VN-Solver: Vision-based Neural Solver for Combinatorial Optimization over Graphs

Mina Samizadeh
University of Delaware
Newark, Delaware, USA
minasmz@udel.edu

Guangmo Tong
University of Delaware
Newark, Delaware, USA
amotong@udel.edu

ABSTRACT

Data-driven approaches have been proven effective in solving combinatorial optimization problems over graphs such as the traveling salesman problems and the vehicle routing problem. The rationale behind such methods is that the input instances may follow distributions with salient patterns that can be leveraged to overcome the worst-case computational hardness. For optimization problems over graphs, the common practice of neural combinatorial solvers consumes the inputs in the form of adjacency matrices. In this paper, we explore a vision-based method that is conceptually novel: can neural models solve graph optimization problems by *taking a look at the graph pattern*? Our results suggest that the performance of such vision-based methods is not only non-trivial but also comparable to the state-of-the-art matrix-based methods, which opens a new avenue for developing data-driven optimization solvers.

CCS CONCEPTS

• **Applied computing** → **Multi-criterion optimization and decision-making**.

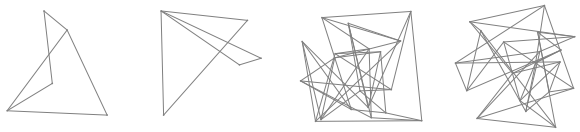
KEYWORDS

Neural Combinatorial Optimization, Computer Vision

ACM Reference Format:
Mina Samizadeh and Guangmo Tong. 2023. VN-Solver: Vision-based Neural Solver for Combinatorial Optimization over Graphs. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, Birmingham, UK, 5 pages. https://doi.org/10.1145/3583780.3615156

1 INTRODUCTION

Combinatorial optimization problems are not only of great theoretical interest but also central to various application domains [13]. Traditional approaches suffer from their forbidding execution time and the need for hand-crafted algorithmic rules [3]. Recent years have witnessed a surge in the development of neural combinatorial solvers, aiming to efficiently solve combinatorial optimization problems through machine learning techniques [1]. The premise behind such methodologies is that the input-solution pairs in practical applications often exhibit certain data-dependent distributions, which, once successfully learned, can be leveraged to circumvent the worst-case NP-hardness. As the inherent structure of many problems is often relational [18], it is of paramount interest to examine the potential of neural solvers for addressing optimization problems on graphs. We will particularly focus on deterministic combinatorial optimization problems in this paper.



(a) (b) (c) (d)

Figure 1: (a): a simple Hamiltonian graph; (b): a simple non-Hamiltonian graph; (c) and (d): visualizations that are relatively complicated.

The state-of-the-art neural solvers are largely matrix-based, i.e., reasoning over the adjacency matrix by using deep neural networks [4]. In direct contrast to the matrix-based methods, this paper explores the so-called vision-based methods that consume graph visualizations as inputs. Such methods are desired when the input of the problem is given as images (e.g., maps of the traveling salesman problem [14]), as it does not require us to convert the images to matrices. More importantly, we are motivated by the fact that for humans, visualizations can be much more intuitive than the adjacency matrix for certain instances. Taking the Hamiltonian cycle problem as an example, let us consider Figs. 1a and 1b with the following two adjacency matrices

$$A_a = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } A_b = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

It is obvious at a glance that Fig. 1a is Hamiltonian while Fig. 1b is not, but it would take arguably more time to figure out the same results if we were checking the adjacency matrices. There of course exist instances that cannot be easily recognized by humans – for example, Figs. 1c and 1d, which are Hamiltonian and in fact correspond to the same graph. However, since deep learning methods have demonstrated visual perception ability better than humans [16, 21], it is possible that instances like Fig. 1c can be successfully handled by image classifiers like ResNet [17] or VIT [20], without consulting the adjacency matrices. In this paper, we explore the feasibility of such methods.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3615156

