Are LLMs Truly Graph-Savvy? A Comprehensive Evaluation of Graph Generation

Ege Demirci, Rithwik Kerur, Ambuj Singh
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
{egedemirci,rkerur,ambuj}@ucsb.edu

Abstract

While large language models (LLMs) have demonstrated impressive capabilities across diverse tasks, their ability to generate valid graph structures remains underexplored. We evaluate fifteen state-of-the-art LLMs on five specialized graph generation tasks spanning delivery networks, social networks, quantum circuits, gene-disease networks, and transportation systems. We also test the LLMs using 3 different prompt types: direct, iterative feedback, and program-augmented. Models supported with explicit reasoning modules (o3-mini-high, o1, Claude 3.7 Sonnet, DeepSeek-R1) solve more than twice as many tasks as their general-purpose peers, independent of parameter count. Error analysis reveals two recurring failure modes: smaller parameter size Llama models often violate basic structural constraints, whereas Claude models respect topology but mismanage higher-order logical rules. Allowing models to refine their answers iteratively yields uneven gains, underscoring fundamental differences in error-correction capacity. This work demonstrates that graph understanding stems from specialized training methodologies rather than scale, establishing a framework for developing truly graph-savvy language models. Results and verification scripts available at github.com/Are-LLMs-Truly-Graph-Savvy.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing by achieving state-of-the-art performance on a diverse range of tasks, from translation and summarization to on-the-fly reasoning (Brown et al., 2020). Despite these impressive advancements in text generation, their ability to handle structured data, particularly graphs, remains work in progress. Graphs, which consist of nodes (representing entities) and edges (representing relationships), are fundamental to a wide spectrum of applications including social network analysis, biological systems modeling, and transportation planning. However, while LLMs demonstrate remarkable fluency in natural language, their performance in generating and reasoning about graph structures is often hindered by a persistent challenge: *hallucination*. In many cases, LLMs produce graph outputs that are syntactically plausible yet factually or structurally incorrect (Merrer and Tredan, 2024). While these failures are well documented on individual graph benchmarks, no broad, cross-domain evaluation has yet been performed.

Classical graph generation research offers two different paths: *parametric* deep generators such as GraphRNN, NetGAN, Graphite, GRAN and diffusion-based models (You et al., 2018; Bojchevski et al., 2018; Grover et al., 2018; Liao et al., 2019), and *non-parametric* construction methods that rewire or optimize graphs with commute-time or curvature objectives (Topping et al., 2022; Sterner et al., 2024). These prior approaches reliably satisfy hard structural constraints but lack the zero-shot flexibility and domain-aware semantics that make LLMs attractive for real-time graph design.

In this paper, our contribution is threefold:

(i) We introduce a novel evaluation framework comprising five specialized graph problems designed to challenge and assess LLMs' structural reasoning capabilities: (1) a Time-Dependent Delivery Network with complex spatiotemporal constraints; (2) a Directed Social Network with hierarchical influence relationships; (3) a Quantum Circuit Design requiring an understanding of quantum gate operations; (4) a Gene-Disease Association Network modeling bipartite relationships; and (5) an Optimal Transportation Network with robust connectivity requirements. These problems intentionally extend beyond conventional datasets to mitigate the effects of memorization, identified as confounding factors in the evaluation of LLM performance. Since these problems are open-ended,

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 884–897
July 28-29, 2025 ©2025 Association for Computational Linguistics

they allow for many structurally valid graphs instead of a single canonical solution. The model needs to explore a much larger design space and cannot simply *guess* a unique template, which increases the risk of hallucination and coverage failures.

(ii) We conduct a comprehensive evaluation using fifteen state-of-the-art LLMs spanning multiple architectural families and parameter scales. This selection enables us to conduct thorough comparisons across different architectures, which previous taxonomies by Ren et al. (2024) indicate are crucial for understanding the specific limitations of models in graph processing.

(iii) We systematically investigate three prompting paradigms: direct prompting, iterative feedback, and program-augmented prompting. Building upon the reasoning frameworks of the study, we examine whether these prompting approaches can effectively address the hallucination challenges documented by Tonmoy et al. (2024) and improve structural fidelity in the graph output.

1.1 Prior Work

We review prior attempts to evaluate LLM graph skills. Early efforts to explore the graph capabilities of LLMs have yielded promising but mixed results. Wu et al. (2025) introduce GraphEval36K, a 40-problem, 36 900-case coding benchmark that probes LLMs' algorithmic graph reasoning and highlights performance gaps between proprietary and open-source models. Yao et al. (2024) introduced *LLM4GraphGen*, which systematically evaluates the ability of LLMs to generate graphs based on structural rules and distributions. Their findings suggest that while models like GPT-4 exhibit some capacity for rule-based and distribution-based graph generation, conventional prompting methods (e.g., few-shot or chain-of-thought) do not consistently improve performance. In parallel, Wang et al. (2023) proposed the NLGraph benchmark, a set of graph reasoning tasks that ranges from basic connectivity checks to complex algorithmic challenges such as maximum flow and bipartite graph matching. Their study showed that while LLMs demonstrate preliminary reasoning abilities, their performance deteriorates as task complexity increases, and standard prompting strategies often fail to enhance results. Notably, both studies highlight that LLMs have difficulty generalizing beyond examples they have seen. This raises concerns about whether they genuinely learn graph structures or simply rely on memorization, and shows the need for more robust evaluations that go beyond standard datasets and assess LLMs' ability to construct and reason about unseen graphs.

Advances in reasoning-focused fine-tuning frameworks further illustrate both the potential and limitations of LLMs for graph-related tasks. The graph chain-of-thought (Graph-CoT) framework of Jin et al. (2024) promotes iterative reasoning by structuring LLM reasoning paths through explicit graph structures and demonstrating improved performance in complex graph-related inference tasks. Similarly, the Graph of Thoughts (GoT) framework introduced by Besta et al. (2024) models reasoning as a graph rather than a traditional tree, allowing LLMs to explore non-linear reasoning paths that better capture dependencies in structured data. Although these methods significantly improve reasoning accuracy, they do not fully address graph generation. Additionally, approaches such as the GCoder by Zhang et al. (2024) have explored integrating LLM with code-based methodologies to solve generalized graph problems, and have demonstrated substantial improvements over traditional natural language reasoning paradigms. Meanwhile, broader investigations into hallucination mitigation, such as the comprehensive survey by Tonmoy et al. (2024), underscore the need for more robust evaluation protocols that explicitly detect and quantify structural inconsistencies in graph outputs. These collective efforts indicate that while LLMs are becoming increasingly capable of handling graph-based reasoning, their ability to reliably generate novel, structurally valid graphs remains an open challenge requiring further study. Very recent work has begun using LLMs as agents that collaboratively grow dynamic social graphs (Chang et al., 2025; Ji et al., 2025). These studies reinforce the plausibility of LLM-driven graph construction but also document emergent biases and rule violations, echoing our motivation for a principled, multi-task evaluation.

