





Ask and You Shall Receive (a Graph Drawing): Testing ChatGPT’s Potential to Apply Graph Layout Algorithms

Sara Di Bartolomeo¹ , Giorgio Severi¹ , Victor Schetinger² , Cody Dunne¹ 
¹ Northeastern University, ² TU Wien

Abstract

Large language models (LLMs) have recently taken the world by storm. They can generate coherent text, hold meaningful conversations, and be taught concepts and basic sets of instructions—such as the steps of an algorithm. In this context, we are interested in exploring the application of LLMs to graph drawing algorithms by performing experiments on ChatGPT, one of the most recent cutting-edge LLMs made available to the public. These algorithms are used to create readable graph visualizations. The probabilistic nature of LLMs presents challenges to implementing algorithms correctly, but we believe that LLMs’ ability to learn from vast amounts of data and apply complex operations may lead to interesting graph drawing results. For example, we could enable users with limited coding backgrounds to use simple natural language to create effective graph visualizations. Natural language specification would make data visualization more accessible and user-friendly for a wider range of users. Exploring LLMs’ capabilities for graph drawing can also help us better understand how to formulate complex algorithms for LLMs; a type of knowledge that could transfer to other areas of computer science. Overall, our goal is to shed light on the exciting possibilities of using LLMs for graph drawing—using the Sugiyama algorithm as a sample case—while providing a balanced assessment of the challenges and opportunities they present. A free copy of this paper with all supplemental materials to reproduce our results is available on osf.io.

CCS Concepts

• **Human-centered computing** → **Graph drawings**; • **Computing methodologies** → **Artificial intelligence**;

1. Introduction

A graph layout algorithm maps nodes and edges in a graph to coordinates in space—an essential step in rendering visible its abstract topology. These algorithms generally optimize for readability criteria such as reducing the number of edge crossings in the resulting drawing [Pur02, DBCSD23]. Decades of research in the field have produced many layout algorithms, such as the popular Sugiyama algorithm [STT81] for layered graphs, which we use in this paper. When considering the needs of a user, however, it is challenging to understand which algorithm to choose and how to control its parameters to obtain a desired result [KM20]. Whether the graph is a social network, -omics diagrams, or a subway map, a domain expert will have an implicit, subjective expectation of what needs to be seen. Translating these needs into a choice of aesthetic criteria is neither a simple nor an exact task.

An ideal system would let a user provide a graph and explain in their own words how to visualize it. For example, an art historian could express their needs as “I want to see the collaboration network of Kandinsky with him at the center, thicker edges showing more co-exhibitions, and with all other Russian painters visible”. In contrast, a graph drawing researcher might say: “I want to minimize edge crossings and...” This interaction is not yet possible, but OpenAI’s ChatGPT has recently enabled the general public to use large language models (LLMs) and demonstrates the potential for natural language interfaces.

In September 2022, Jacob Brazeal [Jac22] tested using GPT-3 to run a path-finding algorithm. His results showed the model could apply

multiple steps correctly, inspiring us to explore the feasibility of using ChatGPT for more complicated graph layout algorithms. In particular, we dissected a layout algorithm into multiple “bite-sized” tasks that a language model could interpret. Since both inputs and outputs of these algorithms can be represented as text, e.g., inputting a list of nodes and edges and outputting a table of coordinates, in theory LLMs could act as general-purpose solvers. Applying a graph layout algorithm via a generative text model would require no programming, just a natural-language description of the problem. This may enable users to more rapidly specify novel constraints on the layout. However, generative models have downsides: (a) they require careful consideration of the words used to describe the graph and the problem, and (b) due to their stochastic nature, the correctness of the outputs cannot be ensured a priori.

To explore the possible benefits and downsides of this approach, we designed a set of experiments that would help illustrate the art of the possible as well as evaluate LLM correctness against existing algorithms. We also discuss how to best formulate graph drawing problems so that they are easily understood by an LLM. Even though some mistakes and imprecision in the returned solutions are to be expected using this heuristic LLM-based layout algorithm, our results show that the majority of the results returned are valid solutions. These results, combined with the fast pace at which LLMs are evolving, lead us to believe that we can expect more effective layout results in the near future. All the code used for the analysis—as well as our supplemental material with experiments, all queries, and all answers—is available at osf.io/n5rxd.

