BENCHMARKING LARGE LANGUAGE MODELS ON NETWORK OPTIMIZATION

Ethan Cui

Foothill College, Los Altos Hills, USA

ABSTRACT

The most recent large language models (LLMs) have impressive problem-solving capabilities in tasks such as code generation and completing math problems. However, a majority of current benchmarks on recent LLMs test only on academic or competition style questions, leading to gaps in our understanding of the capabilities of LLMs. This paper introduces the first benchmark that evaluates LLMs on network optimization problems from operations research, including shortest-path routing, min-cost flow, and multicommodity flow. Such problems test LLM's abilities to both follow structured instructions and conforming to complex constraints along with long-term reasoning on complex tasks. By controlling the complexity of synthetically generated instances, we reveal critical differences in the reasoning and planning capabilities of seven frontier models. Our work bridges the Operations Research (OR) and LLM communities, positioning network optimization as a rigorous and scalable framework for evaluating and advancing LLM reasoning.

KEYWORDS

Network Flow Optimization, Large language model, Model Evaluation

1. INTRODUCTION

The most recent large language models (LLMs) have impressive problem-solving capabilities in tasks such as code generation and completing math problems. Benchmarks such as GSM8K, MATH, SAT, OCW, MMLU STEM, AIME2024 [1, 2, 3, 4, 5, 6, 7] have played a pivotal role in charting progress, however, many of these benchmarks are now saturated [8, 21, 23, 27]. They now offer only limited differentiation among the newest LLMS models and thus limited insights for the next step in development. Moreover, the mathematical questions in datasets such as the SAT and AIME are drawn from grade-school or competition-level problems, with limited connection to real-world decision-making contexts.

In contrast, classical problems from the operations research (OR) domain—especially those involving network optimization—are linked to planning and resource allocation. In this work, we introduce the first benchmark that systematically evaluates LLMs using canonical network optimization problems. Our benchmark spans three classes of increasing complexity: (1) shortestpath routing without capacity constraints, (2) minimum-cost flow with capacity constraints, and (3) multi-commodity flow problems. Such problems test LLM's abilities to both follow structured instructions and conforming to complex constraints [14, 15] along with long-term reasoning [16, 17] on complex tasks.

These problems are artificially created by us, allowing us to control complexity factors such as graph size, complexity, and flow demands. Such features allow us to evaluate how well LLMs reason over structured, multi-step optimization tasks.

DOI : 10.5121/oraj.2025.12201

From these questions we present results from 7 of the most recent LLMs from OpenAI [23-25], Google [21-22] and DeepSeek [26-27], revealing key differences in their reasoning and planning capabilities. Additionally our findings highlight gaps in current model capabilities and suggest promising directions for both LLM development and their potential viability for use with realworld OR tasks.

The primary contributions of this paper are as follows:

1. **A Benchmark for Network Flow Optimization Tasks:** We introduce the first benchmark tailored to network flow optimization problems. This includes a suite of evaluation datasets based on classical combinatorial optimization formulations, along with a set of diagnostic metrics designed to probe model reasoning, constraint handling, and instruction-following capabilities.

2. **Comprehensive Evaluation of state-of-the-art LLMs**: We evaluate the performance of seven of the most recent large language models (LLMs) with both text-based and "thinking" [18. 19, 20] modes, which perform internal planning where the model is prompted to reason step-by-step before producing a final answer. We provide a comparative analysis across models and settings. For instance, models in the GPT series exhibit a tendency toward over-optimism in cost estimation and constraint satisfaction, while models in the Gemini series display a more conservative, sometimes overly pessimistic approach to feasibility and cost trade-offs.

By introducing this benchmark, we aim to bridge the OR and Generative AI communities—demonstrating how classical OR problems can serve not only as powerful evaluation tools but also as inspiration for future LLM advancements in optimization and decision-making.

2. BACKGROUND –NETWORK OPTIMIZATION PROBLEMS

Network flow optimization is a classical problem in operations research and computer science, concerned with determining the most efficient way to route flow through a network subject to capacity, demand, and cost constraints. It forms the foundation for many real-world applications, including transportation, communication, and supply chain planning [9, 10, 11, 12]. In this paper, we use three canonical problems—described in detail below—as the basis for curating our benchmark dataset.

2.1. Shortest Path Problem

Given a directed graph $G = (V, E)$ with a cost function $c{:}E \rightarrow \mathbb{R}_{\geq 0}$ and two nodes $s, t \in V$, the shortest path problem seeks a path P from s to t that minimizes the total cost:

$$\min_{P \in \mathcal{P}_{st}} \sum_{(i,j) \in P} c_{ij}$$

where $P_{\{st\}}$ denotes the set of all simple paths from s to t in G.