Lastly, Merrer and Tredan (2024) examined how LLMs generate known graphs such as Zachary's Karate Club and Les Misérables. However, their approach is limited in scope as it relies on a small set of benchmark graphs, many of which are widely available in public datasets and may have been seen during model training. Furthermore, their evaluation is based on single-prompt interactions without testing the robustness of model responses across multiple attempts or under varied prompt condi-

tions. This narrow evaluation methodology fails to capture the broader generalization and reasoning abilities of LLMs in generating unseen graph structures, leaving critical questions unanswered regarding their ability to construct complex, structured graphs beyond memorization.

Through (i) crafting five diverse, unconstrained graph tasks, (ii) benchmarking fifteen distinct LLM architectures, and (iii) evaluating three prompting strategies, we offer a comprehensive evaluation of LLM graph-generation capabilities. Our results quantify current performance boundaries with statistical rigor and establish a reusable framework for assessing and improving structural fidelity in LLM outputs. Via our unique approach of targeting structural reasoning rather than memorization, we directly address the gap identified by recent surveys (Yu et al., 2025; Li et al., 2024), and take a step toward building graph-savvy language models that generate and reason about complex networks with higher fidelity and consistency.

2 Methodology

In this section, we describe the procedures used to design our five specialized graph-generation tasks, the verification pipeline for evaluating generated solutions, and the experimental setup employed to assess model performance. We evaluate the ability of Large Language Models (LLMs) to generate valid graphs using five tasks that each emphasize a distinct set of structural and logical challenges. These tasks are inspired by classical problem domains, including combinatorial optimization, network analysis, and biological systems modeling. Full prompts and constraints can be seen in the Appendix.

Time-Dependent Delivery Network: This scenario requires scheduling deliveries across multiple locations using a fleet of vehicles. Constraints include vehicle and storage capacities, dynamically adjusted travel times, and delivery time windows. It is similar to a time-windowed Vehicle Routing Problem (VRP) (Toth and Vigo, 2001) often encountered in logistics and supply-chain management, where resource utilization and schedule feasibility are essential.

Directed Social Network with Influence Relationships: We construct a social network in which users (categorized by trust scores) exert directed influence over others. The graph must remain acyclic while respecting category-based constraints (e.g., celebrities requiring sufficient outgoing edges). This setup reflects common problems in social network analysis (Amelkin and Singh, 2019), trust-based recommendation systems, and hierarchical structures where influence needs to be rigorously defined and free of feedback loops.

Quantum Circuit: This task involves organizing qubits, gates (single- and multi-qubit), and measurement operations under strict limitations on gate adjacency, temporal layering, and measurement rules. It mirrors quantum circuit scheduling challenges (Romero-Alvarez et al., 2024), where quantum gates must be placed in a Directed Acyclic Graph (DAG)-like structure, to ensure no conflicting operations and respect hardware constraints (such as non-adjacent CNOT requirements).

Gene-Disease Association Network: A bipartite graph is formed between genes and diseases, with each node set governed by specific degree constraints and edges indicating association strengths in the range [0.0,1.0]. In particular, our design draws inspiration from recent findings on the bipartite structure of vertebrate centromeres (Sacristan et al., 2024). This problem is an example of biological networks (e.g., gene-regulatory or gene-disease association mappings) that capture the confidence of links between genetic factors and clinical conditions. The valid bipartite structure and bounded association strengths are essential for realistic biological modeling.

Optimal Transportation Network: In this problem, LLMs need to develop a strongly connected, cost-effective, and resilient network of cities (nodes) and directed roads (edges). Important constraints include limits on road length and cost to ensure accessibility for the population. Additionally, the design should incorporate redundancy through multiple edges to enhance resilience (Medya et al., 2018). This problem is similar to multi-constraint transportation (Li et al., 2023) or flow networks, with a particular focus on two-edge robustness and minimizing path lengths to ensure that the network remains reliable and efficient under stress.

We evaluate a set of fifteen state-of-the-art LLMs, spanning multiple architectures and parameter sizes. These include GPT-4o (January 29 version), GPT-4o-mini, o1, and o3-mini-high by OpenAI (2024a,b,c); Claude 3.5 Sonnet, Claude 3.5 Haiku, and Claude 3.7 Sonnet (with extended thinking) by Anthropic (2024a,b,c); Gemini 2.0 Pro and Gemini 2.0 Flash by Google (2024a,b); Llama 3.1

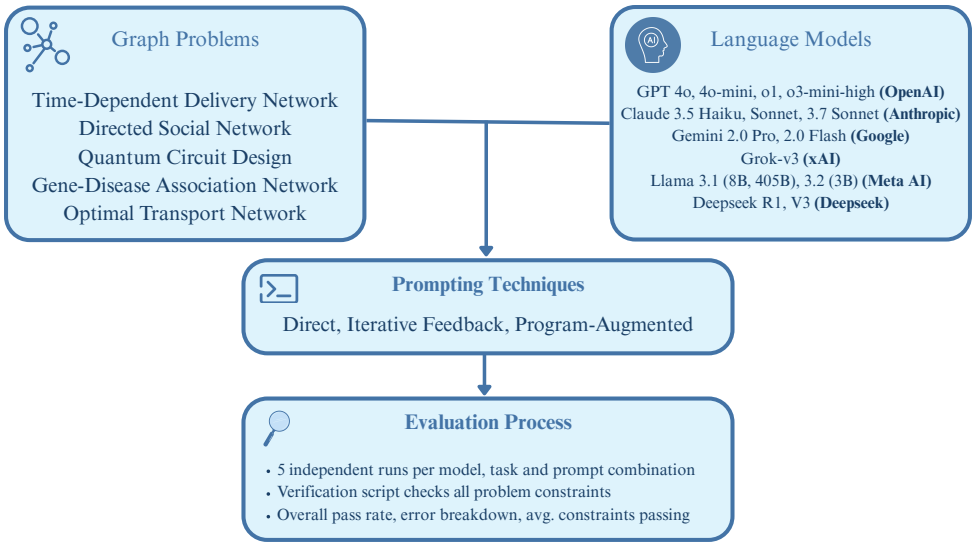

Figure 1: Experimental framework for evaluating LLMs' graph generation capabilities.