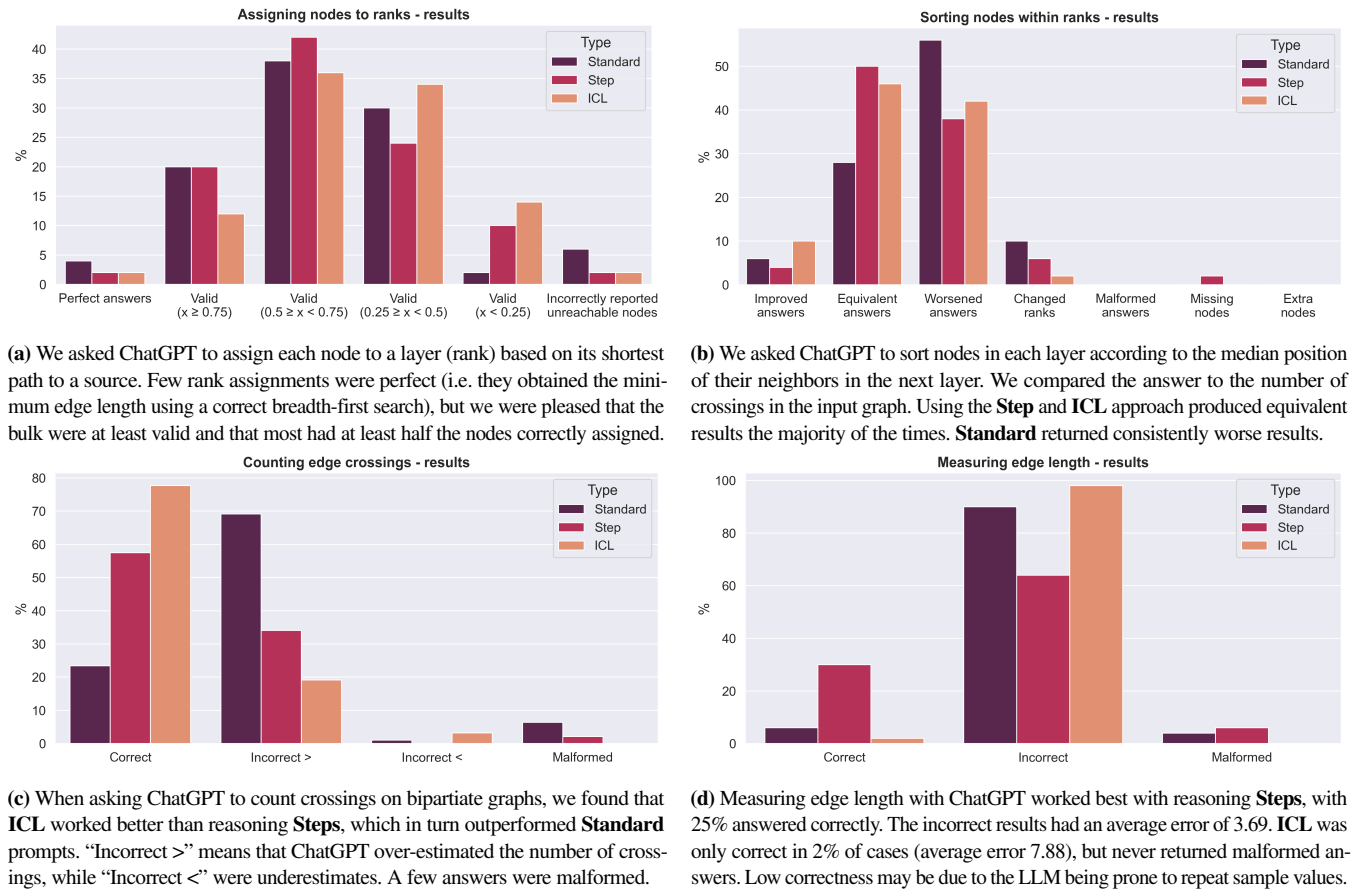


Figure 1: Results from our experiments asking ChatGPT to perform graph drawing tasks. See more results in our appendix.