2.2. Minimum-Cost Flow Problem (Single Commodity)

In the minimum-cost flow problem with capacity constraints, we are given a directed graph$G = (V, E)$ with:

- edge capacities $u_{ij} \geq 0$ and costs c_{ij} for each $(i, j) \in E$
- a source node $s \in V$ and a sink node $t \in V$
- a required flow demand $d > 0$ from s to t

The objective is to find a flow $f: E \to R^{\geq 0}$ that satisfies:

$$\min_f \sum_{(i,j)\in E} c_{ij} f_{ij}$$

subject to:

$$\sum_{j:(i,j)\in E} f_{ij} - \sum_{j:(j,i)\in E} f_{ji} = \begin{cases} d, & \text{if } i = s \\ -d, & \text{if } i = t \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in V$$

$$0 \leq f_{ij} \leq u_{ij} \quad \forall (i,j) \in E$$

Note that we only consider the problem which allows splitable flows, where the flow is allowed to be split across multiple paths from s to t.

2.3. Multicommodity Minimum-Cost Flow Problem

The multicommodity flow problem generalizes the single-commodity case by considering multiple source-destination pairs (commodities). Let K denote the set of commodities. For each $k \in K$, let (s_k, t_k) be the source and sink, and $d_k > 0$ be the flow demand. Each commodity k has a flow $f^k: E \to \mathbb{R}_{(\geq 0)}$.

The goal is to minimize the total cost across all commodities:

$$\min_{\{f^k\}} \sum_{k\in K} \sum_{(i,j)\in E} c_{ij} f_{ij}^k$$

subject to, for all $k \in K$:

$$\sum_{j:(i,j)\in E} f_{ij}^k - \sum_{j:(j,i)\in E} f_{ji}^k = \begin{cases} d_k, & \text{if } i = s_k \\ -d_k, & \text{if } i = t_k \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in V$$

$$\sum_{k\in K} f_{ij}^k \leq u_{ij} \qquad \forall (i,j) \in E$$

$$f_{ij}^k \geq 0 \qquad \forall (i,j) \in E, \forall k \in K$$

As in the single-commodity case, flows are allowed to be split among multiple paths. The shared edge capacities across commodities introduce coupling constraints that make the problem significantly more challenging.

3. BENCHMARK CONSTRUCTION

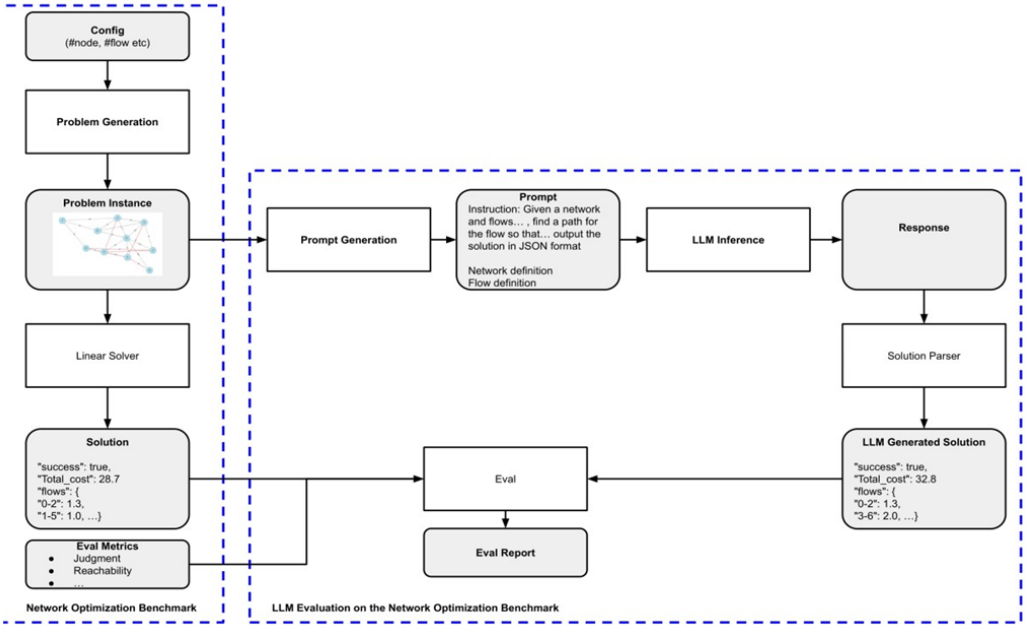

Figure 1: Overall Benchmark Curation and Evaluation Process

As illustrated in Figure 1, our work consists of two phases. The first phase -- "Network Optimization Benchmark" involves data curation, where we artificially generate evaluation datasets based on classical network optimization problems. It consists of two major blocks – Problem Generation and Linear Solver. The Problem Generation block takes a configuration as input and produces random problem instances. The Linear Solver block takes the problem instance and produces a solution. The benchmark also includes a set of metrics for evaluating the solution of the network optimization problem. The second phase -- "LLM evaluation on the Network Optimization Benchmark" uses this benchmark to evaluate the performance of LLMs. It has four major steps: 1) Prompt generation which converts the structured graph problem instance into a prompt for LLM; 2) executing the prompt against the LLM and produces the response; 3) parsing the solution from the LLM response; 4) evaluating the generated solution against the ground truth solution based on the evaluation metrics from the benchmark. We describe the first phase in detail in the current section. The second phase is described in the next section.

3.1. Problem Generation

As shown in Figure 1, given a problem configuration—including the number of network nodes, fan-in and fan-out, edge costs, bandwidth, and flow demands—the problem generation component randomly creates a network and a set of flows that need to be routed. For the multicommodity flow problem, the configuration also specifies the number of flows.