(8B), Llama 3.1 (405B), and Llama 3.2 (3B) by Meta AI (2024a,b); DeepSeek-V3 and DeepSeek-R1 by DeepSeek AI (2025, 2024); and Grok-V3 by xAI (2025). Models from the Llama family are run in Ollama (2025), allowing direct control over parameter settings and token decoding, while the remaining models are accessed through their respective chat-based interfaces following each provider's recommended prompt-completion protocol. We explore three prompting paradigms:

- *Direct Prompting:* The model receives a single, comprehensive prompt containing the entire task description, without additional feedback during generation.

- *Iterative Prompting:* After the initial direct prompt, if the model's output is unsatisfactory, it receives the verification script output as feedback. This feedback helps to refine the subsequent response, allowing for a multi-step corrective process.

- *Program-Augmented Prompting:* In the initial prompt, we include both the task description and the verification script. The model is encouraged to refer to this script during the generation process to self-assess and ensure that the output meets the specified structural requirements.

For each of the five tasks, we generate solutions using every model and prompting style combination across five independent runs. This approach is necessary because LLMs are inherently non-deterministic, meaning they can produce different responses to the same prompt due to the stochastic elements in their decoding processes. Conducting multiple independent runs allows us to capture this variability.

All models were evaluated in a *zero-shot* configuration: no demonstration examples were included in any prompt, even during iterative feedback. Each model received only the task description (and, for iterative prompting, the prior output plus verification feedback) without few-shot exemplars. Decoding parameters like temperature were left at their defaults for each interface to isolate the effects of the model architecture and prompting paradigm.

We save each generated output in a JSON file, which includes the graph definition (such as nodes and edges) and any numerical attributes (like costs and trust scores). After saving the output, we use a task-specific verification script to validate the generated graph. This script parses the JSON file into the required Python data structures and checks each constraint. During this process, any errors or constraints that are not met in the output are recorded in a separate JSON file. This file summarizes which constraints were satisfied and explicitly lists any errors made by the model. All violations are automatically mapped—via the predefined constraint groups lookup—to one of the three error categories (Structural, Logical, Attribute) by the verification script, so no manual post-processing is required.

We classify verification failures into three categories: **Structural**, **Logical**, and **Attribute**. Struc-

tural errors capture violations of global graph invariants, such as connectivity (e.g., missing a path that ensures two-edge robustness in the Optimal Transportation Network), acyclicity (e.g., the presence of a cycle in the Directed Social Network), and bipartite-constraint breaches (e.g., gene–gene edges in the Gene–Disease Association Network). Logical errors correspond to domain-specific rule violations, such as time-window compliance failures (deliveries scheduled outside the $[9, 11]$ window in the Time-Dependent Delivery Network), vehicle-capacity breaches (exceeding a vehicle's payload on a route), and strategic road-placement errors (insufficient outgoing edges from hub cities C0 or C7). Attribute errors refer to invalid node or edge metadata, for example, trust scores outside $[0, 100]$, undefined gate types or qubit labels in the Quantum Circuit Design, or association strengths outside $[0.0, 1.0]$ in the Gene–Disease network.

We then aggregate these files across the five runs, and look at the following metrics:

- **Overall Pass Rate:** The fraction of outputs that satisfy all constraints for a given (model, prompt style) pair.
- **Error Breakdown:** The frequency of constraint failures in structural vs. logical vs. attribute categories.
- **Average Constraint Passing:** The average count of successfully met constraints, offers more granularity than a strict pass/fail.

Finally, we compile all verification reports to create a per-run summary of pass/fail outcomes. Another report aggregates the results at the model and prompting method level, computing average pass rates and error counts across the five runs.

3 Results

Our evaluation reveals variations in graph generation capabilities among state-of-the-art LLMs, providing empirical evidence on the extent to which LLMs are genuinely graph-savvy. The results show critical insights into architectural differences, the efficacy of different prompting strategies, and the distinctive challenges posed by structured graph problems.

3.1 Performance Stratification Across Model Architectures

As shown in Figure 2(c), we observe a pronounced stratification in performance across model families, with specialized reasoning models demonstrating markedly superior capabilities. o3-mini-high and o1 (OpenAI's reasoning-focused models released in January 2025 and December 2024, respectively) achieved exceptional performance with average pass rates of 82.7% and 78.7%, substantially outperforming the cross-model average of 34.0%. Claude 3.7 Sonnet, Anthropic's hybrid reasoning model released in February 2025, followed with a 69.3% success rate, while DeepSeek-R1, another reasoning-specialized architecture, achieved a 48.0% pass rate.

This performance distribution aligns with our hypothesis that graph generation requires sophisticated structural reasoning beyond basic pattern recognition. Notably, the four models fine-tuned with enhanced reasoning capabilities (o3-mini-high, o1, Claude 3.7 Sonnet, and DeepSeek-R1) occupy four of the top five positions in overall performance, suggesting that training methodologies targeting complex reasoning transfer effectively to graph-related tasks.

In contrast, smaller parameter-count models and those without explicit reasoning enhancements struggled significantly. Llama 3.1 (8B) and Llama 3.2 (3B) achieved only 1.3% pass rates, while ChatGPT 4o-mini reached just 14.7%, indicating fundamental limitations in graph representation abilities. This pattern supports our premise that graph generation constitutes a distinctive challenge requiring specialized architectural capabilities rather than merely scaling parameters. Although scaling parameters increases the performance of the model, in the case of Llama 3.1, it does not bring it close to any of the 4 models with reasoning enhancements.

3.2 Problem-Specific Performance

The performance gradient across tasks remained consistent across model families: **the Time-Dependent Delivery Network** presented the greatest challenge (with error counts averaging 18-49 for most models under direct prompting), followed by **the Gene-Disease Association Network** (10-38 errors). This hierarchy persisted despite iterative feedback, suggesting fundamental differences in task complexity rather than mere prompting limitations. The consistency of this pattern indicates that temporal reasoning with multiple interacting constraints presents a qualitatively different challenge compared to static structural properties.

