Visualization Biases MLLM's Decision Making in Network Data Tasks

Timo Brand*,† Henry Förster*,† Stephen G. Kobourov* Jacob Miller*

Technical University of Munich, Heilbronn, Germany

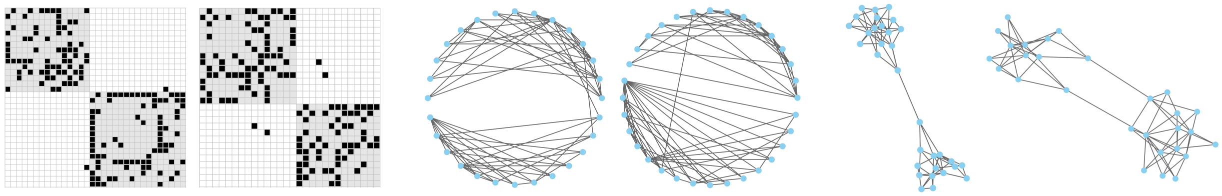

Figure 1: Visualizations for two example graphs with a bridge and a 2-edge-connected example graph, using (from left to right) a pixel-based visualization of the adjacency matrix `AM`, a circular layout `Circular`, and a spring layout `Spring`. Each block of two visualizations contains a `positive` example of a graph with a bridge and a `negative` example of a graph that is 2-edge connected. Can you tell which one is which?

Abstract

We evaluate how visualizations can influence the judgment of MLLMs about the presence or absence of bridges in a network. We show that the inclusion of visualization improves confidence over a structured text-based input that could theoretically be helpful for answering the question. On the other hand, we observe that standard visualization techniques create a strong bias towards accepting or refuting the presence of a bridge – independently of whether or not a bridge actually exists in the network. While our results indicate that the inclusion of visualization techniques can effectively influence the MLLM's judgment without compromising its self-reported confidence, they also imply that practitioners must be careful of allowing users to include visualizations in generative AI applications so as to avoid undesired hallucinations.

Index Terms: network visualization, MLLM, bias, bridge, visualization mirage, visual proof

1 Introduction

With recent developments in generative AI, *large language models* (LLMs) are increasingly used as decision makers in practice. Their nascent applications span a wide variety of domains, e.g., law [7], finance [8], and healthcare [16]. LLMs are now also beginning to be able to process multi-modal input. In this context, it has been verified that multi-modal large language models (MLLMs) possess some visualization literacy [3, 6]. Hence, MLLM decision makers might benefit from visualizations being provided in addition to the raw data, akin to how human decision makers use visual analytics [12] to support their decisions. Another noteworthy aspect is that human users interacting with MLLM decision makers could attempt to influence the MLLM's decision making process by augmenting data with visualizations, which may or may not be desired.

We see the need for assessing how MLLMs' decision-making processes can be guided by providing helpful visualizations. In a preliminary study, Förster et al. [5, supplemental material] asked an MLLM whether a network contained a Hamiltonian cycle. The confidence of the MLLM's response could be improved when providing a *visual certificate*, that is, a visualization highlighting the Hamiltonian cycle, compared to providing an adjacency matrix representation of the network as input. This provides initial support for the hypothesis that visualization actually helps MLLMs in deriving correct solutions for tasks related to network data. However, given their tendency to "hallucinate", MLLMs might be prone to *visualization mirages* [9]; visualizations whose initial reading might support an erroneous hypothesis that is invalidated upon closer inspection. In particular, the choice of visualization style might already create a bias. We consider the following questions:

RQ1 *Can the accuracy and confidence of MLLMs analyzing network data be improved when a suitable visualization is provided as part of the input?*

RQ2 *Does the inclusion of visualization create a bias in the decision-making process of MLLMs and, if so, is such a bias dependent on the visualization style used?*

We focus on answering RQ1 and RQ2 for a specific task – determining if a network contains a *bridge*; a single edge whose removal separates the network. If a network has no bridge, it is also called *2-edge-connected*, as at least two edges must be removed before it becomes separated. In our experiments, we let a MLLM determine if a network contains a bridge and record the correctness and self-reported confidence. We use standard visualization techniques, which we believe to be most likely adopted by MLLM users.