3.2. LP Solver

The solution to each problem instance is obtained using a linear programming (LP) solver [13]. The solver takes as input the network topology and flow specifications, formulates the corresponding LP based on flow conservation and capacity constraints, and outputs the optimal

routing configuration that minimizes total cost as solution, if there exists a feasible solution; or outputs as infeasible.

3.3. Evaluation Dataset Statistics

3.3.1. Shortest Path Problem

For the shortest path problem, we construct evaluation datasets with varying graph sizes, organized into four slices. Each slice corresponds to a specific graph size, ranging from 10 to 40 nodes in increments of 10. For each graph size, the dataset contains 50 problem instances generated on random directed graphs with fan-in and fan-out and a single source-destination pair. The column “Edges per graph” in Table 1 gives the maximum, average and minimum numbers of edges over the 50 problem instances in each slice. The column “Success / Failure” indicates if a successful path exists at all.

Table 1: Evaluation Dataset statistics for Shortest Path Problem

	Edges per graph [max, avg, min]	Success / Failure
10 nodes	33 / 25.62 / 18	48 / 2
20 nodes	62 / 50.02 / 41	46 / 4
30 nodes	84 / 75.82 / 64	47 / 3
40 nodes	119 / 100.8 / 89	45 / 5

The flow length—defined as the number of hops in the shortest path between the source and destination—is used as a key metric to characterize the problem difficulty. As the number of nodes in the graph increases, both the average and maximum flow lengths also increase, reflecting the greater complexity of navigating larger graphs. The distribution of flow lengths for each slice is shown in the histograms in Figure 2.

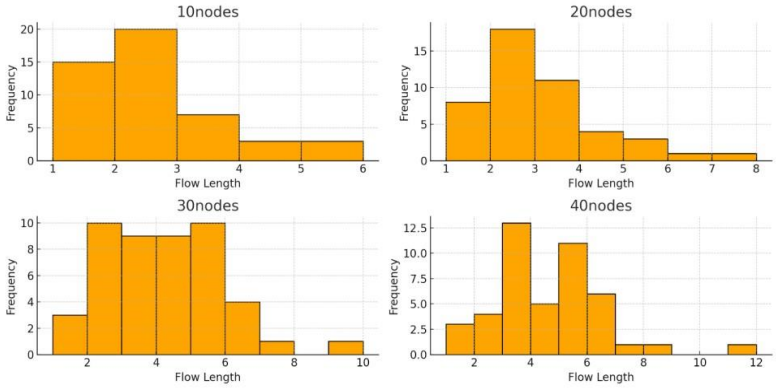

Figure 2: Flow Length Distribution for the Shortest Path Problem

A successful instance is defined as one in which a valid shortest path between the source and destination can be identified. A failure instance occurs when no such path is found. The success and failure counts for each slice of the dataset are summarized in Table 1 under the "Success/Failure" column. Overall, the success rate remains high across different graph sizes with a slight decline as graph complexity increases.

3.3.2. Minimum-Cost Flow Problem with Capacity Constraints

For the minimum-cost flow problem with capacity constraints, we construct evaluation datasets with varying graph sizes. Similar to the shortest-path problem case, we organized it into four slices corresponding to a specific graph size, ranging from 10 to 40 nodes in increments of 10 as shown in Table 2. For each graph size, the dataset contains 50 problem instances generated on random directed graphs with edge cost and bandwidth constraints and a single source-destination pair.

The column “Edges per graph” in Table 2 gives the maximum, average and minimum numbers of edges over the 50 problem instances in each slice.

Table 2: Evaluation Dataset statistics for the Min-Cost Flow Problem

	Edges per graph [max, avg, min]	Success / Failures
10 nodes	33 / 25.62 / 18	38 / 12
20 nodes	62 / 50.02 / 41	41 / 9
30 nodes	84 / 75.82 / 64	38 / 12
40 nodes	119 / 100.8 / 89	39 / 11

The flow length—defined as the length of the minimum-cost path that satisfies flow conservation constraints between the source and destination. The distribution of flow lengths for each slice is shown in the histograms in Figure 3. As the number of nodes in the graph increases, both the average and maximum flow lengths also increase. Compared to the shortest path problem, flow lengths in this setting tend to be longer, due to the additional constraints on flow conservation and the influence of edge costs. This highlights the increased complexity of the minimum-cost flow problem.