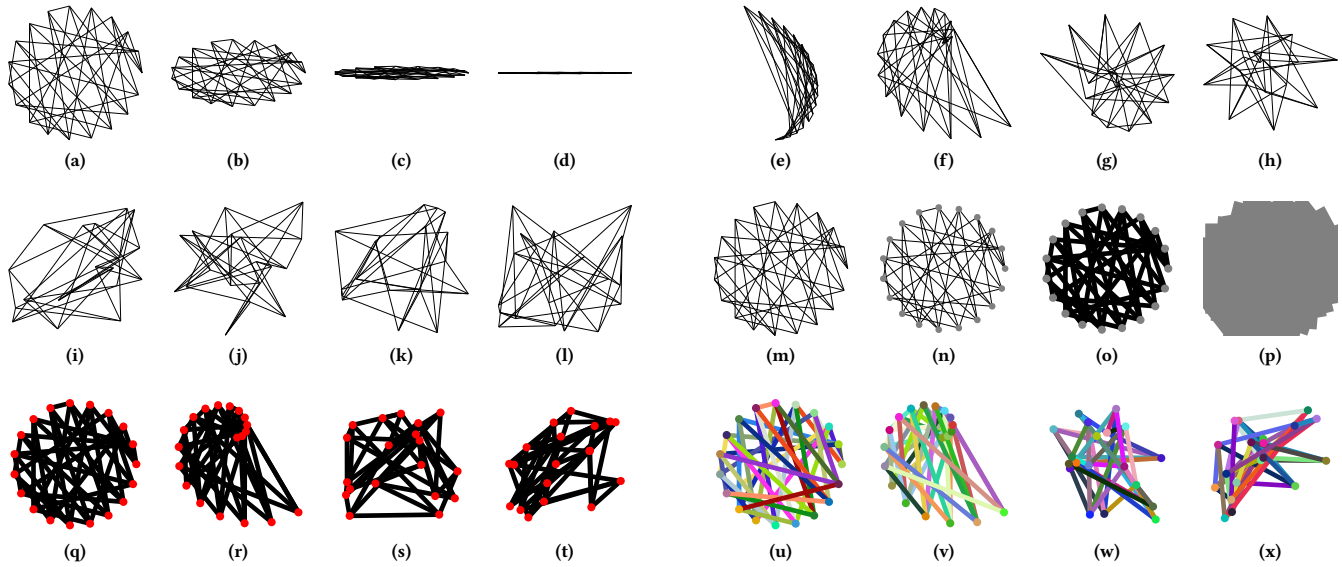


Figure 2: Visualizations of different layouts, node sizes, edge thicknesses, and coloring schemes.

Contributions. We present a conceptually simple framework called the vision-based method for solving deterministic combinatorial optimization problems over graphs. The proposed framework consists of three modules: *graph embedding*, *image generation*, and *image classification*. The first module decides how to embed a graph in Euclidean space; the second module visualizes the embedding as an image of pixels; the third module decides if the input is a true instance. We adopt ResNet for image classification and discuss possible choices for graph embedding and image generation. Our experiments promisingly show that vision-based methods are effective with statistical significance and no less powerful than the state-of-the-art matrix-based methods for the Hamiltonian cycle problem. Our experiments also provide insights into the role of the visualization modules in determining the framework's efficacy. For example, we observe that structured visualizations are evidently better than random visualizations. The code is available at [1].

2 VN-SOLVER: A VISION-BASED METHOD

A deterministic graph optimization problem is specified by a function F that maps each graph $G = (V, E)$ to true or false. For example, in planarity testing [11], $F(G) = 1$ if and only if G is a planar graph. Some of such problems can be readily solved (e.g., connectivity testing [8]), while others are computationally hard under common complexity assumptions (e.g., Hamiltonian cycle or path [15]). From the perspective of statistical learning, a neural solver seeks to learn F using empirical evidence, i.e., pairs of $(G, F(G))$, with the goal of minimizing the generalization error

$$\mathcal{L}(\tilde{F}, \mathcal{D}) = \mathbb{E}_{G\sim\mathcal{D}}\left[l(F(G), \tilde{F}(G))\right], \tag{1}$$

where $\tilde{F}$ is the learned function, $\mathcal{D}$ is an unknown distribution over graphs, and, l is a certain loss function.

