hypothesis. If length-based competition (Section 6) is a real selection rule, one added false edge should redirect the named cycle through it when it shortens the cycle. It should not when it lengthens the cycle. On the presented gold$+1$ list, both the gold cycle and the new cycle are valid. Deciding whether a cycle exists does not depend on length, so a model that only answered that question would ignore the edge's length. A model that prefers short cycles would not.

Experiment. For each cyclic gold graph, we add one edge $(u, v) \notin E^\star$ to the clean gold list. We then re-run the R_G text-only prompt on $E^\star \cup \{(u, v)\}$. We choose (u, v) so the new cycle through it is *shorter than*, *equal to*, or *longer than* the girth of $G^\star$. These are the three arms (Appendix E). Cross-model comparisons use the 60 graphs shared by all four models. We record whether the named cycle uses the inserted edge.

Table 29: Single-edge insertion into the clean gold list (R_G prompt, text-only), on the 60 graphs evaluated for all four models. Shorter/Equal/Longer: the inserted edge's resulting cycle length relative to gold girth. Cells are the share of graphs where the named cycle uses the inserted edge.

Model	Shorter Arm	Equal Arm	Longer Arm	Shorter − Longer
Claude	$34/36$ (94.4%)	$23/57$ (40.4%)	$6/55$ (10.9%)	83.5pp
GPT	$34/36$ (94.4%)	$21/57$ (36.8%)	$7/55$ (12.7%)	81.7pp
Gemini	$36/36$ (100.0%)	$22/57$ (38.6%)	$2/55$ (3.6%)	96.4pp
Qwen	$21/36$ (58.3%)	$20/57$ (35.1%)	$8/55$ (14.5%)	43.8pp

Observation. Table 29 shows a sharp length effect for all four models. The inserted edge is used in 58.3–100.0% of shorter-arm graphs but only 3.6–14.5% of longer-arm graphs. That is a gap of 43.8–96.4 points. The equal-length arm falls in between (35.1–40.4%), which fits a tie-break rather than a strict-inequality rule. Paired shorter-vs.-longer comparisons are significant for all four models after Holm correction.

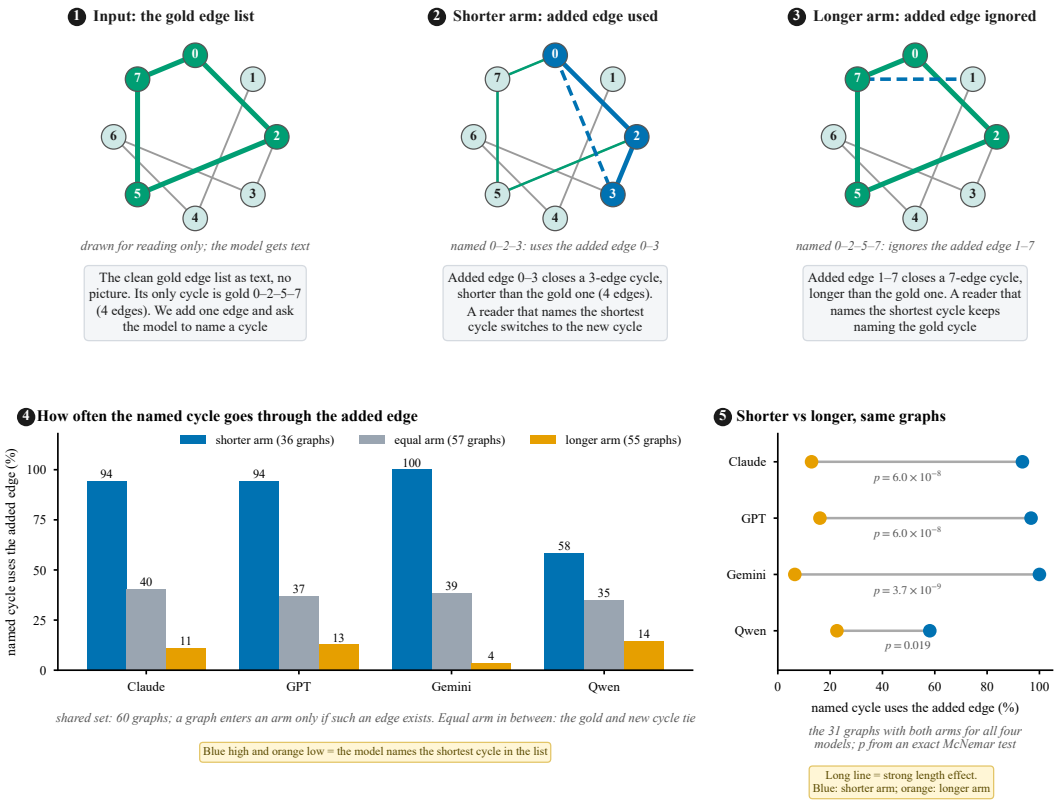


Figure 19: Walkthrough of this appendix. Panels (1)–(3) use the toy graph of Appendix E, whose stimuli are text lists, not pictures; the named cycles there illustrate what a reader that names the shortest cycle would answer. (1) The input is the clean gold edge list, whose only cycle is the gold 4-edge cycle. (2) Shorter arm: the added edge closes a 3-edge cycle, so a shortest-cycle reader names the new cycle through the added edge. (3) Longer arm: the added edge closes a 7-edge cycle, so a shortest-cycle reader keeps naming the gold cycle. (4) Share of graphs where the model's named cycle goes through the added edge, per arm (Table 29). (5) Shorter vs. longer arm on the 31 graphs that have both arms for all four models, with exact McNemar p (Table 30). On this input both cycles are valid answers, so the test shows which cycle is chosen, not an error.

Table 30: Robustness of the length effect in Table 29. Full set: all graphs evaluated for that model. Common subset: the 31 graphs where all four models have both a shorter and a longer arm. p: exact McNemar test on the common subset.

Model	Full: Shorter	Full: Longer	Common: Shorter	Common: Longer	p
Claude	87.3%	11.6%	93.5%	12.9%	6.0×10^{-8}
GPT	88.6%	8.7%	96.8%	16.1%	6.0×10^{-8}
Gemini	100.0%	1.8%	100.0%	6.5%	3.7×10^{-9}
Qwen	48.1%	10.8%	58.1%	22.6%	0.019

Table 30 shows the same pattern on each model's full set and on the common 31-graph subset. For Claude, restricting to graphs where its clean-list decision was already correct gives 81.3% vs. 5.2%.