2. Background

A language model defines a probability distribution over a sequence of linguistic units, and can be used to predict the most likely next unit in a sequence. ChatGPT [Ope22], a recent addition to the GPT (Generative Pre-trained Transformer) family [RNSS18, RWC*20, BMR*20] of causal language models, is currently regarded as the state of the art of conversational agents. Given an existing sequence of tokens, where each token represents a pre-defined sub-component of a natural language word, corresponding to a unique integer number, these models are trained to iteratively and autoregressively predict the most likely next tokens. These tokens, chained-together, end up forming words, sentences, and even entire documents. This characteristic training procedure distinguishes them from the other famous family of transformer models (represented by BERT [DCLT19]) trained for masked language modeling, where the objective is to learn to fill in missing tokens in a sequence.

In contrast to its predecessors, ChatGPT leverages Reinforcement Learning from Human Feedback (RLHF) to align responses generated by the model with the expectations of end users. The exact details of ChatGPT’s model have not been fully disclosed by OpenAI. The closest documented system is TEXT-DAVINCI-003 [Ope23], which is described as a version of InstructGPT [OWJ*22] fine-tuned with RLHF. We are not in a position to fine-tune ChatGPT for our tasks, but we believe that its RLHF training, together with its simple interface by which users can

specify complex prompts, makes it particularly well-suited to follow the user-specified steps of a graph layout algorithm.

Beyond *natural* language, GPT models have shown impressive performance when generating different textual data such as programming language code [CTJ*21], and multiple *emergent* abilities [WTB*23] have been observed with the progressive increases in model capacity and training volumes. We can situate our paper within a larger body of work that tries to explore such emergent abilities. The majority of this exploration has been focused in NLP tasks [KCK*23, MIB*23], which is not strictly our case. However, the manipulation of a graph’s topological space within the internal representation of a LLM can be related to the problem of grounded conceptual spaces [PP22]. The fast evolution of generative models has brought growing interest in using them for data visualization [SDBEA*23]. Although neural network approaches have been tested for graph layout algorithms [GLA*21, KMP18, DLHK19], this iteration of generative models is so recent that there has been little research yet on applying them to graph drawing. Hence, it is our intention to test to what extent these models can be used to apply graph layout algorithms.

3. Experiments

We re-create the classic Sugiyama [STT81] layout algorithm for layered graphs using ChatGPT. This algorithm had several component tasks we could test, as well as several utility tasks for us to evaluate. It is

important to keep in mind that the expected result of a graph layout algorithm is a coordinate assignment, mapping nodes to coordinates in space. Thus, we expected the LLM to generate numerical values for every node, *not* a graphical rendering. While we do experiment with generating SVG illustrations in the appendix, rendering is not the focus of this paper. We treat rendering as a successive step which can be done with any graphical library.

The Sugiyama algorithm incorporates several steps: (1) cycle removal, (2) layer assignment, (3) sorting nodes within layers, and (4) final positioning. Here, we prioritize discussing the pivotal steps of layer assignment (section 3.1) and sorting nodes within layers (section 3.2). We also explore the related tasks of counting crossings and edge length to evaluate the quality of the layout (section 3.3). Actual queries to ChatGPT and additional discussion of these tasks is available in the appendix. Our appendices also detail additional experiments with executing several utility (and fun!) tasks using ChatGPT. These include explaining the Sugiyama algorithm in poetry, generating test graphs, converting between file formats, defining additional graph properties, generating graphs from scenes (e.g. for movie StoryLines [IM12]) as well as entirely new scenes from graphs, and creating SVG illustrations of the graph.

We ran our experiments using graphs from Rome-Lib [BGL*00], a popular benchmark dataset for graph layout algorithms. We used only the graphs with 10 or 11 nodes so that we did not exceed the fixed budget of tokens that ChatGPT could process simultaneously. Unless otherwise noted, we randomly selected subsets of 50 graphs to run our experiments. To test the correctness of the results, every answer from ChatGPT was compared against a ground truth; we show the results from these comparisons in Figure 1. All the examples presented were ran on clean chat threads to avoid previous inputs contaminating the results. We used OpenAI’s web interface (at the time of writing there is no available API) and the ChatGPT Plus default model from 2023-02-13 – 2023-03-01.