Matrix-based methods take the adjacency matrix A_G of G as the input and often process A_G through various neural architectures where the last layer gives a distribution over $\{0, 1\}$ suggesting if the input graph is a true instance. Our vision-based method is conceptually different and consists of three simple steps:

graph embedding: $F_{\text{em}} : V \to \mathbb{R}^2$
image generation: $F_{\text{graph}} : \mathbb{R}^{2\times|V|} \to \mathbb{R}^{3\times \text{dim}_x \times \text{dim}_y}$
image classification: $F_{\text{class}} : \mathbb{R}^{3\times \text{dim}_x \times \text{dim}_y} \to \{0, 1\}$.

F_{em} maps each node v to a point $(v_x, v_y) \in \mathbb{R}^2$ in the 2D Euclidean space, and each edge (u, v) corresponds to the line segment with edge points (u_x, u_y) and (v_x, v_y). Based on the embedding (i.e., a collection of points in the 2D space), F_{graph} generates an image of size $\text{dim}_x \times \text{dim}_y$ in RGB pixels. Finally, a classifier F_{class} decides if the input is a true instance. We denote such a vision-based neural solver as VN-Solver. The recent advance in computer vision has provided us with a great collection of tools for image classification, and we will leverage them for the third step. In particular, we adopt the celebrated ResNet model [10] in our experiments. While image classification is not a burden, the methods for graph embedding and image generation are open to different design principles.

For graph embedding, one can imagine that some methods will definitely not work – for example, putting all the nodes on a straight line will make the edges not recognizable. We primarily focus on two principled methods: circular layout [2] and spiral layout [5]. The circular layout puts the nodes uniformly on an eclipse $x^2/a + y^2/b = 1$, where the ratio between a and b controls the shape. Figs. 2a-2d are examples with different ratios, where we can sense that an extreme ratio (e.g., Fig. 2d) may not be useful for recognizing graph properties (which we verify in experiments). For each $v_i \in V = \{v_1, ..., v_n\}$, the spiral layout, dated back to the spiral of Archimedes [7], assigns its coordinate as $(i \cdot \cos(i \cdot r), i \cdot \sin(i \cdot r))$, where $r \in \mathbb{R}$ is the offset parameter. Examples with different offsets can be found in Figs. 2e-2h. One can even use random embeddings where the coordinates of the nodes are uniformly random within, for example, $[0, 1] \times [0, 1]$ – Figs. 2i-2l.

In generating an image for a given embedding, we plot nodes as solid circles and edges as solid line segments. The size is 224 by 224 in pixels, which is aligned with ResNet [10]. The primary

[1] https://github.com/minasmz/VN-Solver

factors we consider are the size of the node circles and the thickness of the edges segments (Figs. 2m-2p). In addition to gray images (e.g., Figs. 2a-2p), the images may be colored in various ways. For example, in the *uniform-color* scheme, the nodes are in one color and the edges are in another color (e.g., Figs. 2q-2t), while in the *random-color* scheme, we randomly pick one color for each entity (i.e., node or edge) – Figs. 2u-2x. While human perception may not be very sensitive to these factors, images with different node circle sizes can be very different from each other from the perspective of machine learning models. In fact, all the visualizations in Fig. 2 are associated with the same underlying graph. It would be interesting if one can observe that some of them are more informative than others in deciding certain graph properties.