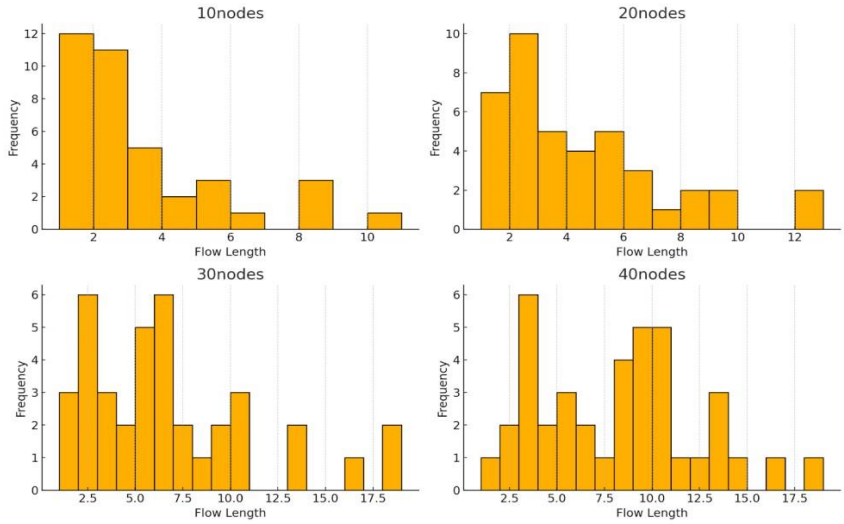

Figure 3: Flow Length Distribution for the Min-Cost Flow Problem

A successful instance is defined as one in which the LP solver identifies a valid path between the source and destination that satisfies all problem constraints, including flow conservation and edge

capacity limits. A failure instance occurs when no such feasible solution is found. The success and failure counts for each slice of the dataset are summarized in Table 2 under the "Success/Failure" column. Overall, the success rate is lower compared to the shortest path problem, reflecting the increased complexity of the minimum-cost flow formulation.

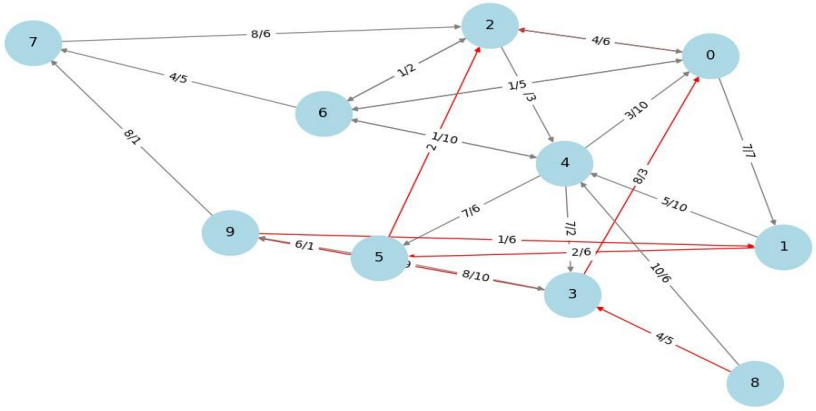

Figure 4: A Sample Min-Cost Flow Problem in a 10-node Graph

Figure 4 visualizes a sample min-cost flow problem where node 8 is the source, and node 2 is the destination. Optimal paths (8, 3, 9, 1, 5, 2) and (8, 3, 0, 2) are marked in red.

3.3.3. Multicommodity Minimum-Cost Flow Problem

For the Multicommodity Minimum-Cost Flow Problem, we construct evaluation datasets with varying graph sizes, similar to the shortest-path problem and mini-cost flow cases. For each problem instance generated on random directed graphs with edge cost and bandwidth constraints and multiple source-destination pairs.

The column “Commodity number” in Table 3 the number of flows for each slice and the success and failure cases over the 50 problem instances in each slice.

Table 3: Evaluation Dataset statistics for Multicommodity Min-Cost Flow Problem

	Commodity number	Success / Failures
10 nodes	2	27 /23
20 nodes	4	20 / 30
30 nodes	6	15 / 35
40 nodes	8	6 / 44

Figure 5 illustrates the flow length—defined as the length of the minimum-cost path that satisfies flow conservation constraints between the source and destination for all commodities. Similarly, as the number of nodes in the graph increases, both the average and maximum flow lengths also increase.

