Training AI to Recognize Realizable Gauss diagrams: the Same Instances Confound AI and Human Mathematicians

Abdullah Khan[1][a], Alexei Lisitsa[2][b], Alexei Vernitski[1][c]
[1] *Department of Mathematical Sciences, University of Essex, Essex, UK*
[2]*Department of Computer Science, University of Liverpool, Liverpool, UK*
ak20749@essex.ac.uk, a.lisitsa@liverpool.ac.uk,asvern@essex.ac.uk

Keywords: computational topology, Gauss diagrams, realizable diagrams, machine learning

Abstract: Recent research in computational topology found sets of counterexamples demonstrating that several recent mathematical articles purporting to describe a mathematical concept of realizable Gauss diagrams contain a mistake. In this study we propose several ways of encoding Gauss diagrams as binary matrices, and train several classical ML models to recognise whether a Gauss diagram is realizable or unrealizable. We test their accuracy in general, on the one hand, and on the counterexamples, on the other hand. Intriguingly, accuracy is good in general and surprisingly bad on the counterexamples. Thus, although human mathematicians and AI perceive Gauss diagrams completely differently, they tend to make the same mistake when describing realizable Gauss diagrams.

1 Introduction

The concept of realizable Gauss diagrams belongs to the mathematical area of topology and, more specifically, the study of closed planar curves. For a closed planar curve, such as shown at (Fig. 1, a) its *Gauss code* (or *Gauss word*) can be obtained by labelling all intersection points by different symbols (or numbers) and then by travelling all the way along the curve and writing down the labels encountered on the way. For example, one of the Gauss codes of the curve shown at (Fig. 1, a) is 123123. It is easy to see that the Gauss code of a curve with n intersection points has a length $2n$ and it is a *double occurrence* word, that is, each symbol occurs exactly twice in it. With any *double occurrence* word w one can associate its chord diagram; it consists of a circle with all symbols of w placed clockwise around the circle and chords which link the points labelled by the same symbol, as in (Fig. 1,b).

If a double occurrence word and its corresponding chord diagram can be obtained from a planar curve, both the word and the diagram are called *realizable*. Not every Gauss diagram is realizable; for example, the diagrams in (Fig. 2) and (Fig. 3) are not realizable.

[a] https://orcid.org/0000-0002-3056-008X
[b] https://orcid.org/0000-0002-3820-643X
[c] https://orcid.org/0000-0003-0179-9099

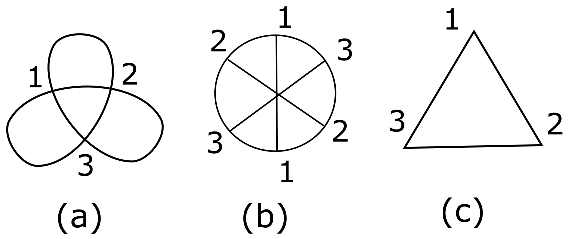

Figure 1: a) a planar curve; b) its Gauss diagram and c) its interlacement graph. The corresponding Gauss word is **123123**

In the 1840s Gauss (Gauss, 1900) asked which chords diagrams are realizable, and gave a necessary, but not sufficient condition; *in a realizable diagram every chord intersects an even number of other chords*. We will refer to the chord diagrams satisfying this condition as Gauss diagrams. Full characterisation of realizability was first provided in the 1930s by Dehn (Dehn, 1936). Dozens of variants and re-reformulations of the criteria and algorithms for checking realizability have appeared since then, see e.g. (Marx, 1969; Francis, 1969; Lovász and Marx, 1976; Rosenstiehl, 1976; Rosenstiehl and Tarjan, 1984; Dowker and Thistlethwaite, 1983; Kurlin, 2008; Shtylla et al., 2009). It was shown in (Rosenstiehl, 1976) that the realizability of a Gauss diagram can be decided just using its interlacement graph. An *interlacement graph* of a chord diagram is an undirected graph whose vertices are the chords of the diagram, and in which there is an edge between two vertices, if the chords corresponding to these vertices intersect. For example, (Fig. 1, c) is the interlacement graphs of (Fig. 1, b). In experiments in this paper we will be dealing only with *prime* Gauss diagrams, that is those whose interlacement graph is connected (as the name suggests, if a Gauss diagram is not prime then it can be decomposed into parts which are prime Gauss diagrams, so if one wants to check whether the Gauss diagram is realizable, it can be done individually for each part).