Interpretation. On this text input, naming the shorter valid cycle is correct behaviour, because the prompt presents gold+1 as the graph. Selection prefers the shortest cycle of whatever list is in front of it. One controlled edge redirects the named cycle, and the effect flips sign with length. Only the inserted edge's length relative to girth changes between arms. So this avoids confounds that come with any single model's own Ring extraction, unlike the stored-output correlations of Section 6. It

describes how selection works on text lists. It does not show a failure, and it does not show a visual preference for fake over real edges.

Limitation. The intervention is on text, not on the image. It shows that length steers selection once a false edge is in a list. It does not show that a model's visual extraction inserts false edges by length. That link remains observational (Appendix G). Girth is the only reference length. We did not sweep intermediate lengths finely enough to estimate a threshold or a preference curve. A deletion-based necessity test is not reported, because the eligible sample was too small for corrected inference.

S CHORD INSERTION ON TREES

Takeaway. With one chord added to a tree, GPT and Gemini name the inserted cycle exactly at both lengths. Claude and Qwen do so far less often on the long arm. Claude and Qwen also say "cycle" on almost every unmodified tree.

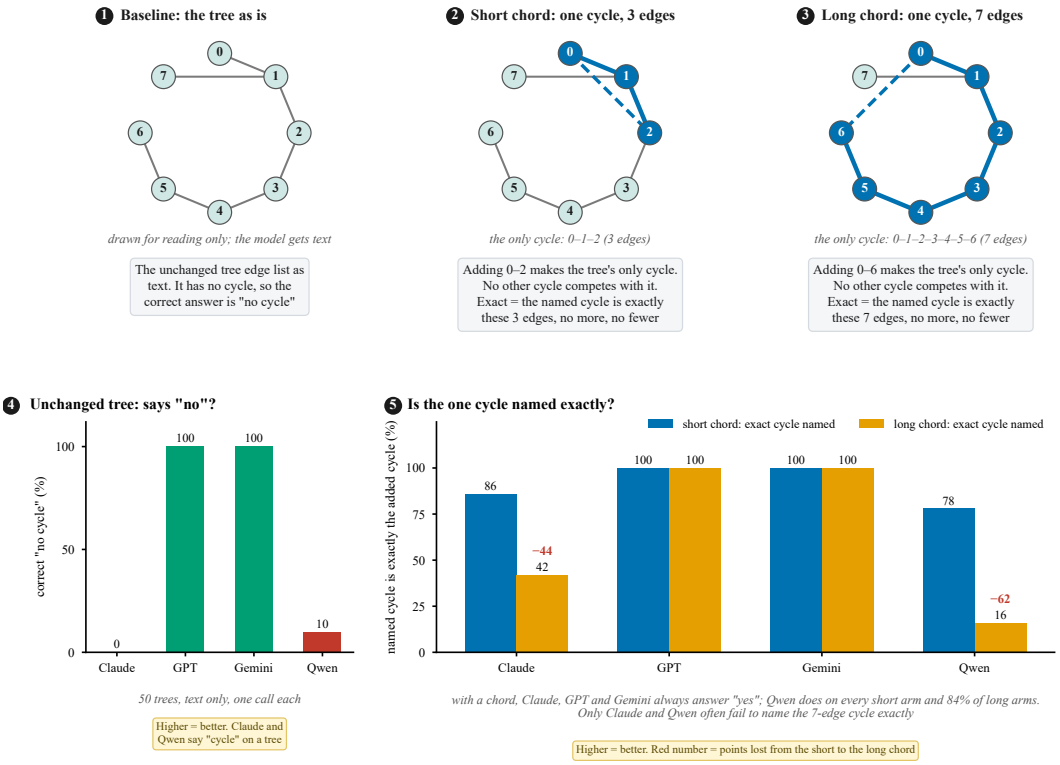


Figure 20: Walkthrough of this appendix. Panels (1)–(3) use the toy tree of Appendix E; the model receives the edge list as text, not a picture. (1) Baseline: the unchanged tree has no cycle, so the correct answer is "no cycle". (2) Short chord: one added edge creates the tree's only cycle, with 3 edges. (3) Long chord: one added edge creates the only cycle, with 7 edges. No other cycle competes in either arm. (4) Share of unchanged trees where the model correctly answers "no cycle". (5) Share of trees where the named cycle is exactly the added cycle, for the short and long chord; red numbers give the drop from short to long (Table 31).

Hypothesis. If the length effect is a general property of how these models handle cycles in text lists, it should also appear when a chord turns a tree into a graph with one cycle. A single cycle has no competitor, so any length effect here would not come from competition.

Experiment. For each of the 50 gold trees, we add one chord (u, v) to the clean gold edge list. This creates the tree's only cycle. The *short* arm gives a cycle of 3–4 edges, and the *long* arm gives one of 7 edges. We run the R_G prompt on the tree edges plus the chord. An unmodified-tree run serves as the baseline.

Table 31: Chord insertion on the 50 gold trees (R_G prompt, text-only). Baseline: unmodified tree gold list, correct answer no. Short/Long: one chord added, creating the tree's only cycle at 3–4 or 7 edges. Exact: the named cycle is exactly the inserted cycle.

Model	Baseline: No	Short: Yes	Short: Exact	Long: Yes	Long: Exact
Claude	0/50 (0.0%)	50/50 (100.0%)	43/50 (86.0%)	50/50 (100.0%)	21/50 (42.0%)
GPT	50/50 (100.0%)	50/50 (100.0%)	50/50 (100.0%)	50/50 (100.0%)	50/50 (100.0%)
Gemini	50/50 (100.0%)	50/50 (100.0%)	50/50 (100.0%)	50/50 (100.0%)	50/50 (100.0%)
Qwen	5/50 (10.0%)	50/50 (100.0%)	39/50 (78.0%)	42/50 (84.0%)	8/50 (16.0%)

Observation. Table 31 shows three results.

Baseline. GPT and Gemini correctly say "no cycle" on every unmodified tree. Claude (0/50) and Qwen (5/50) almost never do, in line with their tree yes-bias (Table 3). Claude's 0/50 comes from a single call on a clean text list with no image, so its yes-bias does not come from extraction.