Although it is possible to ask ChatGPT to write code to execute the tasks we defined, we wanted to test its ability to apply the algorithm and reason on the problems without executing any code. We tried several approaches to formulating problems. In some cases, we gave ChatGPT examples of solved problems in the query along with the usual explanation of the task. We mark this In-Context Learning (ICL) [XRLM22] when it is used. To construct these examples, we randomly sample $k = [3, 5]$ other instances of the task from our pool, and augment the prompt with the input and correct answers to those instances. We also experimented with splitting the task in to *reasoning steps* and asking ChatGPT to provide answers for each step (these are marked **Step**). This technique is also known as chain-of-thought prompting (CoT). We use a zero-shot CoT [KGR*22], which means that we do not provide step-by-step examples of the solution to similar instances of the task, but rather ask the model to write down the detailed explanation for each step towards the original task. Cases in which we explained the task to ChatGPT but provided no examples and requested no reasoning steps are marked as **Standard**. Examples and comparisons between the different approaches can be found in the appendix.

3.1. Layer assignment

Layer assignment (a.k.a. rank assignment) is an important step in the Sugiyama layout [STT81], as well as the default *dot* algorithm [GKNV93] in Graphviz. Graphs without an inherent layering

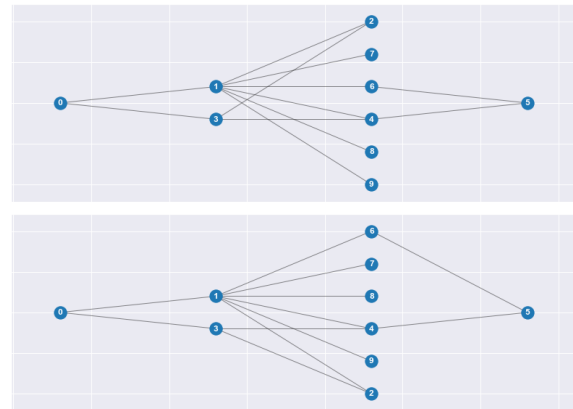


Figure 2: An example of a more readable graph produced by asking ChatGPT to sort nodes in each layer using the median heuristic. The input graph (top) has 7 crossings, while the output (bottom) has only 2.

must have each node assigned to a layer before a layered graph layout algorithm can be used. Of the many methods for assigning layers [Sug02, HN13], we choose a simple approach: select a source, then assign to every node a layer that is equal to the length of the shortest path from it to the source (i.e., a node that is 2 hops from a source on layer 0 will be assigned to layer 2). This is the method used by STRATISFIMAL LAYOUT [DBRGD21]. (Note that layer assignment with directed graphs is usually preceded by a cycle removal step; we considered all edges as undirected.) This straightforward approach requires either recursion or a queue data structure to conduct a breadth-first search.

For each graph, we recorded the percentage of nodes assigned to the correct layer. Figure 1a shows us that ChatGPT rarely assigns the layers perfectly and occasionally will even incorrectly report nodes as unreachable. However, the vast bulk of answers were at least valid assignments and most of the time ChatGPT assigned at least half of the nodes correctly. Interestingly, ChatGPT often recognized in its answers that what we were asking was the application of a breadth-first search—even without us specifying so in the prompt.