In recent studies (Grinblat and Lopatkin, 2018; Grinblat and Lopatkin, 2020; Biryukov, 2019) very simple realizability conditions were formulated, expressible in terms of the interlacement graphs. However, it was later shown in (Khan et al., 2021b; Khan et al., 2021a) that these conditions are necessary but not sufficient, and explicit counterexamples were found. It is not the only situation when a mistake in mathematical publications is found, but it has created a unique opportunity because these counterexamples are numerous and reasonably small, therefore, can be included in datasets for machine learning.

In this paper we approach the classical concept of Gauss realizability from a new angle, that of machine learning. Specifically, we explore learnability of being realizable by a classical model of multi-layered perceptron using four different encodings of Gauss diagrams. We show that encodings which we denote by IG and SM yield the highest accuracy of trained models. We further show that the accuracy of trained models drops dramatically when tested on those diagrams from (Khan et al., 2021b; Khan et al., 2021a) which are counterexamples to the realizability conditions in (Grinblat and Lopatkin, 2018; Biryukov, 2019; Grinblat and Lopatkin, 2020). Thus, although human mathematicians and AI perceive Gauss diagrams completely differently, it seems that they experience difficulties on the same family of Gauss diagrams.

2 Study design

2.1 Encodings of Gauss diagrams

Before we can use machine learning to recognize properties of Gauss diagrams, we need to represent Gauss diagrams using suitable encodings. We have identified four natural encoding for Gauss diagrams, which we denote OH, SM, PM, IG. They are described below. A priori it is not clear which encoding will perform best in machine learning and be most accurate.

Note that from the mathematical point of view, each of these encodings is a binary matrix, but for the purposes of machine learning, in our experiments we concatenate all rows of this matrix into a one-dimensional binary array.

As you will see, the size of the four encodings is not the same. Let us list the sizes here together, for convenience.

- OH: $n \times 2n = 2n^2$
- SM: $2n \times 2n = 4n^2$
- PM: $n \times n = n^2$
- IG: $n \times n = n^2$

2.1.1 One-hot encoding of Gauss words (OH)

Given a Gauss diagram with n chords, label its chords with elements of $\{1,\dots,n\}$ and represent it as a Gauss word w (recall that a Gauss word is built by travelling around the Gauss diagram and recording what chords are observed). Then encode the symbols of the Gauss word using *one-hot encoding* (see e.g. (Zheng and Casari, 2018)). This means that we build a binary matrix OH of size $n \times 2n$ such that $OH_{ij} = 1$ (or 0) if the j-th symbol in w is (is not) i.

2.1.2 Sparse matrix encoding (SM)

Here again, first use a Gauss word w to represent a given Gauss diagram, but then w is encoded differently. Build a binary matrix SM of size $2n \times 2n$ such that $SM_{ij} = 1$ (or 0) if the i-th symbol in w is equal to (is not equal to) the j-th symbol in w.

2.1.3 Permutation matrix encoding (PM)

Again, given a Gauss diagram, represent it by a Gauss word w. When we use this encoding, we assume that the diagram is a Gauss diagram, that is, satisfies Gauss's condition stated in Section 1, and not merely a chord diagram. Then each symbol $\{1,\dots,n\}$ occurring in w occurs exactly once at an odd-numbered position in w and exactly once at an even-numbered position in w. Build a binary matrix PM of size $n \times n$ such that $PM_{ij} = 1$ (or 0) if the symbol at the i-th odd position in w (that is, the symbol at position $2i-1$) is equal to (is not equal to) the symbol at the j-th even position in w (that is, the symbol at position $2j$). Equivalently, one can say that matrix PM is produced from the $2n \times 2n$ matrix SM of the SM encoding by deleting all even-numbered rows and odd-numbered columns in SM. Yet another way to think of this matrix is to notice that all symbols $\{1,\dots,n\}$ occur in odd-numbered position in w, and all symbols $\{1,\dots,n\}$ occur in even-numbered position in w;

however, the order in which they appear is different; the matrix *PM* is the permutation matrix changing one of these orders into the other.