With a chord. Claude, GPT, and Gemini say "yes" on every tree in both arms. Qwen does so on the short arm and on 84.0% of the long arm.

Exact cycle. The sharper test is whether the named cycle is exactly the inserted one. GPT and Gemini stay at 100.0% in both arms. Claude falls from 86.0% (short) to 42.0% (long), and Qwen from 78.0% to 16.0%. Both drops are significant after Holm correction.

Interpretation. For GPT and Gemini, the tree control shows that the decision procedure works at longer lengths. That procedure is to find the one cycle a text edge list contains. Their length-dependent failures therefore need a competing cycle, which one chord on a tree does not create. Claude and Qwen drop even with a single candidate cycle. This points to a separate weakness: rebuilding a longer cycle exactly, with no competition involved. It adds to their extraction-loss problems (Section 6) and to their gap on the clean gold list (Appendix Q).

Limitation. The long arm uses the smallest available distance of at least 6, so every long cycle has 7 edges. We did not sweep length continuously.

T PER-EDGE PICTURE QUESTIONS (UNMODIFIED IMAGE; NO EDGE ADDED OR DELETED)

Takeaway. We asked each model single yes/no questions about edges in the unchanged Ring picture. Two results stand out. First, crossings matter for some models only. Claude and Qwen say "yes" to a real edge less often when it has many crossings. GPT and Gemini show no such drop. Both say "yes" slightly more often to heavily crossed edges than to lightly crossed ones. Second, the models differ in how often they say "yes" to an edge that does not exist (a false alarm). Qwen does this most. GPT and Gemini do it rarely, and mostly for pairs of ring neighbours (Figure 3).

Context. Here we ask about edges in the picture exactly as drawn. No edge is added, removed, or moved, so every crossing stays as it was. This differs from T1 and T2, which add an edge to a typed list. Appendix M already shows that the pictures are readable as a whole: the models compare node degrees well. It also shows that GPT and Gemini read paths well, while Claude and Qwen often do not. Those tests do not ask about single edges. This one does.

Hypothesis. Table 11 shows that the edges models drop from their lists have more crossings. If crossings make a single edge harder to see, a real edge with many crossings should get a "yes" less often than one with few. The effect should belong to the edge itself, not to the graph as a whole. So we compare edges within the same picture.

Experiment. For each Ring picture we choose eight pairs of nodes (Appendix E). For each pair we ask: is this edge drawn? There are two pairs of each kind:

- *neighbour non-edge:* two nodes next to each other on the circle, with no edge between them;
- *far non-edge:* two nodes at least 8 positions apart on the circle, with no edge between them;



Figure 21: Walkthrough of this appendix. (1) On the unchanged Ring picture, the model answers "Is the edge u–v drawn?" for eight pairs of nodes, two of each kind: a real edge with many crossings (A) or few crossings (B), and a pair with no edge whose nodes are ring neighbours (C) or far apart on the circle (D). The correct answer is "yes" for A and B and "no" for C and D. (2) Real edges: how often the model correctly says "yes". Qwen, and to a lesser degree Claude, say "yes" less often when the edge has many crossings. GPT and Gemini say "yes" slightly more often when it has many crossings (Table 32). (3) Pairs with no edge: how often the model wrongly says "yes". Qwen does so for both kinds; GPT and Gemini do so rarely, and mostly for ring neighbours.

- *heavily crossed real edge:* taken from the quarter of the graph's real edges with the most crossings;
- *lightly crossed real edge:* taken from the quarter with the fewest crossings.

All four models answered on the same 75 graphs. That gives 150 questions of each kind per model. For the statistical test we use each model's full set of graphs. For each graph we take the "yes" rate on lightly crossed edges minus the "yes" rate on heavily crossed edges. We test these differences with a Wilcoxon signed-rank test. Heavily crossed edges that in fact have zero crossings are left out of this test.

Observation. Table 32 shows two patterns.

Real edges. Qwen says "yes" to heavily crossed real edges much less often than to lightly crossed ones (64.0% vs. 84.7%). Claude shows the same direction, more weakly (78.0% vs. 84.0%). Both differences stay significant after Holm correction. GPT and Gemini show no drop. Both say "yes" more often to heavily crossed edges than to lightly crossed ones: GPT by 4 points (98.0% vs. 94.0%) and Gemini by 8 points (88.7% vs. 80.7%).

Non-edges. Qwen gives the most false alarms. It says "yes" to 46.7% of neighbour non-edges and 40.0% of far ones. Claude's false alarms are about the same for both kinds (27.3% and 25.3%). GPT

Table 32: Edge-presence judgments on the Ring images (150 probes per column: 2 per graph $\times$ 75 shared graphs). Adjacent/Far Non-Edge: false-alarm rate (the model says "yes" to an edge that does not exist) for two ring neighbours vs. two nodes at least 8 positions apart on the circle. High/Low-Crossing Gold: how often the model says "yes" to a real edge from the most- vs. least-crossed quarter of that graph's real edges.

Model	Adjacent Non-Edge (FA)	Far Non-Edge (FA)	High-Crossing Gold	Low-Crossing Gold
Claude	41/150 (27.3%)	38/150 (25.3%)	117/150 (78.0%)	126/150 (84.0%)
GPT	18/150 (12.0%)	6/150 (4.0%)	147/150 (98.0%)	141/150 (94.0%)
Gemini	21/150 (14.0%)	2/150 (1.3%)	133/150 (88.7%)	121/150 (80.7%)
Qwen	70/150 (46.7%)	60/150 (40.0%)	96/150 (64.0%)	127/150 (84.7%)

(12.0% vs. 4.0%) and Gemini (14.0% vs. 1.3%) give the fewest false alarms, and most of theirs fall on neighbour pairs. Gemini has the largest gap between neighbour and far pairs.

Interpretation. No single property of the drawing explains all four models.

- Qwen misses heavily crossed real edges and also accepts many non-edges. This fits its lists, which often lose the gold cycle and contain many false edges.
- GPT and Gemini wrongly accept mostly neighbour pairs, and crossings do not lower their acceptance of real edges. This pattern is largely consistent with the clearance effect of Figure 3, panel 4: a real chord that passes close to a node can look like two neighbour edges. So it is not only a general preference for neighbour edges (Appendix G). A general rule of avoiding crossed edges does not explain it either.
- Claude's false alarms are flat, and its crossing effect is small. This points to errors spread over the picture, not tied to one geometric feature.