Error analysis reveals that failures in **the Directed Social Network** stemmed primarily from

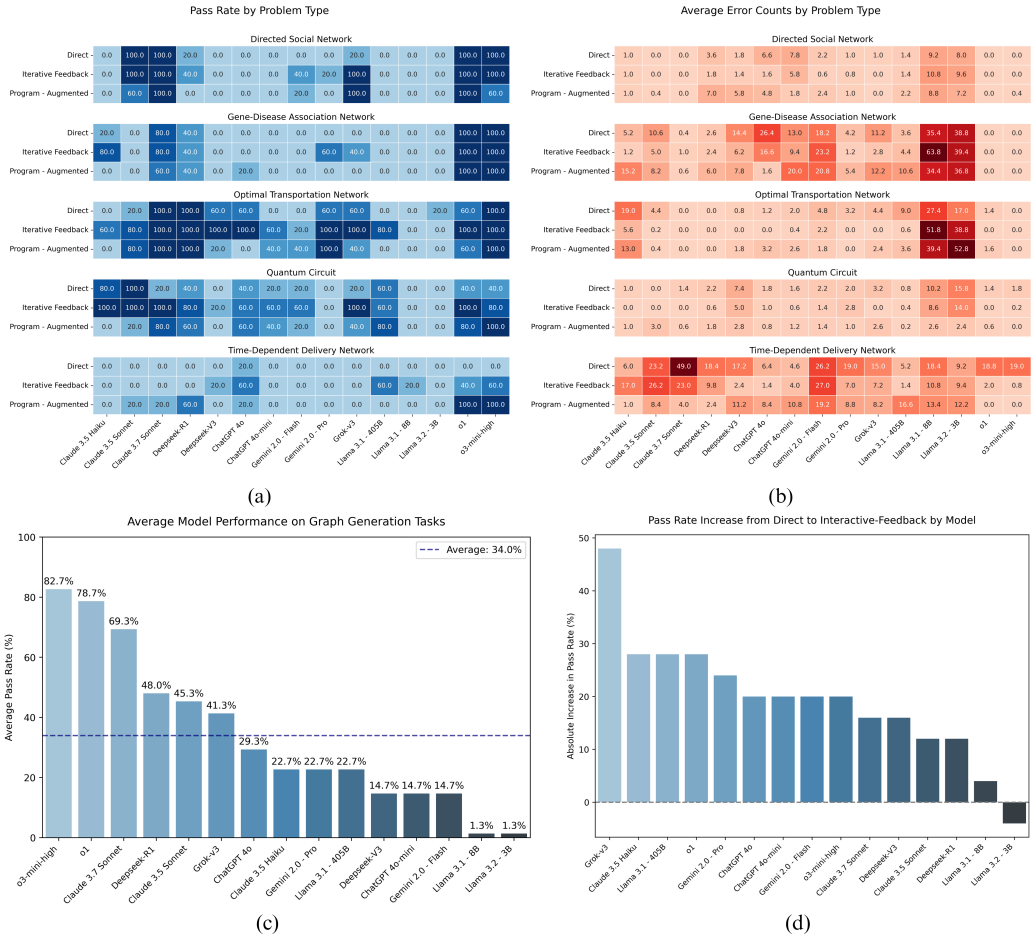

Figure 2: **Performance analysis of LLMs on graph generation tasks.** Figure panels summarize key trends across fifteen LLMs and five problem domains. **(a)** Pass rates per model and task reveal that only a few models consistently satisfy all constraints across problems, with stronger results under iterative prompting. **(b)** Error heatmaps show the specific types of graphs that each model struggles with. **(c)** Average pass rates across all tasks highlight the performance stratification between reasoning-enhanced and general-purpose models. **(d)** Performance deltas from iterative feedback quantify each model's ability to self-correct, with Grok-v3 showing the largest improvement.

specific constraint violations. The Claude Sonnet family showed minimal errors, averaging between 0 and 1 errors per run, while others, like ChatGPT 4o, produced between 6.6 and 7.8 errors under direct prompting, particularly regarding celebrity outgoing edge requirements. Furthermore, specialized reasoning models exhibited a better ability to uphold global structural properties like acyclicity. The deliberately introduced gap in trust score categorization (50-70) shows a consistent tendency across models to hallucinate classifications for these ambiguous values rather than adhering strictly to provided rules. This classification completion bias persisted across multiple prompt iterations especially for simpler models, suggesting an intrinsic tendency to complete perceived patterns rather than strictly adhering to explicit constraints. This is a concerning finding for domain applications requiring rigid adherence to rules.

The Gene-Disease Association task shows another structural pattern. Traditional LLMs struggled specifically with maintaining bipartite integrity (creating forbidden gene-gene or disease-disease connections) and balancing degree constraints simultaneously. Llama 3.1 (405B) generated 35.4 errors on average under direct prompting, with approximately 70% related to bipartite violations and degree constraint failures. Even with iterative feedback, these models continued to generate structurally invalid networks, suggesting a fundamental difficulty in conceptualizing strict categorical separation between node types. In contrast, reasoning-specialized models primarily made errors in strength attribute assignments while maintaining valid bipartite structures.

For **the Quantum Circuit** task, lower-performing models like Llama 3.1 (8B) and DeepSeek-V3 (which recorded 7.4 errors under direct prompting) primarily struggled with gate adjacency requirements and constraints related to layered operations. This led to the creation of technically invalid quantum circuits. In contrast, errors from Claude and OpenAI models focused more on gate optimization and final state compliance. These were more subtle violations that resulted in operationally valid but suboptimal circuits. This pattern suggests a hierarchy in understanding quantum circuits, where basic structural validity must be established before addressing optimization capabilities. The tendency to selectively violate constraints indicates that domain-specific requirements may be overshadowed by more familiar structural patterns, which raises concerns for specialized domain applications.

The Optimal Transportation Network task revealed a distinctive error pattern focusing on cost-distance consistency and accessibility requirements. Even models with high overall pass rates struggled with balancing mutually constraining objectives: Smaller parameter Llama models (8B, 3B) generated 27.4-38.8 errors under direct prompting, primarily violating strategic road placement constraints while maintaining valid connectivity. In contrast, reasoning models made significantly fewer errors (0-1.4) and effectively balanced multiple competing constraints. This suggests that multi-objective optimization in graphs represents a distinctive capability of reasoning-enhanced architectures that general-purpose models have not yet mastered.

The most pronounced error pattern emerged in **the Time-Dependent Delivery Network task**, where even high-performing models exhibited cascading failure modes. Error analysis reveals that violations typically began with time window inconsistencies that propagated to vehicle capacity and storage compliance failures. Claude 3.7 Sonnet's unusually high error count (49.0) under direct prompting stems primarily from creating temporally impossible delivery sequences that subsequently violated multiple dependent constraints. This suggests that temporal reasoning in graphs triggers a distinctive failure mode where local inconsistencies propagate through interconnected constraint networks.