2.1.4 Interlacement graph encoding (IG)

The adjacency matrix of the interlacement graph of a Gauss diagram is used as an encoding. For a diagram with n chords its size is n^2. A practical way of constructing this matrix is this: given a Gauss diagram, represent it by a Gauss word w; then build a binary matrix IG of size $n \times n$ such that $IG_{ij} = 1$ (or 0) if in w there is an odd (even) number of occurrences of symbol i between the two occurrences of symbol j.

$$\begin{array}{cccccc}
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{array} \qquad
\begin{array}{ccc}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{array}$$

i) iii)

$$\begin{array}{cccccc}
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{array} \qquad
\begin{array}{ccc}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{array}$$

ii) iv)

Table 1: Different encodings of the "trefoil" planar curve shown at Fig.1 a): i) one-hot; ii) sparse matrix; iii) permutation matrix; iv) interlacement graph. Incidentally iii) and iv) are the same for this example. In general PM and IG encodings are different.

2.2 Machine Learning Model

We consider a problem of supervised learning of the binary classifier of realizability property of Gauss diagrams. While there many possible machine learning approaches which could be applied for this task we confine ourselves with the classical model of multilayer perceptron (Rosenblatt, 1958; Rosenblatt, 1961; Rumelhart et al., 1986) and its implementation in WEKA Workbench for Data Mining (Witten et al., 2016). Multi-layer perceptron (MLP) is a kind of a feedforward neural network models which supports supervised learning using *backpropagation* (Rumelhart et al., 1986). It is one of the oldest and well-studied models of machine learning, which is also known to be an *universal approximator* (Cybenko, 1989; Hornik et al., 1989). MLP implementation in WEKA supports *sigmoid* activation function (Han and Moraga, 1995). In the initial experiments we have used default settings of MLP in WEKA Workbench:

`L = 0.3` *(learning rate)*

`M = 0.2` *(momentum rate)*

`N = 500` *(number of epochs to train)*

`V = 0` *(percentage size of validation set to use to terminate training)*

`S = 0` *(seed for Random Number Generator)*

`E = 20` *(threshold for number of consecutive errors to use to terminate training)*

`H = a` *(one hidden layer with (attribs + classes) / 2 many nodes)*

2.3 Data sets

We have used *Gauss-lintel* open source tool (Khan et al., 2021c) to generate various datasets. For that purpose the tool was extended to handle new types of encodings. The main encoding used in *Gauss-lintel* is permutation based (Khan et al., 2021d), so its translation to PM is trivial, while OH, SM and IG encodings introduced in the previous Section, were implemented additionally by translations from PM.

Originally *Gauss-lintel* was used for exhaustive generation of classes of Gauss diagrams satisfying different properties. For the purpose of this work it was extended by the procedure for generation of *random* Gauss diagrams using built-in predicate `random_permutation(+List, -Permutation)` in SWI-Prolog (Wielemaker et al., 2012).

The datasets were generated in Attribute-Relation File Format (ARFF) acceptable by WEKA. We use notation like `IG-9-1000x2` to denote a dataset of Gauss diagrams with 9 chords in IG encoding containing 1000 random realizable and 1000 random non-realizable Gauss diagrams.

3 Experiments and Discussion

We have performed two types of the experiments reported in the following subsections.

3.1 Encodings comparison

We have trained MLP binary classifier models for the sets of random Gauss diagrams of various sizes for all four encodings. We have used 80%/20% random split of the datasets into training/testing datasets.