Limitation. "Heavily" and "lightly" crossed mean the top and bottom quarter within each graph, not a fixed number of crossings. In graphs with few crossings, some "heavily crossed" edges have no crossings at all (29% of the shared heavily crossed questions). This makes the contrast smaller and harder to detect.

U Observational Crossing Bins

Takeaway. Total crossings do not order the drawings well. Crossings on the least-crossed shortest gold cycle (K) track accuracy better. Rings with very few crossings still collapse.

Hypothesis. If total crossing count caused the collapse, rings with few crossings should be easy. If crossings on the gold cycle plus neighbour shortcuts (Figure 3) cause it, rings with few total crossings can still fail. A drawing with $K{=}0$ can still offer many neighbour shortcuts.

Experiment. Sections 5–6 show the collapse by comparing Ring (mean 53.4 crossings) with Base (mean 1.9). Here we bin the existing 500 Ring drawings by total crossings and, separately, by K (Tables 33–34). The analysis is observational. Graph size, edge count, source, and vertex order all vary with the bins. Density was not manipulated on a fixed graph. Figure 22, panel 3, plots V_{strict} against K.

Observation. We report five results.

1. Total crossings. Accuracy does not fall steadily. The lowest bands (all GITQA cycle-order rings) are not the easiest. At 11–20 crossings, GPT (78.0%) and Gemini (90.2%) do better than at 0–5 crossings (44.3–50.0% and 60.7–63.6%). That middle band is almost all generated graphs.

2. K. Accuracy declines overall and reaches 0% for all four models at $K \geq 41$ (30 graphs). It does not fall steadily from the first bin, though (Table 34). GPT rises from 52.5% at $K{=}0$ to 61.6% at $K{=}6$–10 before dropping. Gemini rises from 62.7% to 75.9% at $K{=}1$–2, and Claude from 18.6% to 27.6%. On the $K \geq 41$ graphs, Base accuracy stays at 63% (GPT) and 100% (Gemini).

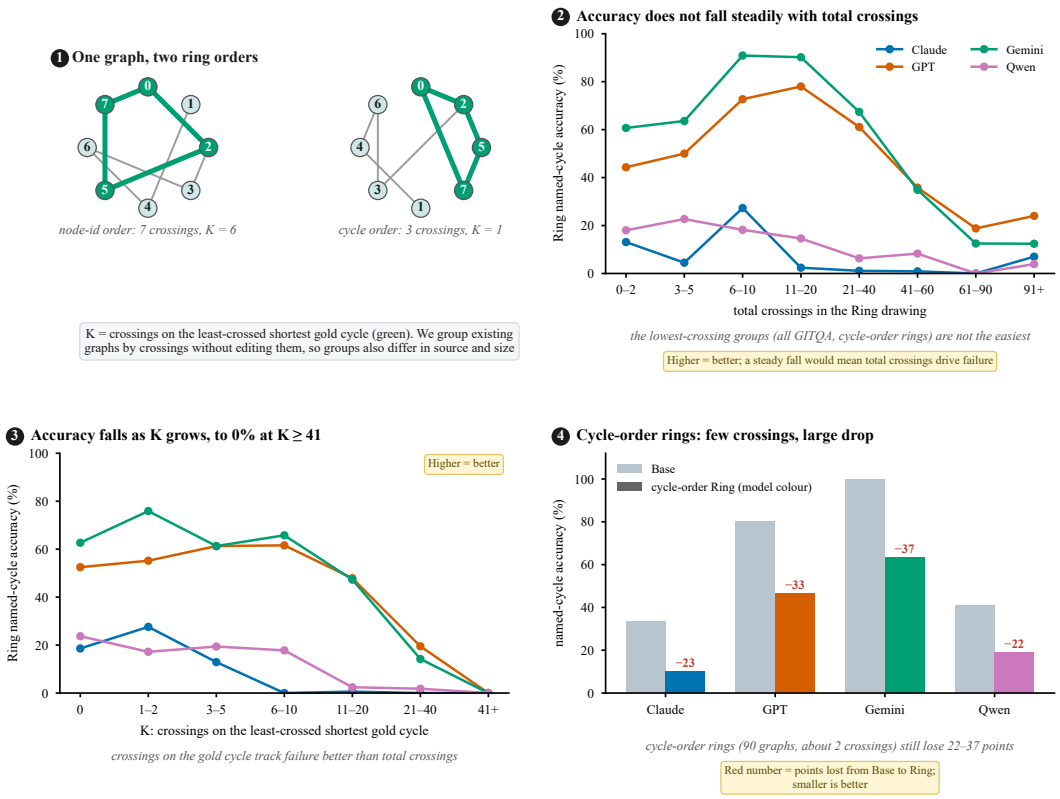


Figure 22: Walkthrough of this appendix. (1) The same graph in node-id order and in cycle order; K counts crossings on the least-crossed shortest gold cycle (green). Bins are observational, not a manipulated sweep. (2) Ring named-cycle accuracy by total crossings does not fall steadily (Table 33). (3) By K, accuracy declines and reaches 0 at 41+ (Table 34). (4) Cycle-order rings, with few crossings, still lose accuracy relative to Base (Table 36).

Table 33: Ring V_{strict} by total Ring crossings (500 cyclic graphs). Graphs: count (GITQA / generated). Brackets: how many percentage points lower Ring V_{strict} is than Base V_{strict} on the same graphs (↓ = a drop from Base to Ring), from exact per-item counts.

Crossings	Graphs	Claude	GPT	Gemini	Qwen
0–2	61 (61/0)	13.1% [↓24.6]	44.3% [↓41.0]	60.7% [↓39.3]	18.0% [↓29.5]
3–5	22 (22/0)	4.5% [↓18.2]	50.0% [↓13.6]	63.6% [↓36.4]	22.7% [↓4.5]
6–10	11 (6/5)	27.3% [↓18.2]	72.7% [↓18.2]	90.9% [↓9.1]	18.2% [↓18.2]
11–20	41 (1/40)	2.4% [↓61.0]	78.0% [↓22.0]	90.2% [↓9.8]	14.6% [↓36.6]
21–40	95 (0/95)	1.1% [↓46.3]	61.1% [↓35.8]	67.4% [↓31.6]	6.3% [↓42.1]
41–60	109 (3/106)	0.9% [↓31.2]	35.8% [↓56.9]	34.9% [↓61.5]	8.3% [↓29.4]
61–90	32 (28/4)	0.0% [↓43.8]	18.8% [↓62.5]	12.5% [↓87.5]	0.0% [↓50.0]
91+	129 (129/0)	7.0% [↓25.6]	24.0% [↓61.2]	12.4% [↓87.6]	3.9% [↓31.8]

3. Within each source. Ring V_{strict} correlates more strongly with K than with total crossings in seven of eight model-by-source cells. For GPT on GITQA, Spearman $\rho = -0.38$ vs. -0.18.