3 EMPIRICAL STUDIES

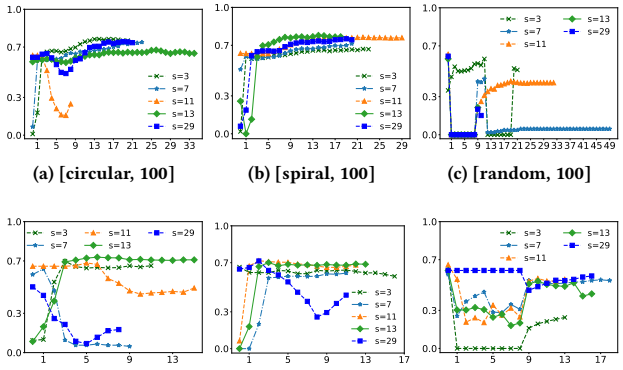
In this paper, we focus on the Hamiltonian cycle problem. Based on the House of Graphs [6], we create two datasets, Small and Large. The Small dataset contains 4,115 graphs with 4 to 20 nodes, where 2,277 are Hamiltonian and 1,838 are non-Hamiltonian; the Large dataset consists of 13,192 graphs with 20 to 50 nodes, where 7,453 are Hamiltonian and 5,739 are non-Hamiltonian. We examine the following methods in our experiments.

- **VN-Solver**. Unless otherwise specified, we have $a/b = 1$ for the circular layout and $r = 0.3$ for the spiral layout; for all visualizations, each node occupies 2×2 in pixels and the thickness of the line segments is 2 in pixels. We train ResNet using Adam with a learning rate of 0.001 and an exponential learning rate decay of 0.09. We set the maximum epoch as 200 and stop when the F1 score does not increase for 8 epochs.
- **Graphormer**. We consider graph transformer [19], which is one of the state-of-the-art matrix-based methods. We adopt the graphormer-slim architecture composed of 12 attention layers and each layer has 8 heads. The training is done via Adam with a learning rate of 0.001, and a decay of 0.01. The early stopping is the same as that in the VN-Solver.
- **Naive-Bayesian**. This is a feature-oblivious Bayesian method that makes the prediction based on the prior distribution over $\{0, 1\}$ estimated from the training samples. For example, the prediction is uniformly at random from $\{0, 1\}$, if half of the training samples are true instances.

For each learning-based method, the training-testing process is repeated five times, and we report the average performance in terms of common metrics together with the standard deviation. The testing size is fixed to be 500; we experiment with sample sizes for training, where 80 (resp., 20) percent of the samples are used for training (resp., validation).

3.1 Observations

Feasibility. The main results can be found in Table 1. In addition, the generalization performance of VN-Solver after each training epoch is given in Fig. 3, where results for circular and spiral layouts suggest a reasonable learning process, while the random visualizations are not very informative. From these results, it is clear that VN-Solver can perform better when more training data is given or more training epochs are used. This confirms that VN-Solver indeed can learn from data towards solving the Hamiltonian cycle problem, which gives the first piece of evidence supporting the feasibility of vision-based neural combinatorial solvers.



(a) [circular, 100] (b) [spiral, 100] (c) [random, 100]

(d) [circular, 200] (e) [spiral, 200] (f) [random, 200]

Figure 3: The F1 score of VN-Solver after each epoch on dataset Small. Each sub-float is labeled by the layout method and the sample size in training, and it shows the results under five different seed values. The early stopping ensures that a decent checkpoint (not necessarily the last epoch in the figure) is selected as the final model.

Effectiveness. Promisingly, the results in Table 1 also suggest that VN-Solver is not only feasible but also no less effective than the state-of-the-art matrix-based method in most cases. In particular, it is comparable to Graphormer when the visualizations are gray, and it outperforms Graphormer under the uniform-color scheme. Notably, the results of VN-Solver and Graphormer are statistically significant, as they clearly outperform Naive-Bayesian.

Coloring scheme. One interesting observation is that colored visualization can consistently lead to better performance than gray ones do. This is especially true when the training size is not very large: with 200 training samples on the Large dataset, switching the coloring scheme from gray to uniform can increase the F1 score from 0.61 to 0.90 under the circular layout. One plausible reason is that the graph structure is more salient to image classifiers like ResNet when the nodes and edges are colored. For similar reasons, we see that switching from uniform-color scheme to random-color scheme will slightly decrease the performance.