The results are summarized in the following table

The table presents only weighted average precision, more detailed summaries of all experiments including TP rate, FP rate, Recall, F measure and confusion matrices can be found online together with

Dataset	OH	SM	PM	IG
`8-1000x2`	0.88	0.87	0.85	**0.91**
`9-1000x2`	0.77	0.78	0.76	**0.84**
`10-1000x2`	0.72	0.74	0.70	**0.75**
`10-2000x2`	0.76	**0.80**	0.75	**0.80**
`11-2000x2`	0.67	**0.77**	0.74	0.76
`12-2000x2`	0.67	**0.77**	0.71	0.75

Table 2: The precision of MLP models trained with different Gauss diagrams encodings

all datasets used in this study[1]. The reported results suggest that all encodings yield similar precision of learned models. For smaller sizes 8-10 IG encoding consistently outperforms other encodings, while starting from size 11 SM encoding slightly outperforms IG. One speculative explanation of IG performance might be that the properties of interlacement graphs, which are themselves abstracted codes of the diagrams, determine realiziability of the corresponding diagrams. So having "direct access" to the graph properties/features might be beneficial for machine learning of realizability. As to SM encoding, its good performance might be attributed to the fact that it has largest size of all encodings ($4n^2$ for n-crossing diagrams) and the largest size of the hidden layer of neurons in the corresponding perceptron. This advantage of SM goes though with the much longer training time due to the increased size of the neural network.

The exploration of the wider class of ML models and more rigorous account of the speculative explanations are the topics for our further research here.

3.2 Models behaviour on special counterexamples datasets

In (Khan et al., 2021a) using computational approach and *Gauss-lintel* tool the special sets of Gauss diagrams have been identified. They satisfy all criteria from (Grinblat and Lopatkin, 2018; Grinblat and Lopatkin, 2020; Biryukov, 2019) to be realiziable, but are not in fact realisable. Thus these counterexamples invalidate mentioned criteria and informally speaking have non-trivial reasons to be non-realizable. There are exactly 1, 6, 36, 235 of such diagrams of sizes 9,10,11,12 respectively. The diagrams of sizes 9 and 10 as well as realizability conditions from (Biryukov, 2019) can be seen in Appendix A. In the second series of experiments we compared the accuracy of learned MLP models when tested on random sets of diagrams (using 80%/20% split as before) and on these special sets.

The results are summarised in the following table.

[1]https://doi.org/10.5281/zenodo.5797950

Dataset	R testing	S testing	Misclassified
IG-10-2000x2	0.80	0.00	6 out of 6
IG-11-2000x2	0.76	0.22	28 out of 36
IG-12-2000x2	0.75	0.07	218 out of 235
OH-10-2000x2	0.76	0.50	3 out of 6
OH-11-2000x2	0.67	0.19	29 out of 36
OH-12-2000x2	0.67	0.41	139 out of 235
SM-10-2000x2	0.80	0.50	3 out of 6
SM-11-2000x2	0.77	0.31	25 out of 36
SM-12-2000x2	0.77	0.19	190 out of 235
PM-10-2000x2	0.75	0.00	6 out of 6
PM-11-2000x2	0.74	0.11	32 out of 36
PM-12-2000x2	0.71	0.20	189 out of 235

Table 3: The accuracy of MLP models when tested on random (R) and on special (S) sets of diagrams. The numbers of diagrams misclassified in the latter case are shown in Misclassified column.

Quite surprising outcome of these results is that ML struggles to classify correctly the diagrams from these special sets confirming thereby inherent difficulty of the problem of their recognition. The human mathematicians have made mistakes for these diagrams, ML appears to be following humans. The speculative explanation of such a behaviour of ML might be based **1)** on the observation that these special diagrams are rare and **2)** the majority of the diagrams have "simpler reasons" for non-realizability. So, if trained on random diagrams, ML can learn the simpler conditions covering majority of the diagrams, but might have no chance to learn more complicated conditions for the rare special diagrams due to not seeing such diagrams during training process.