3.2. Sorting nodes within layers

The next step of the Sugiyama algorithm [STT81] is to sort nodes within each layer. There are multiple approaches (see [Sug02, HN13]) but we use the *median heuristic*. We sequentially sweep across the layers of the graph, consider each layer in turn, and position each node within it according to the median of its neighbors. Iteratively sweeping one direction then the reverse gives us a relative positioning of the nodes within each layer, from which they can be evenly distributed (e.g. on a grid). Thus, in a horizontal layered graph, the x position of a node is defined by its layer, while the y position is defined by this sorting. While the process is meant to be repeated for many iterations or until convergence, we asked ChatGPT to do it only once. To simplify the description in the following steps, we removed edges that connect nodes that share the same layer. The results (for a total of 50 graphs) are illustrated in Figure 1b. Figure 2 shows a rendering of a graph given as input vs. a rendering of ChatGPT’s response. Below is an example prompt used for this task, followed by the corresponding answer.

```
This is a list of directed edges in a layered graph.
[(0, 1), (1, 2), (3, 2), (0, 3),
(4, 5), (6, 5), (1, 7), (1, 6), (1, 4), (3, 4), (1, 8), (1, 9)]
```

```
This is the description of what nodes are contained
in what layer:
Layer 0: [0]
Layer 1: [1, 3]
Layer 2: [2, 7, 6, 4, 8, 9]
Layer 3: [5]
```

```
Write no code and no explanations. Start from layer 0.
Consider every node in layer 0, and for each node consider
its neighbors in the next layer. For each node in layer
0, calculate the positions of the neighbors in the next layer.
Then, sort the nodes in layer 0 based on the median position
of their neighbors. If a number is NaN, then it is equal to 0.
If a node has no neighbors, assign an associated median of 0.
Repeat the process for every layer.
```

```
Finally, I want you to write the layers and the nodes in
them once more, but the nodes in the layers have to be
sorted according to their associated median value.
```

```
Like this:
Layer 0: [<sorted nodes in layer 0>]
If node A has an associated median value that is less
than the associated median value of node B, then node
A must come before node B.
```

Query

```
Layer 0: [0]
Layer 1: [3, 1]
Layer 2: [7, 6, 8, 9, 2, 4]
Layer 3: [5]
```

Answer

3.3. Counting crossings or edge length

Counting crossings is a fundamental step to evaluate the quality of a layout [BETT94]. With our current graph representations, crossings can only appear between one layer and the next. To simplify the task, we decided to split the problem in sub-graphs: each pair of consecutive layers was described as a bipartite graph. We therefore generated $n - 1$ queries for each graph, where n is the number of layers in a graph (20 original graphs resulting in 94 bipartite graphs in total). We then compared the solutions given by ChatGPT against the ground truth. We found that both prompt shaping techniques were drastically more effective than the **Standard** prompts in eliciting the correct answer, as shown by Figure 1c. While **ICL** appeared to lead to the most successful outcomes, this result may be biased by the relatively large quantity of results with 0 crossings, which were thus more likely to appear in the provided examples.

Edge length is another key readability criteria. The shorter the edge, the easier it is for a human reader to follow [Pur02]. It is therefore desirable to minimize overall edge length in the drawing. Assuming a unitary distance between adjacent layers, we can compute the length of each edge as the absolute value of the index of the layer of the source of the edge minus the index of the target of the edge: $|\text{layers.indexOf}(e.\text{source}) - \text{layers.indexOf}(e.\text{target})|$. The method we used to assign layers (section 3.1), when correctly performed, produces the minimum possible edge length—each has a length of 1. In this case, counting the total edge length is equivalent to counting the number of edges. However, we did not explicitly provide this information to ChatGPT—instead specifying that the distance between consecutive layers was 1—and asked it to count the total edge length. On this task, ChatGPT was surprisingly able to return an exact result for every graph, without necessitating **ICL** or reasoning **Steps**.

To test this ability on a more complex case, we created a new layer assignment for 50 graphs by assigning each node to a random layer. Thus the length of each edge was no longer always 1. Our results from asking ChatGPT to compute edge length with these new graphs are shown in Figure 1d. There were now many more incorrect

answers, but the differences between the performance of the different prompting approaches become much more evident: using **Steps** we had considerably better results than **ICL** or **Standard**.

4. Discussion and conclusions

There is potential for LLMs to be used in visualization and graph drawing—getting to the point where we obtain reliable results could enable users with no coding backgrounds to create novel visualizations without having to write or execute any code. Currently, however, we discovered substantial limitations with this approach. But the encouraging results we obtained on some sub-tasks, coupled with the bottleneck speed of LLM improvements, leads us to be optimistic about the future utility of LLMs for graph drawing tasks. More examples that could be relevant for a natural language interfaces are explored in our appendix.

Potentially invalid results: Asking LLMs to perform layout algorithms can potentially lead to invalid results. The stochastic nature of LLMs and the challenges involved in parsing natural language means there can be few guarantees. Checking the solution manually or against a traditionally-computed baseline is necessary if exact answers are needed. LLM alignment is an active field of research and result quality is expected to improve in the near future, but we doubt that the problem can be fully solved. However, there is one promising avenue to explore. The model we tested was not trained specifically for our algorithmic tasks, but was still able to provide mostly valid and often good answers due to its training as a next-token predictor. Fine-tuning language models for specific tasks is a common practice which could lead to significant improvements in our graph drawing performance.