4. Loss share. Among wrong LIST+DECIDE answers, the loss share rises with crossings from the lowest to the highest band (Table 35). For Claude it goes 56% → 81%, for Gemini 22% → 100%, and for Qwen 47% → 79%. GPT stays competition-dominated.

Table 34: Ring and Base V_{strict} by K, the crossings on the least-crossed shortest gold cycle (500 cyclic graphs). Gap (pp): how many percentage points lower Ring is than Base (↓ = a drop from Base to Ring), from exact per-item counts. Graphs: count (GITQA / generated).

K	Graphs	Claude			GPT			Gemini			Qwen		
		Ring	Base	Gap (pp)	Ring	Base	Gap (pp)	Ring	Base	Gap (pp)	Ring	Base	Gap (pp)
0	59 (58/1)	18.6%	37%	↓ **18.4**	52.5%	85%	↓ **32.5**	62.7%	100%	↓ **37.3**	23.7%	44%	↓ **20.3**
1–2	29 (26/3)	27.6%	48%	↓ **20.4**	55.2%	79%	↓ **23.8**	75.9%	100%	↓ **24.1**	17.2%	55%	↓ **37.8**
3–5	31 (21/10)	12.9%	29%	↓ **16.1**	61.3%	84%	↓ **22.7**	61.3%	100%	↓ **38.7**	19.4%	32%	↓ **12.6**
6–10	73 (19/54)	0.0%	59%	↓ **59.0**	61.6%	96%	↓ **34.4**	65.8%	99%	↓ **33.2**	17.8%	60%	↓ **42.2**
11–20	165 (29/136)	0.6%	45%	↓ **44.4**	47.9%	96%	↓ **48.1**	47.3%	99%	↓ **51.7**	2.4%	40%	↓ **37.6**
21–40	113 (67/46)	0.0%	27%	↓ **27.0**	19.5%	88%	↓ **68.5**	14.2%	97%	↓ **82.8**	1.8%	37%	↓ **35.2**
41+	30 (30/0)	0.0%	10%	↓ **10.0**	0.0%	63%	↓ **63.0**	0.0%	100%	↓ **100.0**	0.0%	17%	↓ **17.0**

Table 35: One-call list-then-decide (LIST+DECIDE) on Ring by total-crossing band: V_{strict} / share of wrong outputs in the loss branch (Gnd_C=0).

Crossings	Claude	GPT	Gemini	Qwen
0–2	15% / 56%	51% / 17%	56% / 22%	23% / 47%
3–5	5% / 43%	73% / 0%	82% / 50%	23% / 29%
6–10	27% / 38%	64% / 0%	82% / 0%	45% / 17%
11–20	7% / 45%	78% / 11%	85% / 33%	15% / 54%
21–40	7% / 68%	60% / 5%	66% / 56%	16% / 55%
41–60	2% / 78%	35% / 7%	39% / 72%	5% / 71%
61–90	0% / 75%	22% / 32%	16% / 100%	0% / 84%
91+	4% / 81%	24% / 39%	8% / 100%	7% / 79%

Table 36: Same graphs, still a named-cycle collapse. Cycle-order Ring drawings (n=90) have mean 1.9 straight-chord crossings. V_{strict} and Base−Ring gap from exact per-item counts.

Model	Base V_{strict}	Ring V_{strict}	Gap (pp)
Claude	33.3% (30/90)	10.0% (9/90)	23.3
GPT	80.0% (72/90)	46.7% (42/90)	33.3
Gemini	100.0% (90/90)	63.3% (57/90)	36.7
Qwen	41.1% (37/90)	18.9% (17/90)	22.2

5. Cycle-order rings. Cycle-order rings have 1.9 mean crossings, yet they still collapse (Table 36; Figure 23). The Base−Ring named-cycle gaps are 23.3, 33.3, 36.7, and 22.2 points (Claude, GPT, Gemini, Qwen).

Interpretation. Total crossings alone do not order the drawings. Low-crossing rings can still invite ring-neighbour shortcuts, while crossings on the gold cycle track failure better. This is a correlation, not a manipulated sweep. Two results fit this reading: the uneven accuracy by total crossings, and the collapse of cycle-order rings. They would be a problem only if those drawings also had few neighbour shortcuts. The rising loss share with crossings for Claude, Gemini, and Qwen matches the per-edge miss pattern of Appendix G.

Limitation. Bands are not balanced by source or size. The lowest and highest total-crossing bands are all GITQA graphs. The middle bands are almost all generated graphs. Band differences are therefore not a clean dose–response. The collection spans a range of crossing density without manipulating it on a fixed graph.