To provide some empirical evidence for such reasoning we have performed one more experiment. We have split the set S of 235 special diagrams of size 12 into two subsets S_1 and S_2 of sizes 100 and 135, respectively. S_1 is then used to replace 100 non-realizable diagrams in the dataset `12-2000x2`. Then we used such modified dataset `12-2000x2M` for training. The resulting model was tested on random diagrams (using 80/20 split) and on diagrams from S_2. The first testing returned average precision 0.73 (slight drop from original 0.75 for unmodified dataset), but the testing on S_2 yielded the rise of precision to 0.62 from the original 0.07 (for unmodified dataset and testing on S). So, it appears that indeed seeing the special diagrams during the training helps to increase considerably the chances of correct classification of other diagrams from this set.

We believe that more rigorous account of such explanations can be given by the application of the recent approach to the learnability of relational properties using model counting, proposed in (Usman et al., 2020). This is a topic of our ongoing research.

One of the limitations of the presented study is that the only one model of ML has been used. We have tested quickly all implemented in WEKA classifiers in default settings on the instances of the realizability problem above and the preliminary results suggest that only Random Forest (RF)(Breiman, 2001) has comparable with MLP precision of learned models. Interestingly, in the observed cases RF also struggles to recognise special diagrams.

Dataset	R testing	S testing	Misclassified
IG-12-2000x2	0.84	0.03	227 out of 235

Table 4: The accuracy of Random Forest model when tested on random (R) and on special (S) sets of diagrams. The numbers of diagrams misclassified in the latter case are shown in Misclassified column.

Systematic exploration of the experiments with alternative ML models will be presented in the extended version of this paper.

4 Conclusion

In this paper we have presented our initial experiments with machine learning applied to the classical problem of computational topology, that is a recognition of realizable Gauss diagrams. We have experimented with four various encodings and identified the encodings enabling the highest precision of learned models. We have discovered an interesting phenomenon where trained ML models drop their performance dramatically when tested on special recently discovered sets of diagrams, which are counterexamples to the published relaiziability criteria. We proposed some speculative explanations and outlined further research directions to get more rigorous account of the observed phenomena.

Acknowledgments

This work was supported by the Leverhulme Trust Research Project Grant RPG-2019-313.

REFERENCES

Biryukov, O. N. (2019). Parity conditions for realizability of Gauss diagrams. *Journal of Knot Theory and Its Ramifications*, 28(01):1950015.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314.

Dehn, M. (1936). Uber kombinatorische topologie. *Acta Math.*, 67:123–168.

Dowker, C. and Thistlethwaite, M. B. (1983). Classification of knot projections. *Topology and its Applications*, 16(1):19 – 31.

Francis, G. K. (1969). Null genus realizability criterion for abstract intersection sequences. *Journal of Combinatorial Theory*, 7(4):331 – 341.

Gauss, C. (1900). Werke.

Grinblat, A. and Lopatkin, V. (2018). On realizability of Gauss diagrams and constructions of meanders. arxiv:1808.08542.

Grinblat, A. and Lopatkin, V. (2020). On realizabilty of Gauss diagrams and constructions of meanders. *Journal of Knot Theory and Its Ramifications*, 29(05):2050031.

Han, J. and Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In J., M. and F., S., editors, *From Natural to Artificial Neural Computation, LNCS, vol 930*, pages 195–201. Springer Berlin Heidelberg.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Khan, A., Lisitsa, A., Lopatkin, V., and Vernitski, A. (2021a). Circle graphs (chord interlacement graphs) of Gauss diagrams: Descriptions of realizable gauss diagrams, algorithms, enumeration. arxiv:2108.02873.

Khan, A., Lisitsa, A., and Vernitski, A. (2021b). Experimental mathematics approach to Gauss diagrams realizability. arxiv:2103.02102.