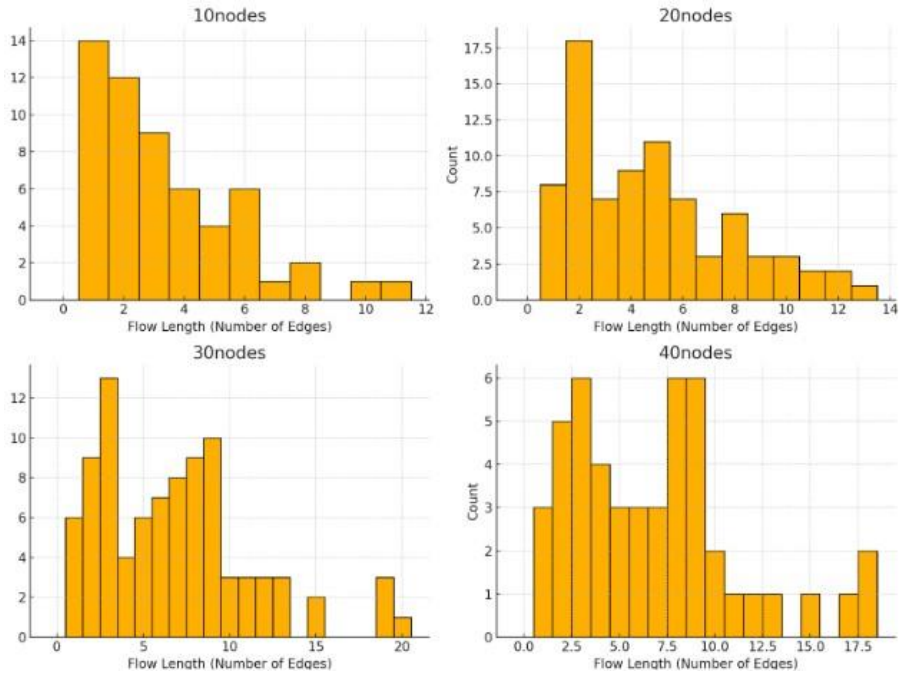

Figure 5: Flow Length Distribution for the Multicommodity Min-Cost Flow Problem

4. EVALUATION RESULTS AND ANALYSIS

In this section, we report the evaluation results for the following 7 state-of-art models from both proprietary commercial models and open-source communities:

- **Gemini 2.0 Flash (Google) [21]:** A lightweight multimodal AI model optimized for speed and efficiency across various tasks, capable of processing text, images, video, and audio inputs
- **Gemini 2.0 Flash (Thinking Model) [22]:** An experimental variant of Gemini 2.0 Flash, fine-tuned for enhanced reasoning capabilities through multi-step problem-solving approaches.
- **GPT-4o (OpenAI) [23]:** OpenAI's multimodal model capable of real-time processing and generation of text, images, and audio, demonstrating strong performance across various benchmarks.
- **GPT-4.5 (OpenAI) [24]:** An enhanced iteration of GPT-4o, OpenAI's largest and best model for chat. GPT-4.5 is a step forward in scaling up pre-training and post-training. GPT-4.5 improves its ability to recognize patterns, draw connections, and generate creative insights.
- **O3-mini (OpenAI) [25]: A** powerful and fast model advances the boundaries of what small models can achieve, delivering exceptional STEM capabilities—with particular strength in science, math, and coding.
- **DeepSeek v3 [26]:** An advanced OSS Mixture-of-Experts (MoE) language model with 671B total parameters, known for its strong performance among all OSS models.
- **DeepSeek R1 [27]:** A reasoning-focused language model developed through reinforcement learning, demonstrating the strongest capabilities in math and problemsolving among all OSS models

4.1. Model Inference

Table 4 shows the input and expected output formats of LLM inference. The left two columns (graph definition and flows definition) constitute the meaty part of the prompt. The prompt also explicitly mentions the nature of the problem (e.g. shortest path problem), also outlines the expected JSON format. As outlined in the rightmost column, the LLM is expected to answer, in the success key, whether a feasible solution can be found. If yes, it needs to present all details of the found path(s) including the full paths, amount of flow through each path, as well as the total cost.

Table 4: Input and Output JSON Formats for Model Inference

Graph definition (LLM input)	**Flows definition (LLM input)**	**Solution (expected LLM output)**

{
"directed": true,
"multigraph": false,
"graph": {},
"nodes": [
{
"id": "0"
},
{
"id": "1" },...
	[
	{
	"src": 8,
	"dest": 2,
	"demand": 2.3412198276810834
	}
	]
		{
		"success": true, "total_cost": 28.777077587535167,
		"flows": {
		"0-2": 1.3412198276810834,
		"1-5": 1.0,
		"3-0": 1.3412198276810834,
		"3-9": 1.0,
		"5-2": 1.0,
		"8-3": 2.3412198276810834,

],
"links": [
{
"source": "0",
"target": "2",
"cost": 2,
"capacity": 7
},
{
"source": "0",
"target": "1",
"cost": 7,
"capacity": 7 },...
]
}
		"9-1": 1.0
		}
		}
		
		or
		
		{
		"success": false,
		}

4.2. Evaluation Results of Shortest Path Problem

We now show evaluation results on the shortest path problem. We explain the metric in each column as follows.

- **Judgment:** This is a binary metric. The shortest path problem is feasible if at least one path exists from source to destination. Otherwise, it is infeasible. First, an LLM needs to arrive at the correct judgment regarding feasibility. Its judgement is wrong if it determines a feasible problem to be infeasible, or vice versa.

- **Reachability:** This binary metric examines the path-finding ability of LLM: whether it can concatenate edges into a path which is truly reachable, connecting source and destination.
- **Cost Match**: This binary metric examines the aggregation ability of LLM: whether it can add edge cost along the found path to the correct sum.
- **Zero Violation**: This binary metric is true if the previous three metrics are true. This is a gating metric to the final optimality metric, since the optimal solution must be a sound solution whose cost is correctly calculated.
- **Optimality Ratio**: We apply this metric only to LLM answers with zero violation. Only these answers have the potential to reach optimality. We normalize the total cost provided by LLM against the total cost derived from the LP solver. If the ratio is 1, then the LLM solution is optimal.

Tables below show average results per LLM and graph size. Since all metrics except optimality ratio are binary, the maximum value is 1, meaning all solutions succeeded on the metric. Likewise, 0 is the minimum value, meaning all solutions failed. 0.5 means half the solutions succeeded.