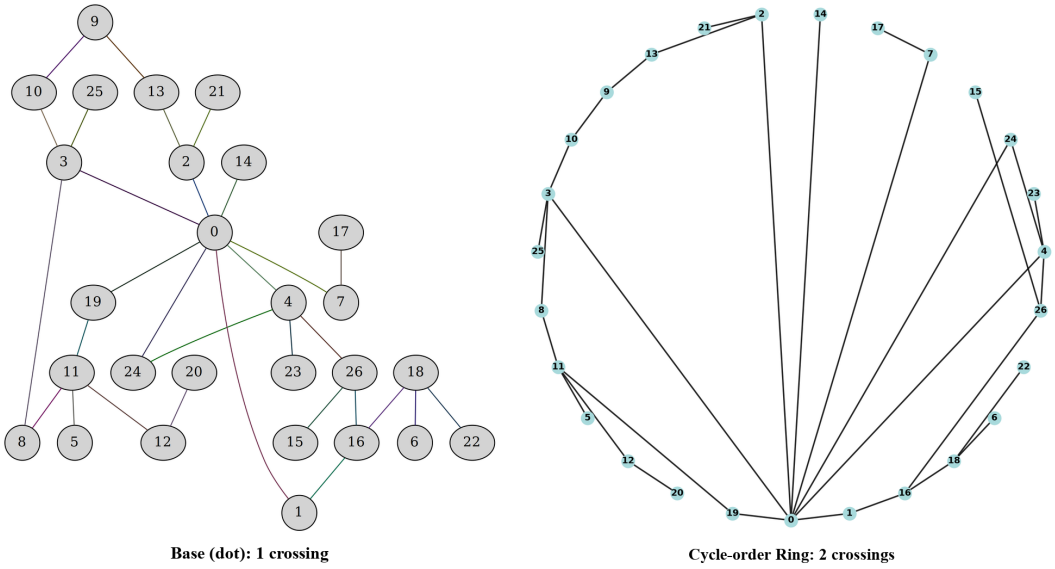


Figure 23: Base versus a cycle-order Ring of the *same* graph. GITQA `graph1105` (`gitqa_35`): the Base drawing has 1 crossing and the Ring has 2. Collection means are 1.9 visible-stroke crossings on all-cyclic Base and 1.9 straight-chord crossings on the 90 cycle-order rings. Named-cycle rates on this set are in Table 36.

V Full Cross-Condition Results

Takeaway. Table 37 collects the results of all experiments for the four models, layouts, and conditions in one place.

Table 37: Comprehensive experimental results across all conditions (500 cyclic graphs, 50 trees). Shuffle uses a 150-graph subset, and the Freeze / Shuffle rows use that subset on both sides; V_{deg} and V_{SP} use 140 and 210 items.

Metric	Condition	Claude	GPT	Gemini	Qwen
V_{pol}	Base / Ring	100.0% → 97.4%	99.8% → 100.0%	99.8% → 84.0%	96.4% → 99.8%
V_{strict}	Base / Ring	39.0% → 4.8%	89.2% → 42.4%	99.0% → 44.0%	41.8% → 8.8%
V_{strict}	List+Decide Ring	6.0%	43.8%	43.2%	11.8%
Prec	Extract Base / Ring	0.951 → 0.504	0.992 → 0.872	0.999 → 0.849	0.963 → 0.681
Rec	Extract Base / Ring	0.964 → 0.591	0.993 → 0.936	0.999 → 0.812	0.947 → 0.692
Sym_X	Extract Base / Ring	99.2% → 100.0%	99.4% → 99.6%	99.8% → 84.0%	95.6% → 95.0%
Gnd_X	Extract Base / Ring	92.8% → 33.4%	98.4% → 86.8%	99.8% → 61.2%	89.0% → 42.6%
Gnd_C	List+Decide Ring	35.0%	87.8%	55.4%	42.4%
F	List+Decide Ring	88.6%	98.6%	99.4%	73.8%
F	Freeze / Shuffle	92.0% → 90.0%	100.0% → 99.3%	100.0% → 100.0%	75.8% → 57.0%
V_{deg}	Base / Ring	99.3% → 93.6%	100.0% → 96.4%	100.0% → 95.7%	92.9% → 95.0%
Sym_{deg}	Base / Ring	94.3% → 87.9%	97.9% → 94.3%	100.0% → 84.3%	93.6% → 79.3%
V_{SP}	Base / Ring	70.0% → 45.2%	97.6% → 86.7%	100.0% → 80.0%	58.6% → 49.0%
Sym_{SP}	Base / Ring	92.4% → 55.2%	96.7% → 84.8%	100.0% → 66.2%	91.4% → 56.7%
R_E	Freeze / Shuffle	90.7% → 88.7%	100.0% → 99.3%	100.0% → 100.0%	49.7% → 53.0%
R_G	Gold List	81.8%	100.0%	100.0%	65.2%
Frozen V_{strict}	Original / Shuffle	9.3% → 5.3%	42.0% → 45.3%	43.3% → 43.3%	8.7% → 14.1%
Finder V_{strict}	Extract Base / Ring	70.4% → 8.4%	92.2% → 46.6%	99.2% → 46.8%	74.6% → 18.1%
Tree correct "no"	Base / Ring	0.0% → 4.0%	78.0% → 6.0%	100.0% → 64.0%	12.0% → 2.0%
Tree Sym_X	Base / Ring	78.0% → 0.0%	82.0% → 4.0%	100.0% → 76.0%	72.0% → 28.0%

PART V — SCOPE AND RELATED WORK

W SCOPE, LIMITATIONS, AND WHAT WOULD CHANGE THE CONCLUSIONS

Section 8 summarizes the boundaries of this study; this appendix gives the full itemized list. The definitions of Gnd, F, and V_{strict} are in Section 3.

- **Benchmark scope.** Labelled GITQA and generated node-link drawings, not unlabelled graphs, flowcharts, or domain schematics. One equal-angle layout family; `circo` was rendered but not scored.
- **Geometry is correlational.** The difficulty score D_{fin} orders Base against Ring and separates the two model pairs, but inside one layout the ranking is weak (Appendix K). The only direct per-edge crossing test (Appendix T, unmodified images) is significant after correction for Claude and Qwen. GPT and Gemini show no drop: both say "yes" more often to heavily crossed edges. No single geometric property explains all four models.
- **Text insertion is not a picture manipulation.** Single-edge insertion shows that, on a typed list presenting gold$+1$ as the graph, the named cycle follows the shortest cycle of the list. That is what the prompt licenses, so it shows what the selection rule is, not that any model fails. It does not show that a model's own visual extraction inserts false edges according to length. A deletion-based necessity test was not reported because the eligible pool was too small. Gnd is computed at evaluation time against $G^{\star}$. Insertion and probe comparisons use the 60 and 75 graphs all four models share (all GITQA, mean $n \approx 27$).
- **What is a direct measurement and what is cross-call.** Direct: on a clean typed list with no image, Claude and Qwen still name a gold cycle on only 81.8% and 65.2% of graphs, and Claude answers "cycle" on $50/50$ clean tree lists (Appendices Q, S). These are single-call measurements on a text input, so they are independent of extraction. What they do *not* show is the cause of the residual (reasoning, thinking budget, prompt, or scoring). Cross-call: comparing a gold-blind finder or an extracted list from one call with a model's answer from another (finder on Base lists versus ASK; Claude's acyclic Base tree lists versus its ASK answer; $\mathrm{Gnd}_{\mathrm{SP}}$ versus V_{SP}). We use those only as supporting hints.
- **Benchmark exposure.** Official GITQA graphs may have entered training data. Base accuracy is not significantly higher on GITQA than on generated graphs for any model (it is lower for three, significantly so for GPT), but GITQA graphs are much larger (mean n=27.4 vs. 16.4), so size confounds that comparison and it cannot settle exposure. The 250 generated graphs are included in the supplementary material; Ring renders and list perturbations are new.
- **Human reference.** Raters did not share drawings, so agreement is not computable; the task is a 2AFC discrimination against a foil (not unaided generation), it covers all 500 graphs, and it rests on 19 errors, so comparisons with model difficulty are underpowered (Appendix N).
- **Model access.** API endpoints with fixed decoding configurations that providers may change; the four models are widely used frontier models, one snapshot each.