Other factors. Table 2(a) shows the results of VN-Solver under the circular layout with different node sizes and edge thicknesses. In general, we see that the performance becomes better when the nodes and edges are visualized with larger sizes. Of course, an extreme size can make the graph structure not recognizable – for example, $(x, y) = (100, 100)$ as shown in Fig. 2p. Finally, we observe that the performance is also sensitive to the parameters of the layouts, as shown in Table 2(c) and 2(d). For the circular layout, increasing the ratio tends to decrease the performance. For the spiral layout, visualizations with a small spiral factor r can be better recognized by ResNet, which is intuitive since, for example, Fig. 2e appears to be more structured than Fig. 2h.

4 FURTHER DISCUSSIONS

We close our paper by listing the following research issues that we believe deserve future investigations in depth.

- **Wider applicability.** It is of interest to examine the feasibility of vision-based methods for other deterministic graph

Table 1: Main results. Top F1 scores are highlighted. The sample size for training is from $\{100, 200, 1000\}$

Dataset: Small			100			200			1000		
			AUC	Accuracy	F1	AUC	Accuracy	F1	AUC	Accuracy	F1
VN-Solver	Gray	Circular	0.55 ± 0.06	0.45 ± 0.03	0.62 ± 0.03	0.43 ± 0.02	0.44 ± 0.02	0.61 ± 0.02	0.86 ± 0.09	0.79 ± 0.09	**0.78** ± 0.08
		Spiral	0.59 ± 0.15	0.52 ± 0.13	0.63 ± 0.04	0.65 ± 0.14	0.54 ± 0.15	0.65 ± 0.06	0.84 ± 0.02	0.78 ± 0.02	0.76 ± 0.02
		Random	0.50 ± 0.01	0.49 ± 0.06	0.50 ± 0.28	0.51 ± 0.02	0.44 ± 0.02	0.61 ± 0.02	0.52 ± 0.05	0.49 ± 0.07	0.37 ± 0.34
	Uniform color	Circular	0.69 ± 0.13	0.61 ± 0.09	**0.63** ± 0.09	0.75 ± 0.14	0.68 ± 0.12	**0.69** ± 0.04	0.93 ± 0.02	0.85 ± 0.02	**0.83** ± 0.03
		Spiral	0.70 ± 0.07	0.62 ± 0.05	**0.65** ± 0.05	0.78 ± 0.06	0.71 ± 0.06	**0.72** ± 0.05	0.86 ± 0.04	0.78 ± 0.06	0.76 ± 0.07
		Random	0.52 ± 0.03	0.45 ± 0.0	0.62 ± 0.00	0.51 ± 0.02	0.43 ± 0.00	0.60 ± 0.00	0.51 ± 0.02	0.48 ± 0.00	0.65 ± 0.00
	Random color	Circular	0.63 ± 0.09	0.54 ± 0.11	0.61 ± 0.03	0.65 ± 0.11	0.56 ± 0.11	0.64 ± 0.04	0.9 ± 0.02	0.83 ± 0.03	**0.81** ± 0.02
		Spiral	0.73 ± 0.06	0.62 ± 0.08	**0.64** ± 0.04	0.64 ± 0.15	0.6 ± 0.1	**0.65** ± 0.03	0.85 ± 0.02	0.73 ± 0.08	0.74 ± 0.04
Graphormer			0.68 ± 0.02	0.60 ± 0.03	0.60 ± 0.14	0.70 ± 0.03	0.63 ± 0.01	0.64 ± 0.11	0.73 ± 0.03	0.66 ± 0.05	0.65 ± 0.18
Naive-Bayesian			0.50 ± 0.02	0.50 ± 0.02	0.54 ± 0.03	0.50 ± 0.02	0.50 ± 0.02	0.55 ± 0.02	0.50 ± 0.02	0.50 ± 0.01	0.55 ± 0.01