Furthermore, across multiple problems, we observed that models frequently generated locally valid edges (satisfying pairwise constraints) that violated global structural properties such as acyclic or strong connectivity. This pattern suggests a limitation in maintaining coherent global graph properties while simultaneously satisfying local edge constraints. This finding has significant implications for applications requiring global structural guarantees.

These detailed error patterns across problem domains collectively indicate that graph hallucination is not a uniform phenomenon but manifests differently depending on the structural properties required. Reasoning-enhanced models demonstrate superior constraint reconciliation abilities, particularly for maintaining global structural properties while satisfying local edge constraints, which is a critical capability for real-world graph applications.

3.3 Constraint Satisfaction by Category

Figure 3(e) demonstrates that reasoning-enhanced models (o3-mini-high, o1, Claude 3.7 Sonnet, and DeepSeek-R1) consistently passed 10-12 structural constraints regardless of prompting strategy. This suggests that structural reasoning capabilities emerge from reasoning-focused training rather than prompt engineering alone.

Figure 3(f) reveals greater variability in logical constraint satisfaction, with iterative feedback substantially improving performance across most models (e.g., Grok-v3 improving from 11.6 to 14.0). This differential responsiveness suggests that logical constraints, which often require multi-step reasoning about consequences, benefit most from decomposed reasoning in iterative feedback loops, aligning with prior findings on step-by-step reasoning (Jin et al., 2024).

Figure 3(g) reveals that attribute constraints pose a relatively manageable challenge for most models, with top-performing reasoning models like Claude 3.7 Sonnet, o1, and o3-mini-high consistently achieving perfect or near-perfect scores of 9.0 passed constraints. Even models with moderate overall performance generally exhibited strong attribute constraint satisfaction, suggesting that handling spatial, quantitative, and categorical graph properties represents a more tractable aspect of graph generation compared to structural or logical constraints for current LLM architectures.

3.4 The Efficacy of Prompting Paradigms

As quantified in Figure 2(d), the improvement from direct prompting to iterative feedback varied dra-

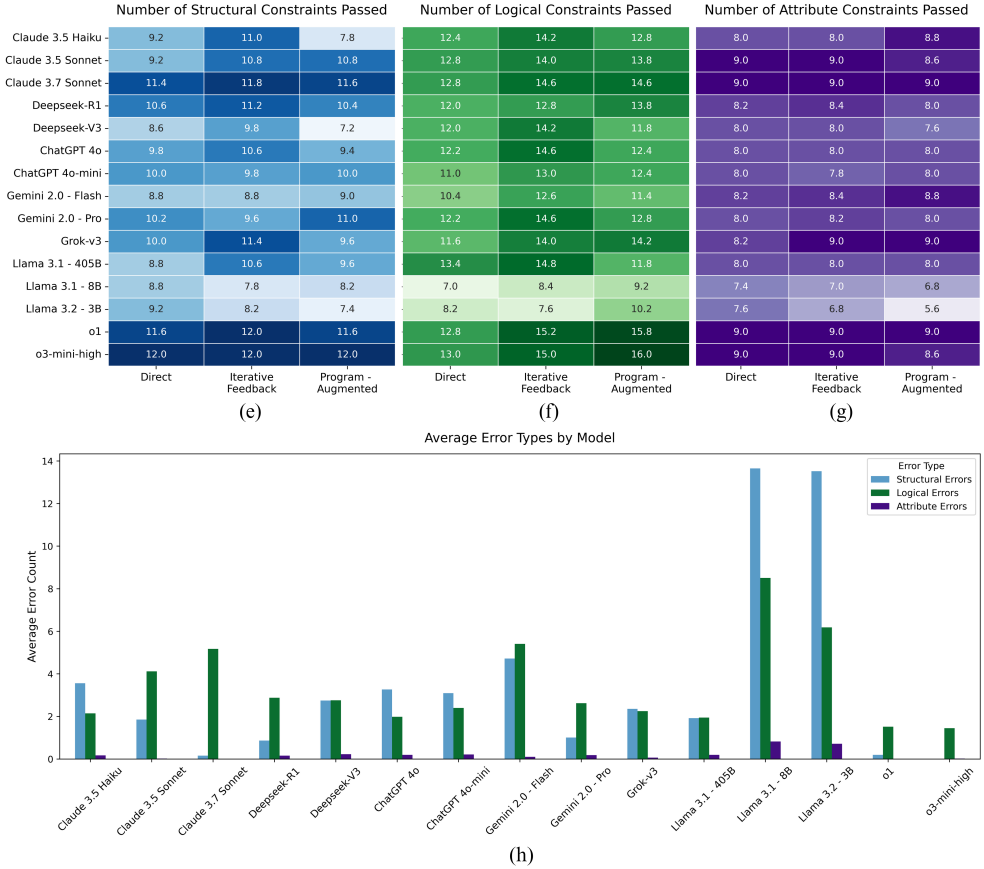

Figure 3: **Constraint satisfaction and error analysis.** Breakdown of model performance across constraint types and error categories. **(e–f-g)** show the average number of structural, logical, and attribute constraints passed per model and prompting strategy. Reasoning-enhanced models (e.g., o1, o3-mini-high, Claude 3.7 Sonnet) consistently score higher, especially on logical constraints. **(h)** displays average error types by model, revealing that Llama models tend to accumulate structural errors, while Claude models exhibit a higher proportion of logical errors. This analysis shows consistent error signatures across architectures and shows that constraint handling is both task-specific and model-dependent.

matically across model families. Grok-v3 exhibited a striking 48% absolute increase, while reasoning-specialized models showed more modest gains (16-28%), suggesting these models possess inherent graph reasoning capabilities less dependent on external guidance. Among the smaller Llama variants (3B and 8B), we observed only minimal improvement (less than 5%). However, the 405B model demonstrated a significant increase of approximately 30% with iterative prompting. This suggests that while increasing model size can help reduce some limitations, it does not completely eliminate them.

Contrary to our hypothesis, program-augmented prompting, which provided explicit verification code, did not consistently outperform iterative feedback and sometimes produced worse results than direct prompting. This finding challenges assumptions about LLMs' ability to leverage programmatic verification during generation and suggests limitations in code comprehension or self-monitoring capabilities. The pattern aligns with Zhang et al. (2024)'s findings that code-based methodologies require tight integration with model architecture rather than simply being provided as context.