Existing network visualization techniques are designed for humans, and it is unclear how much of their design principles apply to MLLMs. Typically network visualizations focus on supporting overview tasks, displaying the entire data in an aesthetic and readable fashion [1, 13, 15]. Such visualizations display ground-truth structural properties of the underlying network faithfully [11] facilitating free-form exploration by users. They can be sufficient in supporting narratives in media by providing select views of the data [4]. Adjacency matrices, circular layouts, and (force-directed) node-link diagrams all adhere to Munzner's expressiveness and effectiveness principles [10], i.e., they show *all and only* the data while making the representation *effective* for the required task, e.g., identifying clusters, path-following, or identifying bridges; see Fig. 1.

2 Experimental Setup

In each trial, we present to one of the MLLMs, (`GPT`[1] or `Qwen`[2]), a small to medium-sized network described in text-form by an ad-

*E-mail: firstname.lastname@tum.de

†These authors contributed equally.

[1] https://platform.openai.com/docs/models/gpt-4o

[2] https://www.alibabacloud.com/help/en/model-studio/what-is-qwen-llm

jacency matrix and ask it whether the network contains a bridge. Each network is composed of two components C_1 and C_2 with either one or two links connecting them. Hence, these edge(s) connecting C_1 and C_2 are exactly the links that may be the bridge in the network. Our independent variable is the additional information given along with the network representation: We consider two text-based inputs, namely an unstructured adjacency matrix (i.e., a random permutation of the rows and columns) and a structured adjacency matrix (i.e., a permutation of the rows and columns so that all nodes of C_1 precede all nodes of C_2). In addition, we consider the setting where we provide a visualization in addition to the unstructured adjacency matrix, which can be a pixel-based visualization of the structured matrix or a standard node-link visualization (circular or force-directed layout).

2.1 Generation of Stimuli

Networks. We generate a set of test-stimuli networks. For each network, we randomly sample two subcomponents C_1 and C_2 using the Barabasi-Albert model [2] implementation in the `python` library `NetworkX`. For the connectivity parameter, we use the value 3 and we re-generate each component until we obtain a 3-edge-connected one. We create two sizes of components, small and large. Small components have between 12 and 18 nodes, and large components have between 32 and 38 nodes, chosen uniformly at random.

We generate networks with a bridge by adding a single edge between a random node in C_1 and a random node in C_2. For networks without a bridge, we do the same and choose another different node in each component and add an edge between them. We say that the former ones are `positive` and the latter ones are `negative` instances. We generate 25 graphs for each of the 4 component size cases (C_1 small/large and C_2 small/large), with a bridge and without one; i.e., in total 100 `positive` and 100 `negative` instances.

Adjacency Information. We are interested in how providing a visualization of the graph impacts the performance of an MLLM in answering questions about graph properties. As such, we supply a text representation of the graph and a possible layout. As text representation, we chose adjacency matrices. That is, we write for each vertex a separate row which contains a 1 if the vertex with the id of that column is adjacent to the vertex with the id of the row, and a 0 otherwise. Before writing the adjacency matrix as text, we randomly permute the nodes to not leave any structure in the order of the rows and columns. This data is later passed alongside a visualization to the MLLM. Mainly to answer RQ1, we also experiment with two variants of text-only configuration:

- `Novis`: We provide only the text representation of the adjacency matrix.
- `Novis`*: We provide a structured textual adjacency matrix representation, i.e., the nodes of C_1 come before the nodes of C_2 in the row/column ordering. The order within each component is randomly chosen.

Visualizations In addition to the text-only configurations, we also experiment with three configurations that pass a visualization of the graph alongside the text of the permuted adjacency matrix.

We generate the following layouts; see also Fig. 1 (left to right):

- `AM`: We provide a pixel-based visualization of the adjacency matrix where we sort the rows/columns so that all nodes of C_1 appear before all nodes of C_2, i.e., of the `Novis`* configuration. Moreover, we enrich the visualization by shadings behind the square matrices representing C_1 and C_2.
- `Circular`: We provide a circular layout generated with the function `circular_layout` of `NetworkX`. We provide the network with a permutation of the nodes that separates the nodes of C_1 and C_2. As a result, the nodes of C_1 and C_2 are separated along the circle containing all the nodes, as in Fig. 1.
- `Spring`: We provide a force-directed layout generated using the `spring_layout` function of `NetworkX`.