We have the following observations:

1. All models perform strongly in terms of constraint observation, reflected by perfect or near-perfect judgment and reachability scores.
2. The cost match is a more challenging task, requiring addition of multiple integers. Furthermore, the performance deteriorates as the graph size increases, because the found path also gets longer.
3. If the found paths pass the scrutiny of all binary metrics, the paths are also optimal, i.e. the shortest path.
4. On the shortest path task, regular LLMs are equally competitive as the reasoning model (o3-mini).

Table 5: Evaluation Results of Shortest Path Problem

(a) Gemini Flash 2.0

Category	Judgment	Reachability	Cost match	Zero violation	Optimality ratio
10 Nodes	1	1	0.5	0.5	1
20 Nodes	1	1	0.44	0.44	1
30 Nodes	1	1	0.12	0.12	1
40 Nodes	1	1	0.1	0.1	1

(b) GPT-4o

Category	Judgmen	Reachability	Cost match	Zero violation	Optimality ratio
10 Nodes	1	1	0.52	0.52	1
20 Nodes	0.96	0.98	0.36	0.36	1
30 Nodes	0.94	1	0.08	0.08	1
40 Nodes	0.94	1	0.08	0.08	1

(c) o3-mini

Category	Judgment	Reachability	Cost match	Zero violation	Optimality ratio
10 Nodes	1	1	0.54	0.54	1
20 Nodes	1	1	0.34	0.34	1
30 Nodes	0.96	1	0.08	0.08	1
40 Nodes	0.94	1	0.08	0.08	1

(d) DeepSeek v3

Category	Judgment	Reachability	Cost match	Zero violation	Optimality ratio
10 Nodes	1	1	0.52	0.52	1
20 Nodes	1	1	0.32	0.32	1
30 Nodes	0.96	1	0.1	0.1	1
40 Nodes	0.94	1	0.1	0.1	1

4.3. Evaluation Results of MinCost Flow Problem

We now show evaluation results on the MinCost Flow problem. Besides metrics already introduced in the previous problem, there are two more metrics.

- **Flow consistency:** This is also a binary metric observing the flow conservation principle. All intermediate nodes on a path must have the same amount of inbound and outbound flow. Also the outbound flow from the source node, as well as the inbound flow into the destination node, must equal to the flow demand.
- **Capacity Violation:** This binary metric ensures that aggregate flows passing on any edge must not exceed its capacity.

The rule to calculate optimality ratio stays the same: we only apply it to the case of zero violation, i.e. all previous five metrics are true. If the optimality ratio is 1, it means LLM found the optimal solution, i.e. minimum cost path to route the demanded flow. If the ratio is greater than 1, the solution is still sound but not the most efficient.

Unlike the shortest path problem where costs are all integers, the flow demands in this problem are decimal numbers, causing the final cost (weighted by flow) to be decimal numbers too. This poses challenges to not only LLMs, but our evaluation. Since the exact match between two decimal numbers is infeasible, we set the match precision level to be **1e-4**.

As seen in the performance tables below, this problem is considerably harder than the shortest path problem.

1. As the graph size grows, all models demonstrate deteriorating performance. Most notably, the success rate of zero violation (sound solution) is low even for the smallest graph, and continues to decline with few exceptions.
2. Reasoning models distinguish themselves from other LLMs, demonstrated by the nearperfect judgment scores and high success rate of zero violation by o3-mini and DeepSeek R1.

Table 6: Evaluation Results of MinCost Flow Problem

(a) Gemini Flash 2.0

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.82	0.9	0.94	0.24	0.68	0.24	1
20 Nodes	0.9	0.84	0.88	0.18	0.7	0.18	1.25
30 Nodes	0.78	0.74	0.86	0.12	0.56	0.12	1.37
40 Nodes	0.78	0.64	0.76	0.04	0.44	0.04	1

(b) Gemini Flash 2.0 Thinking

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.82	0.7	0.94	0.22	0.84	0.2	1.20833333
20 Nodes	0.82	0.78	0.96	0.24	0.8	0.24	1.04770677
30 Nodes	0.74	0.68	0.98	0.1	0.74	0.08	1.09687437
40 Nodes	0.78	0.54	1	0.02	0.7	0	

(c) GPT-4o

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.66	0.42	0.5	0.12	0.44	0.12	1.02323617
20 Nodes	0.4	0.22	0.22	0.06	0.2	0.06	1
30 Nodes	0.28	0.02	0.04	0	0.04	0	
40 Nodes	0.22	0	0	0	0	0	

(d) GPT 4.5

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.86	0.86	0.9	0.28	0.86	0.28	1.00995836
20 Nodes	0.86	0.82	0.96	0.14	0.9	0.14	1
30 Nodes	0.76	0.82	1	0.06	0.84	0.06	1
40 Nodes	0.78	0.84	1	0.02	0.8	0.02	1

(e) o3-mini

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	1	0.76	0.76	0.68	0.76	0.68	1.00624364
20 Nodes	1	0.82	0.82	0.78	0.82	0.78	1.00350982
30 Nodes	0.98	0.74	0.74	0.64	0.74	0.64	1.03402684
40 Nodes	0.98	0.76	0.76	0.6	0.76	0.6	1.02917345