3.5 Error Patterns

Figure 3(h) shows distinctive error patterns across model families that illuminate the nature of graph hallucination:

We identified two predominant error patterns: (1) models with high structural but low logical errors (smaller parameter Llama family), suggesting fundamental difficulty with graph topology; and (2) models with low structural but moderate

logical errors (Claude Sonnet family), indicating stronger topological understanding but challenges with constraint reasoning. These distinct profiles suggest different mechanisms underlying graph hallucination across architectures. OpenAI's models (o1, o3-mini-high) displayed remarkably balanced and minimal error profiles across all categories, while Llama models exhibited compounded failures across structural, logical, and attribute dimensions. Anthropic models showed moderate but balanced error distributions, suggesting a more comprehensive but imperfect graph understanding. These distinctive signatures indicate that architectural design decisions create consistent patterns in graph processing capabilities that transcend individual prompting strategies or task types.

4 Discussion

Our thorough evaluation of fifteen advanced LLMs across five different graph generation tasks provides an insightful answer to the question: "Are LLMs truly graph-savvy?" Our results show that proficiency in graph generation varies markedly across models. Instead, it is closely linked to the design of the models, especially those enhancements that focus on improving reasoning capabilities. Our findings have several important theoretical implications for the development of graph-capable language models:

The consistent superiority of reasoning-enhanced models (o3-mini-high, o1, Claude 3.7 Sonnet, DeepSeek-R1) over larger but general-purpose architectures indicates that graph reasoning requires reasoning-focused training regimens rather than merely scaling parameters or training data. This contradicts the notion that larger models will naturally develop sophisticated graph reasoning, suggesting instead that training innovations specifically targeting complex reasoning are necessary.

The pronounced performance gaps across problem types challenge the notion of general graph reasoning capabilities. Models that excelled at optimal transportation networks often struggled with time-dependent delivery networks, suggesting that LLMs develop domain-specific structural competencies that transfer imperfectly across problem domains. This domain-specificity has implications for applications requiring cross-domain generalization.

The variable efficacy of prompting strategies across model families indicates that prompting can enhance but not fundamentally transform an architecture's graph processing capabilities, challenging perspectives that view prompting as a substitute for architectural innovation. This suggests that prompting should be viewed as complementary to, rather than a replacement for, architectural improvements.

Despite our comprehensive evaluation, several limitations should be acknowledged. First, our iterative feedback paradigm utilized only a single round of feedback, potentially limiting the improvements possible through iterative correction. Future work could explore multi-step interactive protocols that better leverage the potential of decomposed reasoning to address complex graph constraints. Second, while our five graph problems span diverse domains, they represent only a subset of possible graph structures and constraint types. Expanding the evaluation to include additional problem domains such as knowledge graphs, molecule generation, and program synthesis graphs would provide a more comprehensive assessment of LLMs' graph capabilities. Third, our evaluation focused primarily on constraint satisfaction rather than generative creativity or optimization quality. Future work could explore how models balance adherence to constraints with the generation of novel or optimal graph structures, particularly in open-ended design tasks. Finally, the black-box nature of many commercial LLMs limits our ability to analyze the underlying mechanisms responsible for performance differences. Future research could benefit from more transparent model architectures that enable detailed analysis of how graph structures are represented and manipulated internally. These limitations suggest several promising directions for future research. The development of specialized fine-tuning approaches for graph-related tasks could address the observed domain transfer limitations. Hybrid architectures that combine LLMs with graph neural networks or constraint satisfaction solvers might use the complementary strengths of different approaches. In conclusion, our findings demonstrate that while recent architectural advances have significantly improved graph generation capabilities, LLMs' graph-savviness remains highly dependent on architectural design, with specialized reasoning capabilities playing a crucial role. Future advances will likely come from architectural or training innovations specifically targeting structured reasoning rather than simply scaling existing models or refining prompting strategies.

References

Victor Amelkin and Ambuj K. Singh. 2019. Fighting opinion control in social networks via link recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 677–685, New York, NY, USA. Association for Computing Machinery.

Anthropic. 2024a. Claude 3.5 haiku. https://www.anthropic.com/claude/haiku. Large Language Model.

Anthropic. 2024b. Claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet. Large Language Model.

Anthropic. 2024c. Claude 3.7 sonnet. https://www.anthropic.com/news/claude-3-7-sonnet. Large Language Model.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. 2018. Netgan: Generating graphs via random walks. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 609–618. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Serina Chang, Alicja Chaszczewicz, Emma Wang, Maya Josifovska, Emma Pierson, and Jure Leskovec. 2025. Llms generate structurally realistic social networks but overestimate political homophily. In *ICWSM*, pages 341–371. AAAI Press.

DeepSeek AI. 2024. Deepseek-v3. https://api-docs.deepseek.com/news/news1226. Large Language Model.

DeepSeek AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Google. 2024a. Gemini 2.0 flash. https://deepmind.google/technologies/gemini/flash/. Large Language Model.

Google. 2024b. Gemini 2.0 pro. https://deepmind.google/technologies/gemini/pro/. Large Language Model.

Aditya Grover, Aaron Zweig, and Stefano Ermon. 2018. Graphite: Iterative generative modeling of graphs. *CoRR*, abs/1803.10459.

Jiarui Ji, Runlin Lei, Jialing Bi, Zhewei Wei, Xu Chen, Yankai Lin, Xuchen Pan, Yaliang Li, and Bolin Ding. 2025. Llm-based multi-agent systems are scalable graph generative models. *Preprint*, arXiv:2410.09824.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 163–184, Bangkok, Thailand. Association for Computational Linguistics.

Xinguang Li, Jun Zhan, Fuquan Pan, Tong Lv, and Shen Wang. 2023. A multi-objective optimization model of urban passenger transportation structure under low-carbon orientation considering participating subjects. *Environmental Science and Pollution Research*, 30(54):115839–115854.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2024. A survey of graph meets large language model: Progress and future directions. *Preprint*, arXiv:2311.12399.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard Zemel. 2019. Efficient graph generation with graph recurrent attention networks. In *NeurIPS*.

Sourav Medya, Arlei Silva, Ambuj Singh, Prithwish Basu, and Ananthram Swami. 2018. Group centrality maximization via network design. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 126–134. SIAM.

Erwan Le Merrer and Gilles Tredan. 2024. Llms hallucinate graphs too: a structural perspective. *Preprint*, arXiv:2409.00159.

Meta AI. 2024a. Llama 3.1. https://ai.meta.com/blog/meta-llama-3-1/. Large Language Model.

Meta AI. 2024b. Llama 3.2. https://huggingface.co/meta-llama/Llama-3.2-1B. Large Language Model.

Ollama. 2025. ollama/ollama: Get up and running with llama 3.3, deepseek-r1, phi-4, gemma 3, mistral small 3.1 and other large language models. https://github.com/ollama/ollama. Version v0.7.0, accessed 2025-05-16.