Khan, A., Lisitsa, A., and Vernitski, A. (2021c). Gauss-lint algorithms suite for Gauss diagrams generation and analysis. Zenodo, 10.5281/zenodo.4574590, https://doi.org/10.5281/zenodo.4574590.

Khan, A., Lisitsa, A., and Vernitski, A. (2021d). Gauss-lintel, an algorithm suite for exploring chord diagrams. In Kamareddine, F. and Sacerdoti Coen, C., editors, *Intelligent Computer Mathematics*, pages 197–202, Cham. Springer International Publishing.

Kurlin, V. (2008). Gauss paragraphs of classical links and a characterization of virtual link groups. *Mathematical Proceedings of the Cambridge Philosophical Society*, 145(1):129–140.

Lovász, L. and Marx, M. L. (1976). A forbidden substructure characterization of Gauss codes. *Bull. Amer. Math. Soc.*, 82(1):121–122.

Marx, M. L. (1969). The Gauss realizability problem. *Proceedings of the American Mathematical Society*, 22(3):610–613.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rosenblatt, F. (1961). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books.

Rosenstiehl, P. (1976). Solution algébrique du problème de Gauss sur la permutation des points d'intersection d'une ou plusieurs courbes fermées du plan. *C.R. Acad. Sci.*, t. 283, série A:551–553.

Rosenstiehl, P. and Tarjan, R. E. (1984). Gauss codes, planar hamiltonian graphs, and stack-sortable permutations. *J. Algorithms*, 5(3):375–390.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and Mcclelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA.

Shtylla, B., Traldi, L., and Zulli, L. (2009). On the realization of double occurrence words. *Discret. Math.*, 309(6):1769–1773.

Usman, M., Wang, W., Vasic, M., Wang, K., Vikalo, H., and Khurshid, S. (2020). A study of the learnability of relational properties: model counting meets machine learning (mcml). *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*.

Wielemaker, J., Schrijvers, T., Triska, M., and Lager, T. (2012). SWI-Prolog. *Theory and Practice of Logic Programming*, 12(1-2):67–96.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *The WEKA Workbench. Online Appendix for Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 4th edition.

Zheng, A. and Casari (2018). *Feature Engineering for Machine Learning*. O'Reilly Media, Inc.

Appendix A

We present the realiziability conditions for prime Gauss diagrams from (Biryukov, 2019), equivalent to those from (Grinblat and Lopatkin, 2018; Grinblat and Lopatkin, 2020) and formulated in terms of interlacement graphs. According to these conditions, a prime Gauss diagram g is realizable if and only if in its interlacement graph:

- each vertex has *even* degree (original Gauss condition)
- each pair of non-neighbouring vertices has an *even* number of common neighbours (possibly, zero);
- for any three pairwise connected vertices $a, b, c \in V$ the sum of the number of vertices adjacent to a, but not adjacent to b nor c, and the number of vertices adjacent to b and c, but not adjacent to a, is even.

The following figures present counterexamples to these conditions (n=9,10), that is diagrams which satisfy the conditions, but are not realizable.

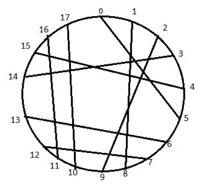

Figure 2: A counterexample diagram, n=9 (Khan et al., 2021b)

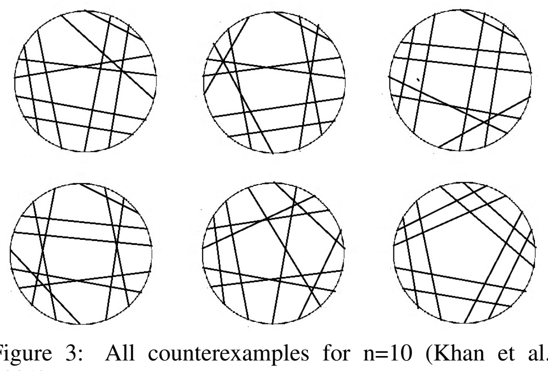

Figure 3: All counterexamples for n=10 (Khan et al., 2021b)