What would change our conclusions. A layout with few crossings and few ring-neighbour shortcuts (Figure 3) that still produced the same collapse would weaken the crossing and neighbour-shortcut account (a clearance effect would remain untested); cycle-order rings already show that total crossings alone do not explain the collapse (Table 36).

X EXTENDED RELATED WORK

This appendix expands Section 2. We group prior work into three lines by what it measures, and close with what remains unmeasured. Table 38 summarizes the closest work.

X.1 Visual graph reasoning with VLMs

Benchmarks on drawn graphs. VisionGraph poses graph-theory problems as images (Li et al., 2024). It scores node and edge recognition as separate sub-tasks, finds edge recognition poor for every model, and argues that this limits downstream reasoning. For its commercial models (GPT-4V and Gemini Pro, and in a further table Qwen-Plus and Qwen-Max), it also reports a stricter secondary accuracy on connectivity and cycle questions: a "yes" counts only with a verifiable path or cycle. This is the closest prior instrument to ours. Extraction and answers are, however, scored as separate tasks and linked only in aggregate.

GITA releases GVLQA, originally named GITQA (Wei et al., 2024); our GITQA graphs come from its GITQA-Aug-Legacy release (Wei, 2024). The augmented splits vary layout, node shape, node outline, and edge thickness. GITA studies these variants as training augmentations (layout helps most), not as a probe of how well off-the-shelf models extract structure.

VGCure covers 22 understanding and reasoning tasks and finds LVLMs weakest on relational and structurally complex queries (Zhu et al., 2025). It proposes MCDGraph, which trains on masked graph infilling, contrastive graph discrimination, and graph description. Its extraction-style tasks are local (neighbour queries, edge counts), not full edge lists.

Broader suites. Other benchmarks cover multi-hop queries on real-world diagrams (Ai et al., 2024), visualized knowledge-graph QA (Du et al., 2026), and multi-graph parsing and reasoning (Ai et al., 2025). The GraphVista agent pairs retrieved text with rendered subgraphs and is evaluated on much larger graphs and more task types (Han et al., 2026). GraphVis renders retrieved knowledge-graph subgraphs as images and fine-tunes an LVLM on them with a curriculum (Deng et al., 2024).

Layout transfer and path following. The Visual Graph Arena tests layout-invariant conceptualization on six tasks: graph isomorphism (easy and hard), Hamiltonian path and cycle, shortest-path length, and largest chordless cycle (Babaiee et al., 2025). It trains on one layout and tests on another (random, Kamada–Kawai, planar), and finds models far below near-perfect human accuracy. Its shortest-path items are a four-way length classification, not a named s–t path. TraversalBench scores exact ordered traversal of a single marked polyline and finds that accuracy drops at the first self-intersection (Petrova et al., 2026). This is complementary evidence on visual continuation, but the stimuli are not graphs $G{=}(V, E)$ and no edge list is requested.

Where extraction has been probed. Some of this work does examine extraction or attribute errors, but apart from the answer being scored. VisionGraph scores edge recognition as its own sub-task. An appendix of the Visual Graph Arena asks MLLMs to transcribe adjacency matrices for a few graphs. VKG-QA includes triple-extraction items and hand-attributes most sampled GPT-4o errors to perception. Graph-to-Vision scores parsing quality with an LLM judge.

What these evaluate: whether the final answer matches ground truth across graph families, task types, layouts, and scale, and, separately, how well structure is extracted. *What none of them evaluate, to our knowledge:* whether the answer agrees, per instance, with the structure the model extracted in the same response. Without that link, one cannot tell whether a correct answer rests on a correct structure, or whether a wrong one starts in extraction or in reasoning over what was extracted.

X.2 Layout, modality, and encoding

How a graph is drawn changes the answer. DynamicGTR compares five Graphviz layouts (`dot`, `neato`, `circo`, `fdp`, `sfdp`) with three textual encodings (Wei et al., 2026). Under a score that combines accuracy and brevity, cycle and connectivity questions favour drawings, while shortest-path, max-flow, and ordering tasks favour text; preferences are estimated per model. In bridge detection, circular drawings push MLLMs toward "no bridge" and spring drawings toward "bridge", regardless of ground truth (Brand et al., 2025). This motivates our Ring layout as a controlled stress condition. Graph Drawing for LLMs compares layout paradigms, drawing aesthetics, and prompts (Didimo et al., 2025). One prompt has the model spell out the adjacency list before solving, the same list-then-decide design as our List+Decide condition, but only the final answer is scored.

Image versus text. GraphTMI compares image, motif, and text encodings for node classification on citation networks (Das et al., 2024). A related case study extends the comparison to node-, edge-

, and graph-level tasks (Zhong & Mottin, 2024). GraphVerse argues that visual graph reasoning should not collapse into perception plus text-based QA (Sun et al., 2026). It uses graph-centric image edits and VGR-Score, in which an LLM judge checks each reasoning step against the true graph, and it hand-labels errors as perception or reasoning. CLEGR asks whether graph-language models use structure at all, and finds that on reasoning they rarely beat soft-prompted, text-only baselines (Petkar et al., 2026).