2.2 Trial MLLM Prompts

Prompt Structure. We now describe how we performed a trial for the experiment. In each trial, we sent a prompt to the MLLM consisting of two parts, a *system message* sent in the `system` role, and a *trial instruction* sent in the `user` role. All experiments have been performed as zero-shot experiments at temperature 0.0. That way, we aim to evaluate the baseline answers that an MLLM would produce when being exposed to the visual stimuli. Moreover, we did not allow MLLMs to access any APIs as depending on the practical use-case this behavior may be unwanted. In particular, this effectively prevents MLLMs from executing code.

System Message. The system message explains the role of the MLLM using the following instructions:

```
You are an expert graph-theory assistant.
The user will provide two candidate statements (A and
B) about a graph. Exactly one of them is true.
Reply on ONE line in the form:
ANSWER: ⟨A/B⟩ | CONFIDENCE: ⟨1-5⟩
No extra text.
```

The system message tells the MLLM that it is expected to perform well in the following trials, instead of attempting to emulate an average user. Second, it conveys that exactly one of the two given options is true and that it should report just one of them.

Trial Instruction. The trial instruction consists of several parts. First, we formulate the question asked. There are two variants of the question (**Q1** and **Q2**), as we want to avoid a bias for the first or second option asked in the question:

```
Q1: Does the graph have an edge that, if removed,
would disconnect it?
Q2: Does the graph have no edge that, if removed,
would disconnect it?
```

We conclude the query with providing the possible answers:

```
Choose exactly one option: A) [OPT_A], B) [OPT_B]
Answer format: ANSWER: ⟨A/B⟩ | CONFIDENCE: ⟨1-5⟩
```

`[OPT_A]` and `[OPT_B]` are placeholders for the following two answer options **A1** and **A2** whose order we make interchangeable:

```
A1: The graph does have such an edge.
A2: The graph does not have such an edge.
```

We create a trial for each combination of question and order of answers for each visualization-network pair. After the query, we provide the adjacency matrix as text, following the word "`Adjacency:`". If the configuration includes a visualization, we append it as a PNG image generated with `NetworkX` and saved using the function `savefig()` setting the parameter `dpi` to 300. For each of the 200 stimuli and each combination of question and order of answers, we conduct a trial with each of the 5 configurations.

3 Experimental Results

We performed the experiments using the MLLMs `GPT` (`gpt-4.1-2025-04-14`) and `Qwen` (`qwen2.5-vl-72b-instruct`). For each trial, we recorded whether the MLLM's response was correct and the self-reported confidence in the range 1 to 5 provided as part of the MLLM's output. For each model, we compute the mean accuracy and Bonferroni-corrected confidence intervals over the set of configurations we are interested in. In the process, we apply the bootstrap statistical analysis method that takes a data collection and creates many thousands of simulated samples (of the same size as the original) by drawing from the original data collection with replacement [14]. For confidence scores, we record the proportion of responses for each reported confidence level.

More precisely, we make two comparisons: First, we compare the three adjacency matrix-based inputs `Novis`, `Novis`* and `AM`,

Table 1: Mean accuracy per model and configuration. The results are over all instances, only `positive`, and only `negative` instances.

Model	Novis			Novis*			AM			Circular			Spring		
	Total	Pos	Neg	Total	Pos	Neg	Total	Pos	Neg	Total	Pos	Neg	Total	Pos	Neg
GPT	0.480	0.240	0.720	0.499	0.325	0.673	0.500	0.778	0.223	0.503	0.123	0.883	0.526	1.000	0.053
Qwen	0.502	0.825	0.178	0.499	0.589	0.408	0.490	0.940	0.036	0.469	0.019	0.937	0.509	1.000	0.000