Prompt engineering: It is important to keep in mind the influence of the prompt over the resulting response that is obtained from ChatGPT. Any difference in wording can give a different result. We experimented with several established prompting techniques, but exploring the entire spectrum of prompt-crafting is outside of our scope. We refer the reader to Liu et al. [LYF*21] for a more complete investigation of this topic.

Scalability limitations: Most transformer-based language models, including ChatGPT, have a fixed budget of tokens they can process simultaneously, including both input and output. A token is a word or part of a word that represents a unitary element of the input and output sequences. As noted earlier, we exclusively experimented with small graphs to avoid this issue. This token limit places a size limit on input graphs, reducing the utility of LLM graph drawing approaches. However, as the size of language models has increased, so has their token processing capacity. We believe scalability will become less of an issue over time.

Lessons learned: We found that breaking down complex tasks into smaller chunks is key to achieving viable answers, instead of requiring the application of the entire algorithm. Similarly to what other research on generative models reported, we also observed stark differences between prompt-shaping techniques on the quality of the results. Our results suggest that prompting techniques based on step-by-step reasoning tend to perform well on this type of tasks. We conclude that there appears to be a bright future for natural language interfaces, but the current generation of LLMs is still not adequately suited for the task.

5. Acknowledgments

This work was supported in part by the U.S. National Science Foundation (NSF) under award number IIS-2145382.