OpenAI. 2024a. Gpt-4o. https://openai.com/index/hello-gpt-4o/. Large Language Model.

OpenAI. 2024b. o1. https://openai.com/o1/. Large Language Model.

OpenAI. 2024c. o3-mini-high. https://openai.com/index/openai-o3-mini/. Large Language Model.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6616–6626, New York, NY, USA. Association for Computing Machinery.

Javier Romero-Alvarez, Jaime Alvarado-Valiente, Jorge Casco-Seco, Enrique Moguel, Jose Garcia-Alonso, and Juan M. Murillo. 2024. Scheduling Process of Quantum Circuits to Optimize Tasks Execution on Quantum Computers . In *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 182–186, Los Alamitos, CA, USA. IEEE Computer Society.

Carlos Sacristan, Kumiko Samejima, Lorena Andrade Ruiz, Moonmoon Deb, Maaike L.A. Lambers, Adam Buckle, Chris A. Brackley, Daniel Robertson, Tetsuya Hori, Shaun Webb, Robert Kiewisz, Tristan Bepler, Eloïse van Kwawegen, Patrik Risteski, Kruno Vukušić, Iva M. Tolić, Thomas Müller-Reichert, Tatsuo Fukagawa, Nick Gilbert, and 3 others. 2024. Vertebrate centromeres in mitosis are functionally bipartite structures stabilized by cohesin. *Cell*, 187(12):3006–3023.e26.

Igor Sterner, Shiye Su, and Petar Veličković. 2024. Commute-time-optimised graphs for gnns. *Preprint*, arXiv:2407.08762.

S. M Towhidul Islam Tonmoy, S M Mehedi Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *Preprint*, arXiv:2401.01313.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *ICLR*. OpenReview.net.

Paolo Toth and Daniele Vigo, editors. 2001. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, USA.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Thirty-seventh Conference on Neural Information Processing Systems*.

Qiming Wu, Zichen Chen, Will Corcoran, Misha Sra, and Ambuj Singh. 2025. GraphEval36K: Benchmarking coding and reasoning capabilities of large language models on graph datasets. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 8095–8117, Albuquerque, New Mexico. Association for Computational Linguistics.

xAI. 2025. Grok-v3. https://x.ai/blog/grok-3. Large Language Model.

Yang Yao, Xin Wang, Zeyang Zhang, Yijian Qin, Ziwei Zhang, Xu Chu, Yuekui Yang, Wenwu Zhu, and Hong Mei. 2024. Exploring the potential of large language models in graph generation. *Preprint*, arXiv:2403.14358.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5694–5703. PMLR.

Shuo Yu, Yingbo Wang, Ruolin Li, Guchun Liu, Yanming Shen, Shaoxiong Ji, Bowen Li, Fengling Han, Xiuzhen Zhang, and Feng Xia. 2025. Graph2text or graph2token: A perspective of large language models for graph learning. *Preprint*, arXiv:2501.01124.

Qifan Zhang, Xiaobin Hong, Jianheng Tang, Nuo Chen, Yuhan Li, Wenzhong Li, Jing Tang, and Jia Li. 2024. Gcoder: Improving large language model for generalized graph problem solving. *Preprint*, arXiv:2410.19084.

Appendix: Graph Generation Problem Statements

This appendix contains the detailed problem statements for the five graph generation tasks used in our evaluation framework.

A.1 Time-Dependent Delivery Network

Problem Description:
Create a delivery network that schedules deliveries across multiple locations using a fleet of vehicles. The network must account for vehicle capacities, location storage capacities, delivery time windows, dynamic travel times, and vehicle speeds to ensure efficient and timely deliveries.

Constraints:

1. **Locations:**
   - **Total Locations:** 15, labeled from `L0` to `L14`.
   - **Attributes:**
     - **Storage Capacity:** Each location has a storage capacity specified in kilograms (kg). Example: `L0` has a capacity of 500 kg.
     - **Time Window:** Each location has a delivery time window represented as a list of two integers `[start_hour, end_hour]` in 24-hour format. Example: `L3` has a time window of `[9, 11]` corresponding to 09:00-11:00.
2. **Vehicles:**

- **Total Vehicles:** 7, labeled from `V1` to `V7`.
- **Attributes:**
  - **Capacity:** Each vehicle has a specific capacity in kilograms (kg). Example: `V1` has a capacity of 100 kg.
  - **Speed:** Each vehicle has a defined speed in kilometers per hour (km/h). Example: `V1` travels at 60 km/h.

3. **Edges (Routes):**

- **Definition:** Represents travel paths between two distinct locations.
- **Attributes:**
  - **From:** The starting location ID (e.g., `L0`).
  - **To:** The destination location ID (e.g., `L1`).
  - **Base Travel Time:** The fundamental travel time for the route in minutes.
  - **Hourly Adjustments:** A dictionary where keys are time ranges in the format `"HH-HH"` (24-hour format) and values are additional travel time in minutes applicable during those hours. Example: `{"8-10": 15}` adds 15 minutes to the base travel time between 08:00-10:00.
  - **Maximum Weight Limit:** The maximum weight a vehicle can carry on that route in kilograms (kg).

4. **Operational Constraints:**

- **Storage Capacity Compliance:** The sum of incoming goods to any location must not exceed its storage capacity.
- **Vehicle Capacity Compliance:** No vehicle should exceed its capacity on any edge it traverses.
- **Time Window Compliance:** Departures and arrivals must respect the time windows of locations. Specifically:
  - **Departure Time:** Must be within the `from` location's time window.
  - **Arrival Time:** Must be within the `to` location's time window.
  - **Loading Time:** Assume a fixed loading time of 10 minutes at each location, which must be accounted for when scheduling departures.

Required Output Format:
<FORMAT>

A.2 Directed Social Network with Influence Relationships

Problem Description:
Create a social network graph representing influence relationships among users. Each user has specific attributes, and influence connections must adhere to defined constraints to maintain the integrity and intended structure of the network.

Constraints:

1. **Users:**

- Total of 20 users labeled from U0 to U19.
- Each user has a "trust_score" ranging from 0 to 100.
- Each user belongs to a "category" based on their trust score:
  - "celebrity" (trust_score $\geq$ 80)
  - "expert" (70 $\leq$ trust_score < 80)
  - "regular" (trust_score < 50)

2. **Edges (Influence Relationships):**

- Directed edges where Ux $\rightarrow$ Uy indicates that Ux influences Uy.
- **No self-loops:** A user cannot influence themselves.
- **Category Constraints:**
  - **Celebrities:** Must have at least 5 outgoing edges.
  - **Regular Users:** Cannot influence experts.
- **Graph Structure:**
  - The graph must be acyclic (no cycles in influence relationships).