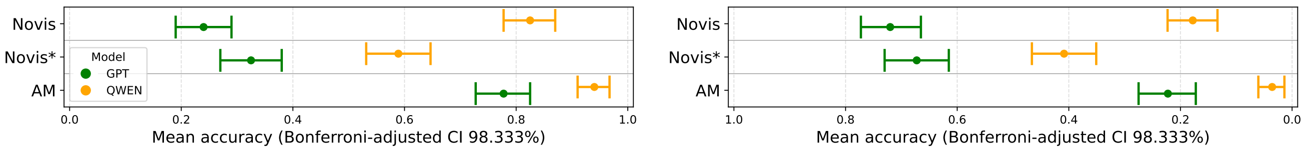

Figure 2: Confidence intervals for the results of the experiments using the adjacency matrix-based configurations `Novis`, `Novis`* and `AM`. (left) shows the results for positive instances and (right) the negative instances. Note the inverted *x*-axis on the right.

which differ in how structured and visual the information is provided. Second, we investigate how the three visual configurations `AM`, `Circular` and `Spring` perform in comparison to the base-line input `Novis`. Overall, the results are split over the `positive` and `negative` instances, to identify if any configuration causes a bias towards more often saying there is a bridge or there is none.

Accuracy. First observe that both models cannot effectively solve the task at hand, independent of the input configuration, with an overall mean accuracy (including both `positive` and `negative` instances) that is almost identical to a random coin flip; see Tab. 1. However, when evaluating `positive` and `negative` instances separately, we see substantial differences in the performances of the different configurations – all configurations perform well on either the `positive` or the `negative` instances, while they perform badly on the other set of instances. We now evaluate this effect further.

Considering the adjacent matrix-based methods `Novis`, `Novis`* and `AM` in Fig. 2, we observe that the MLLM's decision appears to be mainly dependent on the input configuration instead of whether or not the instance actually is `positive` or `negative`. This is evident from Fig. 2, where the accuracy is plotted increasing from left to right in the left subplot for `positive` instances and decreasing from right to left in the right subplot for `negative` instances – hence, in both figures, a datapoint on the left-hand side (0.0 for the left and 1.0 for the right subplot) indicates that the MLLM assessed the input to be 2-connected whereas a datapoint on the right-hand side (1.0 for the left and 0.0 for the right subplot) indicates that the MLLM assessed the input to contain a bridge. Since the means and confidence intervals in both subplots are almost identical, we actually visually see that the MLLM's decision appears to be mainly based on the type of its input but not on the underlying data.

More precisely, with `AM` both models determine the network to contain a bridge in the majority of cases, achieving high accuracy on the `positive` and low accuracy on the `negative` instances. Curiously, for the text-based inputs `Novis` and `Novis`*, there is a discrepancy in the responses provided by `GPT` and `Qwen`. Namely, given only a text-based adjacency matrix, `Qwen` appears to be biased towards deciding that a given network contains a bridge whereas for `GPT` the opposite is true. Hence, for `Novis` on the `positive` instances, `Qwen` has significantly higher accuracy than `GPT`, whereas on the `negative` instances `GPT` is more accurate than `Qwen`. When structuring the data in `Novis`*, we observe that both models become less stern in their decisions, with both models' accuracy getting closer to random guessing. For `Qwen` this effect is more pronounced and statistically significant.

The difference between the accuracy for configuration `AM` and the accuracy for the text-based inputs `Novis` and `Novis`* is statistically significant for both models. This is somewhat surprising as the configurations `Novis`* and `AM` essentially communicate the same ordered data, once as 0's and 1's in text form and once as either white or black pixels in the same matrix. Despite that, their accuracy is significantly different for both `GPT` and `Qwen`; for `Qwen` the effect size is even greater than in comparison to `Novis`.

Next, consider how the visualizations `AM`, `Circular` and `Spring` affect the accuracy compared to the baseline `Novis` of passing only the adjacency matrix as text; see Fig. 3 (note that for negative instances, the accuracy axis again has increasing values from right to left). Again, we observe a bias for both models to make the choice depending on the visualization style used. In particular, `AM` and `Spring` increase the probability for the MLLM to report that the network contains a bridge whereas `Circular` increases the probability to receive an answer indicating a 2-edge-connected network. Hence, for positive instances, `Spring` and `AM` achieve significantly higher accuracy than `Novis` whereas `Circular` performs worse than `Novis`. In contrast, for `negative` instances, with `Circular` significantly outperforming `Novis`, whereas `Spring` and `AM` perform significantly worse than the text-based representation `Novis`.