(f) DeepSeek v3

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.86	0.6	0.66	0.26	0.5	0.18	1.0154907
20 Nodes	0.58	0.42	0.44	0.14	0.36	0.14	1
30 Nodes	0.56	0.4	0.4	0.08	0.22	0.06	1
40 Nodes	0.48	0.22	0.3	0.04	0.12	0.02	1

(g) DeepSeek R1

Category	Judgment	Reachabilit y	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	1	0.76	0.76	0.62	0.74	0.62	1.00449078
20 Nodes	1	0.8	0.82	0.66	0.82	0.66	1.00572502
30 Nodes	1	0.74	0.76	0.5	0.74	0.5	1.02530451
40 Nodes	1	0.68	0.78	0.4	0.78	0.4	1.03064951

4.4. Evaluation Results of Multicommodity MinCost Flow Problem

We now show evaluation results on the Multicommodity MinCost problem. No new metrics are introduced. This problem is the hardest of all.

Consistent with the observation in the MinCost problem, reasoning models distinguish themselves from normal LLMs. When none of the normal LLMs finds a single sound solution with zero violations, reasoning models, especially o3-mini, continue to deliver near-perfect judgment scores and very close optimality ratio over the sound solutions they found.

Also worth noting is reasoning models' adaptability when the scale of the problems grows with more constraints and a bigger graph. When we relax the match precision level (from 1e-4 to 1e2) on the flow consistency, cost match, and capacity constraint, more sound solutions are found. Closer investigations to these newly emerged sound solutions showed that the models found the correct optimal paths, but proactively rounded down the decimal places during the cost calculation.

(a) Gemini Flash 2.0

Category	Judgment	Reachabilit y	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.56	0.8	0	0.04	0.54	0	
20 Nodes	0.4	0.34	0.08	0.12	0.18	0	
30 Nodes	0.3	0.8	0.18	0.18	0.02	0	
40 Nodes	0.18	0.6	0.06	0.06	**0**		

(b) Gemini Flash 2.0 Thinking

Category	Judgmen t	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.56	0.4	0.34	0.04	0.42	0	
20 Nodes	0.4	0.14	0.14	0.08	0.2	0	
30 Nodes	0.38	0.02	0.02	0.02	0.04	0	
40 Nodes	0.14	0	0.02	0.02	**0**		

(c) GPT-4o

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.48	0.02	0	0	0	0	
20 Nodes	0.6	0	0	0	0	0	
30 Nodes	0.7	0	0	0	0	0	
40 Nodes	0.88	0	0	0	0	0	

(d) o3-mini (precision level 1e-2)

Category	Judgment	Reachabilit y	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	1	0.56	0.56	0.56	0.56	0.56	1.0161346
20 Nodes	0.98	0.38	0.38	0.38	0.38	0.38	1.0299260
30 Nodes	1	0.3	0.28	0.22	0.28	0.22	1.0856017
40 Nodes	0.94	0.14	0.1	0.1	0.06	0.06	1.0910866

(e) o3-mini

Category	Judgment	Reachabilit y	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	1	0.56	0.56	0.54	0.56	0.54	1.0167301
20 Nodes	0.98	0.38	0.38	0.2	0.38	0.2	1.0259425
30 Nodes	1	0.3	0.26	0.04	0.24	0.04	1.0375095
40 Nodes	0.94	0.14	0.1	0	0.06		

(f) DeepSeek v3

Category	Judgment	Reachabilit y	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.68	0.26	0.26	0	0.18	0	
20 Nodes	0.6	0	0	0	0	0	
30 Nodes	0.7	0	0	0	0	0	
40 Nodes	0.88	0	0	0	0	0	

(g) DeepSeek R1 (precision level 1e-2)

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.96	0.54	0.54	0.48	0.54	0.48	1.0154247
20 Nodes	0.94	0.32	0.24	0.14	0.3	0.14	1.0435892
30 Nodes	0.76	0.04	0	0	0.02	0	
40 Nodes	0.9	0	0.02	0.02	0	0	

(h) DeepSeek R1

Category	Judgment	Reachability	Flow consistency	Cost match	Capacity constraint	Zero violation	Optimality ratio
10 Nodes	0.96	0.54	0.42	0.34	0.54	0.26	1.0035080
20 Nodes	0.94	0.32	0.16	0.12	0.3	0.08	1.0441572
30 Nodes	0.76	0.04	0	0	0.02	0	
40 Nodes	0.9	0	0.02	0.02	0	0	

5. CONCLUSIONS

This paper introduces the first benchmark that systematically evaluates large language models on classical network optimization problems from operations research. Unlike existing benchmarks focused on academic or competition-style questions, our benchmark challenges LLMs with structured instructions, hard constraints, and long-horizon reasoning tasks grounded in real-world decision-making. By synthetically generating problem instances with controllable complexity, we uncover meaningful differences in the reasoning and planning behaviors of seven state-of-the-art LLMs.