Graphs as text or embeddings. Outside the visual setting, much work examines how LLMs handle graphs given as text or as learned embeddings. This includes graph-problem benchmarks and textual-encoding studies (Wang et al., 2023a; Fatemi et al., 2024), graph-token and retrieval interfaces (Chen et al., 2024b; He et al., 2024), and planning and constrained decoding over knowledge graphs (Chen et al., 2024a; Luo et al., 2025). Separately, neural algorithmic reasoning trains GNNs to imitate classical algorithms, many of them on graphs (Veličković et al., 2020b;a; 2022), and reuses such executors for planning (Deac et al., 2021).

Extraction as a standalone task. Structure extraction has also been scored on its own: edge recognition in VisionGraph (Li et al., 2024), full node-and-edge parsing in DiGraphHal-Bench (Fan et al., 2026), flowchart-to-code link F1 in FlowLearn (Pan et al., 2024), and diagram-to-graph extraction in the concurrent TopoBench-180 (Guo et al., 2026).

What these evaluate: how layout, modality, or interface changes task accuracy, and, in separate tasks, how well structure is extracted. *What none of them evaluate, to our knowledge:* the extraction produced in the same response whose answer is scored, compared across renderings of the same graph.

X.3 Hallucination and faithfulness

Hallucination. Object- and relation-hallucination metrics and benchmarks catalogue errors on natural images (Rohrbach et al., 2018; Li et al., 2023; Wu et al., 2024). The problem persists, and can grow, under reasoning: in multimodal reasoning chains (Dong et al., 2025), with longer chains (Xu et al., 2025), and across scenes (Ross et al., 2025). It also appears in compositional and fine-grained queries (Yang et al., 2025; Bai et al., 2025), which motivates mitigation methods such as visual information steering (Li et al., 2025). Related benchmarks probe compositional (Hsieh et al., 2023) and context-sensitive, text-rich (Wadhawan et al., 2024) visual understanding.

Faithfulness of stated reasoning. Chain-of-thought prompting (Wei et al., 2022) and self-consistency (Wang et al., 2023b) improve multi-step reasoning. Yet fluent step-by-step explanations can be unfaithful to a model's actual computation (Turpin et al., 2023; Lanham et al., 2023). This matters for our list-then-decide condition. A model's stated edge list is exactly the kind of intermediate that should not be taken at face value, so we check it against both $G^\star$ and the named cycle.

Graph-specific hallucination. DiGraphHal-Bench shows that MLLMs hallucinate on complex directed graphs, including crossing edge pairs and cyclic-subpath queries (Fan et al., 2026). It also scores full node-and-edge parsing and shows that structure-focused fine-tuning helps downstream reasoning. It treats parsing and QA as separate tasks, though, and never checks an answer against the model's own extracted list. GraphInsight attributes text-only LLM failures on large graphs to positional bias: edges in the middle of the description are recalled worse (Cao et al., 2025). Our shuffled-list ablation (Appendix P) tests sensitivity to edge order and finds no significant effect. It does not isolate position effects, since shuffling moves edges between positions without removing them.

What these evaluate: whether an output or an intermediate description contains fabricated content, and whether stated reasoning matches the computation. *What none of them evaluate for graph structure, to our knowledge:* whether a fabricated element that reaches the final answer wins because the correct structure was never extracted, or because it was extracted but lost to a competing fabrication. Our loss/competition split is built to make this distinction.

Table 38: What the closest prior work scores about extracted structure. "Separate task": extraction is scored on its own, not against the answer being evaluated.

Work	Extracted structure scored	Answer checked against own list
VisionGraph (Li et al., 2024)	Edge recognition (separate task)	No
VGCure (Zhu et al., 2025)	Local only (neighbours, counts)	No
Visual Graph Arena (Babaiee et al., 2025)	Adjacency transcription, few graphs (appendix)	No
VKG-QA (Du et al., 2026)	Triple extraction (separate items)	No
GraphVerse (Sun et al., 2026)	Reasoning steps vs. true graph, LLM judge	No
DiGraphHal-Bench (Fan et al., 2026)	Full node/edge parsing (separate task)	No
Graph Drawing for LLMs (Didimo et al., 2025)	List elicited before solving, not scored	No
VGCompiler (Wang et al., 2026) (concurrent)	Adjacency-list F1 of its trained pipeline	n/a (pipeline)
Ours	Full edge list vs. $G^\star$, deterministic, four models	Yes, same response

X.4 THE UNRESOLVED GAP

What is still missing. Prior work scores a final answer, a named path or cycle, or an LLM-judged reasoning trace against ground truth, and sometimes scores extraction as a separate task. To our knowledge, none uses evaluation-time filtering of a model's own extraction to explain why an output is wrong. None separates *loss*, where the gold structure never entered the extracted list, from *competition*, where it survived inside a polluted list but lost to a fabricated competitor. The closest concurrent work, VGCompiler, reports the F1 of a recovered adjacency list and classifies the errors of its own trained pipeline (Wang et al., 2026). It does not audit the lists that frontier VLMs produce when prompted directly.

Three open questions. They decide what a fix should target:

1. *Where in the pipeline* does a wrong answer start: in reading the diagram, or in choosing among what was read?
2. *Where in the image* does reading break: on which geometric property of specific edges, rather than on whole-graph difficulty?
3. *What rule* picks the winner once a polluted list exists, and is that rule shared across models?

How we address them. We build on VisionGraph, which already moves beyond binary scoring and scores edge recognition as a separate sub-task. We instead verify the list each model writes in the same response as its cycle answer, against both $G^\star$ and the named cycle, for all four models. Edge lists and named cycles are explicit and checked deterministically, not judged by an LLM from a free-form trace as in GraphVerse's VGR-Score. This separates extraction, gold-cycle retention, list faithfulness, and verification as four jointly measured quantities, on Graphviz `dot` (Ellson et al., 2002) and equal-angle Ring (stress-test) renderings. It also traces cycle competition down to which fake cycle wins and why.

What we find. The answers are model-specific rather than one shared story (Sections 6–7; Appendices G–T):

- Wrong answers are mostly loss for Claude, Gemini, and Qwen, and mostly fake-cycle competition for GPT.
- Gemini is close to an ideal shortest-cycle selector.
- Claude and Qwen keep a residual error on the clean typed list. This is a direct single-call measurement whose cause is unresolved.
- Ring-adjacent invented edges feed fake cycles. For GPT and Gemini, these are largely consistent with a skip-chord clearance confound: a gold chord that skips one ring vertex passes close to it.