Based on Fig. 3 (bottom), we observe that for `GPT`, the bias created by `Spring` is significantly greater than the one for `Qwen`. In addition, `Qwen` achieves a higher divergence from `Novis` based on `Circular`, whereas for `GPT` the effect is stronger for `Spring` and `AM`. This difference is explained by the general bias of the corresponding `Novis` evaluations; see again Fig. 3 (top), where we also see that `GPT` and `Qwen` achieve similar response distributions for `Spring` for `positive` instances and for `Circular` for `negative` instances. In contrast, `AM` we observe different behaviors for `AM`.

Confidence. Both `GPT` and `Qwen` report the highest possible confidence score of 5 in 78% of the trials for `GPT` and in 69% of the trials for `Qwen`, independent of whether the instance is `positive` or `negative`. There is also little variance in their confidence, with regard to whether the models answered correctly or not: `GPT` had a mean confidence of 4.5 (standard deviation of 1.06) in both cases, for `Qwen` it was 4.56 (standard deviation of 0.69).

The share of answers with a certain confidence value for the different configurations is shown in Fig. 4 for `positive` (top subplot) and `negative` instances (bottom subplot). Across both models, we observe no significant differences in reported confidence scores between `positive` and `negative` instances.

Regarding the adjacency matrix-based configurations, `Novis` already leads to very high confidence for both models. Surprisingly, passing the structured adjacency matrix text in `Novis`* led to significantly lower confidence reported by both MLLMs. If the structure however is encoded visually in `AM`, `Qwen` reports slightly higher confidence values compared to `Novis`, whereas for `GPT` the behaviour is again quite different, always having confidence 5.

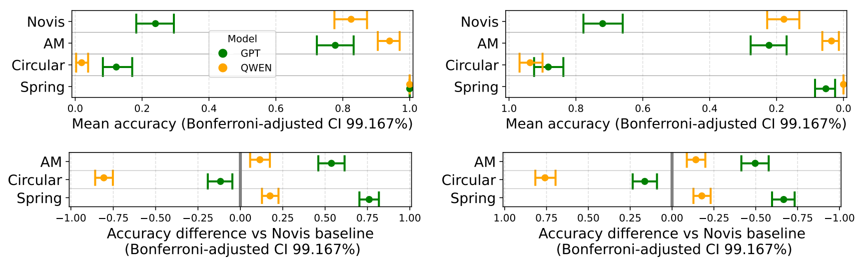

Figure 3: (top) Confidence intervals for the results of the experiments using the baseline `Novis` and the visualization styles `AM`, `Circular`, and `Spring`. (bottom) Confidence intervals for the difference of means test between the baseline at (0.0), and the visualization styles. Intervals entirely to the right (left) of zero indicate significantly better (worse) performance than `Novis`. Note the inverted *x*-axis on the right.

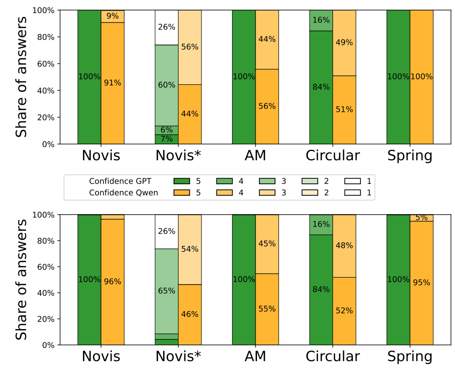

Figure 4: Share of answers with a certain confidence value for (top) `positive` and (bottom) `negative` instances.

Finally, consider the two node-link styles. The `Circular` layout seems to be the least convincing configuration across both models and introduces more doubt compared to `Novis`, with more answers given with confidence 4 (16% for `GPT` and 48–49% for `Qwen` in `Circular` vs. 0% for `GPT` and 4–9% for `Qwen` in `Novis`). The `Spring` layout seems very convincing for both MLLMs, on the `positive` instances both models always report a confidence of 5 in all trials, on the `negative` instances only `Qwen` reports a confidence of 4 on 5% of the experiments. For completeness, recall that `AM` performs slightly better than `Circular`, achieving confidence score 5 for `GPT` in 100% of the trials whereas for `Qwen` the configuration's performance is more similar to `Circular` than to `Novis`.