References

- [BETT94] BATTISTA G. D., EADES P., TAMASSIA R., TOLLIS I. G.: Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry* 4, 5 (1994), 235–282. URL: <https://www.sciencedirect.com/science/article/pii/092577219400014X>, doi:[https://doi.org/10.1016/0925-7721\(94\)00014-X](https://doi.org/10.1016/0925-7721(94)00014-X).
- [BGL*00] BATTISTA G. D., GARG A., LIOTTA G., PARISE A., TAMASSIA R., TASSINARI E., VARGIU F., VISMARA L.: Drawing directed acyclic graphs: An experimental study. *International Journal of Computational Geometry & Applications* 10, 06 (Dec. 2000), 623–648. doi:[10.1142/s0218195900000358](https://doi.org/10.1142/s0218195900000358).
- [BMR*20] BROWN T., MANN B., RYDER N., SUBBIAH M., KAPLAN J. D., DHARIWAL P., NEELAKANTAN A., SHYAM P., SASTRY G., ASKELL A., AGARWAL S., HERBERT-VOSS A., KRUEGER G., HENIGHAN T., CHILD R., RAMESH A., ZIEGLER D., WU J., WINTER C., HESSE C., CHEN M., SIGLER E., LITWIN M., GRAY S., CHESS B., CLARK J., BERNER C., MCCANDLISH S., RADFORD A., SUTSKEVER I., AMODEI D.: Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (2020), vol. 33, Curran Associates, Inc., pp. 1877–1901. URL: <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf>.
- [CTJ*21] CHEN M., TWOREK J., JUN H., YUAN Q., PINTO H. P. D. O., KAPLAN J., EDWARDS H., BURDA Y., JOSEPH N., BROCKMAN G., RAY A., PURI R., KRUEGER G., PETROV M., KHLAAAF H., SASTRY G., MISHKIN P., CHAN B., GRAY S., RYDER N., PAVLOV M., POWER A., KAISER L., BAVARIAN M., WINTER C., TILLET P., SUCH F. P., CUMMINGS D., PLAPPERT M., CHANTZIS F., BARNES E., HERBERT-VOSS A., GUSS W. H., NICHOL A., PAINO A., TEZAK N., TANG J., BABUSCHKIN I., BALAJI S., JAIN S., SAUNDERS W., HESSE C., CARR A. N., LEIKE J., ACHIAM J., MISRA V., MORIKAWA E., RADFORD A., KNIGHT M., BRUNDAGE M., MURATI M., MAYER K., WELINDER P., MCGREW B., AMODEI D., MCCANDLISH S., SUTSKEVER I., ZAREMBA W.: Evaluating large language models trained on code, July 2021. doi:[10.48550/arXiv.2107.03374](https://doi.org/10.48550/arXiv.2107.03374).
- [DBCSD23] DI BARTOLOMEO S., CRNOVRANIN T., SAFFO D., DUNNE C.: Designing computational evaluations for graph layout algorithms: the state of the art, Mar 2023. URL: osf.io/ms27r.
- [DBRGD21] DI BARTOLOMEO S., RIEDEWALD M., GATTERBAUER W., DUNNE C.: STRATISFIMAL LAYOUT: A modular optimization model for laying out layered node-link network visualizations. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 324–334. VIS '21. Preprint & Supplemental Material: <https://osf.io/qdyt9>. doi:[10.1109/TVCG.2021.3114756](https://doi.org/10.1109/TVCG.2021.3114756).
- [DCLT19] DEVLIN J., CHANG M.-W., LEE K., TOUTANOVA K.: BERT: Pre-training of deep bidirectional transformers for language understanding, May 2019. doi:[10.48550/arXiv.1810.04805](https://doi.org/10.48550/arXiv.1810.04805).
- [DLHK19] DE LUCA F., HOSSAIN M. I., KOBOUROV S.: Symmetry detection and classification in drawings of graphs. In *Graph Drawing and Network Visualization* (Cham, 2019), Archambault D., Tóth C. D., (Eds.), Springer International Publishing, pp. 499–513. doi:[10.1007/978-3-030-35802-0_38](https://doi.org/10.1007/978-3-030-35802-0_38).
- [GKNV93] GANSNER E. R., KOUTSOFIOS E., NORTH S. C., VO K.-P.: A technique for drawing directed graphs. *IEEE Transactions on Software Engineering* 19, 3 (1993), 214–230. doi:[10.1109/32.221135](https://doi.org/10.1109/32.221135).
- [GLA*21] GIOVANNANGELI L., LALANNE F., AUBER D., GIOT R., BOURQUI R.: Deep neural network for DrawiNg networks. In *Lecture Notes in Computer Science*. Springer International Publishing, 2021, pp. 375–390. doi:[10.1007/978-3-030-92931-2_27](https://doi.org/10.1007/978-3-030-92931-2_27).
- [HN13] HEALY P., NIKOLOV N. S.: Hierarchical drawing algorithms. In *Handbook on Graph Drawing and Visualization*, Tamassia R., (Ed.), Chapman and Hall/CRC, 2013, pp. 409–453.
- [Jac22] JACOB BRAZEAL: Using GPT-3 to pathfind in random graphs. <https://jacobbrazeal.wordpress.com/2022/09/23/gpt-3-can-find-paths-up-to-7-nodes-long-in-random-graphs/>, 2022.