Ultimately, this work bridges the gap between the operations research and LLM communities, positioning network optimization not only as a rigorous evaluation framework but also as a fertile ground for advancing the reasoning capabilities of future models. We hope this benchmark serves as a stepping stone toward more robust, interpretable, and practical applications of LLMs in decision-making domains. Finally, we acknowledge that our study relies on synthetic data, which serves as a controlled environment for initial validation. Future work will focus on applying the method to real-world scenarios to assess its practical impact.

ACKNOWLEDGEMENTS

I would like to thank Professor Mike Murphy for teaching the Networks course at Foothill College. Learning about computer network systems is what inspired me to write this paper.

REFERENCES

[1] K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, “Training verifiers to solve math word problems,” *arXiv preprint arXiv:2110.14168*, 2021.

[2] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, “Measuring mathematical problem solving with the MATH dataset,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 3512–3524, 2021.

[3] S. Patel, C. Raffel, and A. Raffel, “Are NLP models really able to solve simple math word problems?” *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3983–3995.
[4] Azerbayev, Z., et al., "Llemma: An Open Language Model for Mathematics," arXiv preprint arXiv:2310.10631, 2023. [Online]. Available: https://arxiv.org/abs/2310.10631
[5] Lewkowycz, A., et al., "Solving Quantitative Reasoning Problems with Language Models," Advances in Neural Information Processing Systems, vol. 35, pp. 29363–29377, 2022. [Online]. Available: https://arxiv.org/abs/2206.14858
[6] Hendrycks, D., et al., "Measuring Massive Multitask Language Understanding," arXiv preprint arXiv:2009.03300, 2020. [Online]. Available: https://arxiv.org/abs/2009.03300
[7] M. Jia, "AIME 2024 Dataset," Hugging Face, Mar. 2025. [Online]. Available: https://huggingface.co/datasets/Maxwell-Jia/AIME_2024
[8] Z. Shao et al., "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models," arXiv preprint arXiv:2402.03300, Feb. 2024. [Online]. Available: https://arxiv.org/abs/2402.03300
[9] D. P. Bertsekas and R. G. Gallager, Data Networks, 2nd ed. Upper Saddle River, NJ, USA: PrenticeHall, 1992.
[10] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.
[11] D. Bertsekas, Network Optimization: Continuous and Discrete Models. Belmont, MA, USA: Athena Scientific, 1998.
[12] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, Linear Programming and Network Flows, 4th ed. Hoboken, NJ, USA: Wiley, 2009.
[13] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.
[14] R. Lou, K. Zhang, and W. Yin, "Large Language Model Instruction Following: A Survey of Progresses and Challenges," arXiv preprint arXiv:2303.10475, Mar. 2023.
[15] J. Zhou et al., "Instruction-Following Evaluation for Large Language Models," arXiv preprint arXiv:2311.07911, Nov. 2023. [Online]. Available: https://arxiv.org/abs/2311.07911.
[16] J. Wei et al., “Chain of Thought Prompting Elicits Reasoning in Large Language Models,” arXiv preprint arXiv:2201.11903, Jan. 2022. [Online]. Available: https://arxiv.org/abs/2201.11903[17] Y. Fu, X. Guo, L. Zheng, M. Xu, and M. Wang, “Complexity-Based Prompting for Multi-Step Reasoning,” in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), Singapore, Dec. 2023. [Online]. Available: https://arxiv.org/abs/2305.12255
[18] X. Tian et al., "Think Twice: Enhancing LLM Reasoning by Scaling Multi-round Test-time Thinking," arXiv preprint arXiv:2503.19855, Mar. 2025. [Online]. Available: https://arxiv.org/abs/2503.19855
[19] K. Kudo et al., "Think-to-Talk or Talk-to-Think? When LLMs Come Up with an Answer in MultiStep Reasoning," arXiv preprint arXiv:2412.01113, Dec. 2024. [Online]. Available: https://arxiv.org/abs/2412.01113
[20] J. Sun et al., "Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph," arXiv preprint arXiv:2307.07697, Jul. 2023. [Online]. Available: https://arxiv.org/abs/2307.07697
[21] Google DeepMind, "Google introduces Gemini 2.0: A new AI model for the agentic era," Google Blog, [Online]. Available: https://blog.google/technology/google-deepmind/google-gemini-ai-updatedecember-2024/
[22] Google DeepMind, [Online]. “Gemini 2.0 Flash Thinking Experimental” Available: https://deepmind.google/technologies/gemini/flash-thinking/
[23] OpenAI, "GPT-4o: OpenAI’s New Omnimodel," OpenAI Blog, [Online]. Available: https://openai.com/index/hello-gpt-4o/
[24] OpenAI, "Introducing GPT-4.5," OpenAI Blog, [Online]. Available: https://openai.com/index/introducing-gpt-4-5/
[25] OpenAI, "OpenAI o3-mini," OpenAI Blog, [Online]. Available: https://openai.com/index/openai-o3mini/
[26] DeepSeek-AI et al., "DeepSeek-V3 Technical Report," arXiv preprint arXiv:2412.19437, Dec. 2024. [Online]. Available: https://arxiv.org/abs/2412.19437

[27] DeepSeek-AI et al., "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," arXiv preprint arXiv:2501.12948, Jan. 2025. [Online]. Available: https://arxiv.org/abs/2501.12948.