4 Discussion

We first evaluate RQ1: *Can the accuracy and confidence of MLLMs analyzing network data be improved when a suitable visualization is provided as part of the input?*

On all configurations, the MLLM's responses were rather influenced by a bias intrinsic to the selected MLLM and to the chosen visualization style than by the factual data provided as part of the input. In particular, this was evident from Figs. 2 and 3 where we see that for all visualizations, the distribution of choices by the MLLM was more or less the same, independent of whether or not we provided a `positive` or a `negative` instance as an input. The visualizations `AM` and `Spring` lead both models to respond that the network does contain a bridge, even making them hallucinate the existence of a bridge for `negative` instances very consistently for `Qwen` and still somewhat consistently for `GPT`. The opposite effect was observed for `Circular` where both models consistently reported that the network does not contain a bridge, even for `positive` instances. Again, the effect was slightly more pronounced for `Qwen`. At this stage, we are tempted to refute that accuracy of the MLLM's responses can be improved as the visualizations rather appear to steer the MLLM's judgment into some direction in general.

Regarding the reported confidence values, surprisingly the structured text-based input `Novis`* resulted in poorer confidence than the unstructured `Novis` even though `Novis`* should be more useful for solving the task without executing code. On the other hand, encoding the same structure visually in `AM` did not come with any loss of confidence in `GPT` and a less stark loss of confidence in `Qwen`. Thus, it seems confidence can be improved if the data is to be structured as part of the input to facilitate the MLLM's judgment (this may require more advanced prompting than in our experiment).

Secondly, we evaluate RQ2: *Does the inclusion of visualization create a bias in the decision-making process of MLLMs and, if so, is such a bias dependent on the visualization style used?*

We saw that each of the three visualization styles `AM`, `Circular` and `Spring` creates a bias towards one or the other judgment, which is consistent in its relation to the bias of `Novis` over both models. We also observed that a bias exists for the raw textual data `Novis` as well which is slightly mitigated using structure in `Novis`*. In comparison, the bias by the visualizations is stronger than the bias for `Novis` (see Fig. 3) and hence including a visualization as part of the input is far more likely to steer an MLLM's judgment – independently of whether or not this achieves a desired effect.

5 Conclusion

In our experiments, both MLLMs appeared to be driven far more by the visualization design than by the underlying data; an effect that may seem to parallel human perception but with differences; e.g., the MLLM perceives no significant differences in the illustration pairs in Fig. 1 while a human likely would. In fact, we intended creating purposefully misleading visualization mirages in a pilot study to make MLLMs draw wrong conclusions until we noticed that two edges between two components in the `Spring` configuration were already consistently and confidently interpret as a single bridge.

For visualization research, our results indicate that evaluating MLLMs as "human-like" readers can be problematic, as although they might produce similar results in the aggregate, the fine-grained distinction tells a different story. Understanding these differences is essential as MLLMs become more and more widely used tools for prototyping, studying, and evaluating visualization techniques; a great avenue for future work. For generative AI, the demonstrated increase in confidence scores when providing visualizations in addition to raw data input may be a desirable effect in automated pipelines. However, the fact that visualization itself appears to create strong biases indicates that strong caution has to be taken when designing an application that steers MLLMs using visualization.

Limitations. Our results have several limitations that must be taken into account to avoid over-generalizations. We focused on a single task, the detection of a bridge. Future experiments could investigate if similar effects can be observed for other network tasks. Moreover, we restricted ourselves to two standard layout techniques with circular layout and a force-directed algorithm. Other state-of-the-art algorithms might be taken into account in future studies.

We investigated both a recent commercial MLLM model and an open-source model and observed non-trivial differences between them. Hence, it is not so clear if our results can be generalized to other MLLM models. Finally, our experiments used zero-shot prompting and did not allow for tool access. Potentially, a more fine-tuned prompt, using, for instance, few-shot prompting or RAG, may yield results that are not influenced by the MLLM's bias.