Dataset: Large			100			200			1000		
			AUC	Accuracy	F1	AUC	Accuracy	F1	AUC	Accuracy	F1
VN-Solver	Gray	Circular	0.58 ± 0.33	0.45 ± 0.02	0.62 ± 0.02	0.44 ± 0.39	0.44 ± 0.02	0.61 ± 0.02	0.96 ± 0.03	0.92 ± 0.03	0.92 ± 0.04
		Spiral	0.51 ± 0.19	0.53 ± 0.12	0.5 ± 0.28	0.72 ± 0.24	0.63 ± 0.23	0.72 ± 0.15	0.98 ± 0.02	0.95 ± 0.02	**0.94** ± 0.02
		Random	0.51 ± 0.17	0.48 ± 0.06	0.37 ± 0.34	0.54 ± 0.14	0.57 ± 0.09	0.26 ± 0.36	0.81 ± 0.02	0.72 ± 0.06	0.72 ± 0.03
	Uniform color	Circular	0.83 ± 0.08	0.72 ± 0.16	**0.74** ± 0.10	0.93 ± 0.04	0.90 ± 0.07	**0.90** ± 0.08	0.98 ± 0.01	0.94 ± 0.02	**0.94** ± 0.03
		Spiral	0.86 ± 0.04	0.78 ± 0.03	**0.75** ± 0.07	0.91 ± 0.11	0.81 ± 0.18	**0.83** ± 0.12	0.98 ± 0.01	0.95 ± 0.01	**0.95** ± 0.02
		Random	0.53 ± 0.03	0.48 ± 0.02	0.51 ± 0.29	0.51 ± 0.06	0.47 ± 0.00	0.64 ± 0.00	0.47 ± 0.08	0.41 ± 0.02	0.58 ± 0.01
	Random color	Circular	0.59 ± 0.18	0.54 ± 0.09	0.63 ± 0.03	0.87 ± 0.12	0.77 ± 0.17	0.80 ± 0.10	0.97 ± 0.01	0.91 ± 0.02	0.91 ± 0.03
		Spiral	0.77 ± 0.13	0.58 ± 0.16	0.64 ± 0.09	0.82 ± 0.27	0.79 ± 0.17	0.81 ± 0.10	0.98 ± 0.01	0.94 ± 0.02	0.93 ± 0.03
Graphormer			0.83 ± 0.01	0.76 ± 0.08	**0.74** ± 0.12	0.83 ± 0.02	0.83 ± 0.02	**0.83** ± 0.02	0.93 ± 0.01	0.92 ± 0.01	0.92 ± 0.01
Naive-Bayesian			0.49 ± 0.02	0.51 ± 0.02	0.51 ± 0.03	0.50 ± 0.01	0.50 ± 0.01	0.54 ± 0.04	0.5 ± 0.02	0.49 ± 0.01	0.51 ± 0.03

Table 2: Additional results of VN-Solver. The results in subtables (a)-(c) are F1 scores with 200 samples for training under the uniform-color scheme.

Table 2(a): Results of the circular layout, where each cell (x, y) means that each node (resp., edge) is scaled by a factor of x (resp., y).

	y=0.01	y=0.1	y=1	y=10	y=100
x=0.01	0.63 ± 0.03	0.7 ± 0.04	0.71 ± 0.05	0.73 ± 0.04	0.68 ± 0.02
x=0.5	0.64 ± 0.04	0.69 ± 0.04	0.72 ± 0.02	0.74 ± 0.04	0.66 ± 0.06
x=5	0.64 ± 0.06	0.71 ± 0.03	0.74 ± 0.03	0.74 ± 0.02	0.66 ± 0.06
x=100	0.65 ± 0.05	0.72 ± 0.03	0.75 ± 0.06	0.77 ± 0.02	0.66 ± 0.05

Table 2(b): Circular layout with different ratios of a/b.

	$a/b = 0.001$	$a/b = 0.01$	$a/b = 0.1$	$a/b = 1$
Small	0.62 ± 0.02	0.63 ± 0.13	0.63 ± 0.05	0.69 ± 0.04
Large	0.67 ± 0.11	0.77 ± 0.15	0.78 ± 0.07	0.90 ± 0.08

Table 2(c): Spiral layout with different r.

	$r = 0.1$	$r = 0.3$	$r = 0.5$	$r = 1$
Small	0.73 ± 0.02	0.66 ± 0.06	0.66 ± 0.02	0.63 ± 0.06
Large	0.85 ± 0.04	0.89 ± 0.02	0.86 ± 0.05	0.87 ± 0.05

optimization problems. For example, can neural solvers effectively decide graph isomorphism based on the visualizations?