Required Output Format:
<FORMAT>

A.3 Quantum Circuit Design

Problem Description:
Design a quantum circuit consisting of multiple qubits and quantum gates. The circuit must adhere to specific constraints to ensure proper gate operations, circuit efficiency, and overall functionality. The design should incorporate structural elements like depth and a Directed Acyclic Graph (DAG) while simplifying some of the gate-related rules to enhance accessibility.

Constraints:

1. **Qubits:**

- **Total Qubits:** 10, labeled from `Q0` to `Q9`.
- **Initialization:** All qubits must start in the $|0\rangle$ state.

2. **Gates:**
   - **Types of Gates to Include:**
     - **Single-Qubit Gates:** Hadamard (H), Pauli-X (X), Pauli-Z (Z)
     - **Multi-Qubit Gates:** Controlled NOT (CNOT), SWAP
     - **Measurement:** Measure (Measure)
   - **Gate Operations:**
     - Each gate operates on specific qubits at designated times.
     - **CNOT Gates:** Must operate on qubits that are not adjacent (e.g., Q0 and Q2 are valid; Q0 and Q1 are invalid).
     - **SWAP Gates:** Must operate between pairs of qubits that have identical gate sequences up to that point.
     - **Measurements:** Each qubit can be measured only once and must be the last operation on that qubit.
   - **Gate Restrictions:**
     - **Gate Frequency:** No single-qubit gate can be applied more than twice consecutively on the same qubit.

3. **Circuit Structure:**
   - The circuit must be a Directed Acyclic Graph (DAG); no repeated times for the same qubit.
   - **Layered Operations:** Gates at the same time step must operate on disjoint sets of qubits (i.e., no two gates at the same time can act on the same qubit).
   - **Depth Constraint:** The total number of time steps (layers) must not exceed 30.

4. **Operational Constraints:**
   - **Circuit Reversibility:** Measurements must be the final operations on their respective qubits to maintain circuit reversibility.
   - **Gate Optimization:** The circuit should minimize the total number of gates while satisfying all other constraints.
   - **Final State:** After all operations, all qubits must either be measured or returned to the $|0\rangle$ state.

Required Output Format:
<FORMAT>

A.4 Gene-Disease Association Network

Problem Description:
Create a bipartite network that models the associations between genes and diseases. This network will represent which genes are associated with which diseases, capturing the strength of each association. The network should adhere to defined constraints to ensure biological relevance and structural integrity.

Constraints:

1. **Nodes:**
   - **Genes:**
     - Total of 20 genes labeled from G0 to G19.
     - Each gene has a "name" and a "function".
   - **Diseases:**
     - Total of 20 diseases labeled from D0 to D19.
     - Each disease has a "name" and a "severity_level" (e.g., "Low", "Medium", "High").

2. **Edges (Associations):**
   - Represents the association between a gene and a disease.
   - **Bipartite Constraint:** Associations can only exist between genes and diseases, not within the same set.
   - **Association Strength:** Each association has a "strength" value ranging from 0.0 to 1.0, indicating the confidence of the association.

3. **Degree Constraints:**
   - **Genes:**
     - Each gene must be associated with at least 2 and at most 5 diseases.
   - **Diseases:**
     - Each disease must be associated with at least 3 and at most 10 genes.

4. **Structural Constraints:**
   - The network must be bipartite; no edges should connect nodes within the same set (i.e., no gene-gene or disease-disease associations).

- There should be no duplicate edges (i.e., each gene-disease pair is unique).

Required Output Format:
<FORMAT>

A.5 Optimal Transportation Network

Problem Description:
Design an optimal transportation network represented as a **directed graph** where nodes represent cities and edges represent one-way roads. The network must satisfy constraints to ensure efficiency, connectivity, robustness, and cost-effectiveness.

Constraints:

1. **Nodes (Cities):**
   - **Total:** 8, labeled from `C0` to `C7`.
   - **Attributes:**
     - **Population:** Number of inhabitants in each city.
       * `C0`: 1,000
       * `C1`: 500
       * `C2`: 750
       * `C3`: 600
       * `C4`: 900
       * `C5`: 400
       * `C6`: 800
       * `C7`: 650

2. **Edges (Roads):**
   - **Definition:** Represents a one-way road from one city to another.
   - **Attributes:**
     - **Distance:** Length of the road in kilometers (km). *(Each road must be $\leq$ 300 km.)*
     - **Construction Cost:** Cost to build the road in thousand dollars ($K).

3. **Additional Constraints:**
   (a) **Connectivity:** The network must be **strongly connected**, meaning there is a directed path from any city to every other city.
   (b) **Road Capacity:** No single road should be longer than **300 km**.
   (c) **Cost Optimization:** The **total construction cost** of all roads should not exceed **$10,000K**.
   (d) **Population Accessibility:** Each city must have **at least two incoming roads** to ensure redundancy and accessibility.
   (e) **Strategic Road Placement:** Cities `C0` and `C7` are major hubs and **must have at least three outgoing roads** each to distribute traffic efficiently.
   (f) **Avoiding Redundancy: No two cities** should have more than **one direct road** connecting them in the same direction.
   (g) **Minimizing Total Distance:** The **sum of all road distances** should be minimized to ensure efficient transportation.
   (h) **2-Edge Robustness:** The network must remain strongly connected if **any single road is removed** (i.e., there must be two edge-disjoint paths between every ordered pair of cities).
   (i) **Edge-Disjoint Paths Guarantee:** For every pair of distinct cities, there must exist **at least two completely independent (edge-disjoint) paths** connecting them.
   (j) **Balanced Outgoing Degree:** Except for the designated hubs (`C0` and `C7`), the difference between the maximum and minimum number of outgoing roads among all cities must not exceed **2**. This prevents "overloaded" junctions.
   (k) **Path Efficiency Constraint:** For every pair of cities, the shortest route (by total distance) should be less than **500 km** to ensure quick intercity transit.
   (l) **Cost–Distance Consistency:** For every road, the construction cost (in $K) must be **between 1.0 and 1.5 times its distance (in km)**. *Example:* A road that is 90 km long must have a cost between **90K** and **135K**.
   (m) **Maximum Edge-Hop Constraint:** For every pair of cities, you need to be able to get to every other city in at most 3 edges.

Required Output Format:
<FORMAT>