- [KCK*23] KOCOŃ J., CICHECKI I., KASZYCA O., KOCHANEK M., SZYDŁO D., BARAN J., BIELANIEWICZ J., GRUZA M., JANZ A., KANCLERZ K., KOCOŃ A., KOPTYRA B., MIELESZCZENKO-KOWSZEWICZ W., MIŁKOWSKI P., OLEKSY M., PIASECKI M., RADLIŃSKI Ł., WOJTASIK K., WOŹNIAK S., KAZIENKO P.: ChatGPT: Jack of all trades, master of none, Feb. 2023. doi:[10.48550/arXiv.2302.10724](https://doi.org/10.48550/arXiv.2302.10724).
- [KGR*22] KOJIMA T., GU S. S., REID M., MATSUI Y., IWASAWA Y.: Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems* (Oct. 2022). doi:[10.48550/arXiv.2205.11916](https://doi.org/10.48550/arXiv.2205.11916).
- [KM20] KWON O.-H., MA K.-L.: A deep generative model for graph layout. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (Jan. 2020), 665–675. doi:[10.1109/TVCG.2019.2934396](https://doi.org/10.1109/TVCG.2019.2934396).
- [KMP18] KLAMMLER M., MCHEDLIDZE T., PAK A.: Aesthetic discrimination of graph layouts. In *Graph Drawing and Network Visualization* (Cham, 2018), Biedl T., Kerren A., (Eds.), Springer International Publishing, pp. 169–184. doi:[10.1007/978-3-030-04414-5_12](https://doi.org/10.1007/978-3-030-04414-5_12).
- [IM12] LIU MA Y. T. K.: Design considerations for optimizing StoryLine visualizations. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2679–2688. doi:[10.1109/TVCG.2012.212](https://doi.org/10.1109/TVCG.2012.212).
- [LYF*21] LIU P., YUAN W., FU J., JIANG Z., HAYASHI H., NEUBIG G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, July 2021. arXiv:[2107.13586](https://arxiv.org/abs/2107.13586), doi:[10.48550/arXiv.2107.13586](https://doi.org/10.48550/arXiv.2107.13586).
- [MIB*23] MAHOWALD K., IVANOVA A. A., BLANK I. A., KANWISHER N., TENENBAUM J. B., FEDORENKO E.: Dissociating language and thought in large language models: A cognitive perspective, Jan. 2023. arXiv:[2301.06627](https://arxiv.org/abs/2301.06627), doi:[10.48550/arXiv.2301.06627](https://doi.org/10.48550/arXiv.2301.06627).
- [Ope22] OPENAI: ChatGPT: Optimizing language models for dialogue, Nov. 2022. URL: <https://openai.com/blog/chatgpt/>.
- [Ope23] OPENAI: API, 2023. URL: <https://platform.openai.com>.
- [OWJ*22] OUYANG L., WU J., JIANG X., ALMEIDA D., WAINWRIGHT C., MISHKIN P., ZHANG C., AGARWAL S., SLAMA K., GRAY A., SCHULMAN J., HILTON J., KELTON F., MILLER L., SIMENS M., ASKELL A., WELINDER P., CHRISTIANO P., LEIKE J., LOWE R.: Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems* (Oct. 2022). doi:[10.48550/arXiv.2203.02155](https://doi.org/10.48550/arXiv.2203.02155).
- [PP22] PATEL R., PAVLICK E.: Mapping language models to grounded conceptual spaces. In *International Conference on Learning Representations* (Jan. 2022). URL: <https://openreview.net/forum?id=gJcEM8sxHK>.
- [Pur02] PURCHASE H. C.: Metrics for graph drawing aesthetics. *Journal of Visual Languages & Computing* 13, 5 (2002), 501–516. doi:[10.1006/jvlc.2002.0232](https://doi.org/10.1006/jvlc.2002.0232).
- [RNSS18] RADFORD A., NARASIMHAN K., SALIMANS T., SUTSKEVER I.: Improving language understanding by generative pre-training. p. 12.
- [RWC*20] RADFORD A., WU J., CHILD R., LUAN D., AMODEI D., SUTSKEVER I.: Language models are unsupervised multitask learners, 2020.
- [SDBEA*23] SCHETINGER V., DI BARTOLOMEO S., EL-ASSADY M., MCNUTT A. M., MILLER M., ADAMS J. L.: Doom or deliciousness: Challenges and opportunities for visualization in the age of generative models, Jan 2023. doi:[10.31219/osf.io/3jrcm](https://doi.org/10.31219/osf.io/3jrcm).
- [STT81] SUGIYAMA K., TAGAWA S., TODA M.: Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics* 11, 2 (1981), 109–125. doi:[10.1109/TSMC.1981.4308636](https://doi.org/10.1109/TSMC.1981.4308636).
- [Sug02] SUGIYAMA K.: *Graph Drawing and Applications for Software and Knowledge Engineers*. World Scientific, 2002. doi:[10.1142/4902](https://doi.org/10.1142/4902).
- [WTB*23] WEI J., TAY Y., BOMMASANI R., RAFFEL C., ZOPH B., BORGEAUD S., YOGATAMA D., BOSMA M., ZHOU D., METZLER D., CHI E. H., HASHIMOTO T., VINYALS O., LIANG P., DEAN J., FEDUS W.: Emergent abilities of large language models. *Transactions on Machine Learning Research* (Jan. 2023). doi:[10.48550/arXiv.2206.07682](https://doi.org/10.48550/arXiv.2206.07682).
- [XRLM22] XIE S. M., RAGHUNATHAN A., LIANG P., MA T.: An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations* (Jan. 2022). doi:[10.48550/arXiv.2111.02080](https://doi.org/10.48550/arXiv.2111.02080).