REFERENCES

[1] A. R. Ahmed, F. D. Luca, S. Devkota, S. G. Kobourov, and M. Li. Multicriteria scalable graph drawing via stochastic gradient descent, $(SGD)^2$. *IEEE Trans. Vis. Comput. Graph.*, 28(6):2388–2399, 2022. doi: 10.1109/TVCG.2022.3155564 1

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi: 10.1126/science.286.5439.509 2

[3] A. Bendeck and J. T. Stasko. An empirical evaluation of the GPT-4 multimodal language model on visualization literacy tasks. *IEEE Trans. Vis. Comput. Graph.*, 31(1):1105–1115, 2025. doi: 10.1109/TVCG.2024.3456155 1

[4] L. Bounegru, T. Venturini, J. Gray, and M. Jacomy. Narrating networks. *Digital Journalism*, 5(6):699–730, 2017. doi: 10.1080/21670811.2016.1186497 1

[5] H. Förster, F. Klesen, T. Dwyer, P. Eades, S.-H. Hong, S. Kobourov, G. Liotta, K. Misue, F. Montecchiani, A. Pastukhov, and F. Schreiber. Graphtrials: Visual proofs of graph properties. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–15, 2025. doi: 10.1109/TVCG.2025.3577533 1

[6] J. Hong, C. Seto, A. Fan, and R. Maciejewski. Do llms have visualization literacy? an evaluation on modified visualizations to test generalization in data interpretation. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–13, 2025. doi: 10.1109/TVCG.2025.3536358 1

[7] J. Lai, W. Gan, J. Wu, Z. Qi, and P. S. Yu. Large language models in law: A survey. *AI Open*, 5:181–196, 2024. doi: 10.1016/J.AIOPEN.2024.09.002 1

[8] Y. Li, S. Wang, H. Ding, and H. Chen. Large language models in finance: A survey. In *4th ACM International Conference on AI in Finance, ICAIF 2023, Brooklyn, NY, USA, November 27-29, 2023*, pp. 374–382. ACM, 2023. doi: 10.1145/3604237.3626869 1

[9] A. M. McNutt, G. L. Kindlmann, and M. Correll. Surfacing visualization mirages. In R. Bernhaupt, F. F. Mueller, D. Verweij, J. Andres, J. McGrenere, A. Cockburn, I. Avellino, A. Goguey, P. Bjøn, S. Zhao, B. P. Samson, and R. Kocielnik, eds., *CHI '20: CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, April 25-30, 2020*, pp. 1–16. ACM, 2020. doi: 10.1145/3313831.3376420 1

[10] T. Munzner. Visualization analysis and design. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Courses*, pp. 1–2, 2025. 1

[11] Q. H. Nguyen, P. Eades, and S. Hong. On the faithfulness of graph visualizations. In W. Didimo and M. Patrignani, eds., *Graph Drawing - 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers*, vol. 7704 of *Lecture Notes in Computer Science*, pp. 566–568. Springer, 2012. doi: 10.1007/978-3-642-36763-2_55 1

[12] G. Sun, Y. Wu, R. Liang, and S. Liu. A survey of visual analytics techniques and applications: State-of-the-art research and future challenges. *J. Comput. Sci. Technol.*, 28(5):852–867, 2013. doi: 10.1007/S11390-013-1383-8 1

[13] R. Tamassia, ed. *Handbook on Graph Drawing and Visualization*. Chapman and Hall/CRC, 2013. 1

[14] R. J. Tibshirani and B. Efron. An introduction to the bootstrap. *Monographs on statistics and applied probability*, 57(1):1–436, 1993. 2

[15] X. Wang, K. Yen, Y. Hu, and H. Shen. Smartgd: A gan-based graph drawing framework for diverse aesthetic goals. *IEEE Trans. Vis. Comput. Graph.*, 30(8):5666–5678, 2024. doi: 10.1109/TVCG.2023.3306356 1

[16] H. Zhou, B. Gu, X. Zou, Y. Li, S. S. Chen, P. Zhou, J. Liu, Y. Hua, C. Mao, X. Wu, Z. Li, and F. Liu. A survey of large language models in medicine: Progress, application, and challenge. *CoRR*, abs/2311.05112, 2023. doi: 10.48550/ARXIV.2311.05112 1