- **Advanced embedding methods.** What is the best graph embedding method with respect to solving a given graph optimization problem? This question is quite open and has to be discussed on a case-by-case basis. In addition to classic methods like circular and spiral layouts, one can even use generative models to design learnable embedding methods.

(a) Embedding (b) Style (c) Visualization

Figure 4: An example of generating coloring schemes (c) by giving an embedding (a) a style (b) using neural networks [9]. Is there a style that is useful in terms of deciding the Hamiltonian cycle?

In another issue, besides the 2D Euclidean space, one can embed graphs in any other metric space, which creates new research opportunities.

- **The best visualization style.** Following the investigations on the coloring scheme, node size, and edge thickness, we are wondering how to design better visualization styles. For example, while the random-color scheme does not exhibit extra benefits for our cases (i.e., the Hamiltonian cycle problem), images of perceptual styles with possible semantic meanings might be better recognized by machine learning models thereby offering better generalization performance. Immediately and somehow surprisingly, techniques for image style analysis [9, 12] become a potential component of neural solvers for graph optimization (See Fig. 4 for an example).

ACKNOWLEDGMENTS

This project is supported in part by National Science Foundation under Award IIS-2144285.

REFERENCES

[1] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. 2021. Machine learning for combinatorial optimization: a methodological tour dhorizon. *European Journal of Operational Research* 290, 2 (2021), 405–421.

[2] Swati A Bhavsar, Varsha H Patil, and Aboli H Patil. 2022. Graph partitioning and visualization in graph mining: a survey. *Multimedia Tools and Applications* 81, 30 (2022), 43315–43356.

[3] Ilhem Boussad, Julien Lepagnot, and Patrick Siarry. 2013. A survey on optimization metaheuristics. *Information sciences* 237 (2013), 82–117.

[4] Quentin Cappart, Didier Chetelat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Velickovic. 2021. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544* (2021).

[5] John V Carlis and Joseph A Konstan. 1998. Interactive visualization of serial periodic data. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*. 29–38.

[6] Kris Coolsaet, Sven D'hondt, and Jan Goedgebeur. 2023. House of Graphs 2.0: A database of interesting graphs and more. *Discrete Applied Mathematics* 325 (2023), 97–107.

[7] Arthur Czwalina et al. 1922. *Über spiralen.* Number 201. Akademische verlagsgesellschaft mbh.

[8] Shimon Even and R Endre Tarjan. 1975. Network flow and testing graph connectivity. *SIAM journal on computing* 4, 4 (1975), 507–518.

[9] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2414–2423.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[11] John Hopcroft and Robert Tarjan. 1974. Efficient planarity testing. *Journal of the ACM (JACM)* 21, 4 (1974), 549–568.

[12] Sergey Karayev, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller. 2013. Recognizing image style. *arXiv preprint arXiv:1311.3715* (2013).

[13] Bernhard H Korte, Jens Vygen, B Korte, and J Vygen. 2011. *Combinatorial optimization*. Vol. 1. Springer.

[14] Diego Perez, Julian Togelius, Spyridon Samothrakis, Philipp Rohlfshagen, and Simon M Lucas. 2013. Automated map generation for the physical traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 18, 5 (2013), 708–720.

[15] M Sohel Rahman and Mohammad Kaykobad. 2005. On Hamiltonian cycles and Hamiltonian paths. *Inform. Process. Lett.* 94, 1 (2005), 37–41.

[16] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

[17] Sasha Targ, Diogo Almeida, and Kevin Lyman. 2016. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029* (2016).

[18] Natalia Vesselinova, Rebecca Steinert, Daniel F Perez-Ramirez, and Magnus Boman. 2020. Learning combinatorial optimization on graphs: A survey with applications to networking. *IEEE Access* 8 (2020), 120388–120416.

[19] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems* 34 (2021), 28877–28888.

[20] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. 2021. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*. 558–567.

[21] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. 2023. Object detection in 20 years: A survey. *Proc. IEEE* (2023).