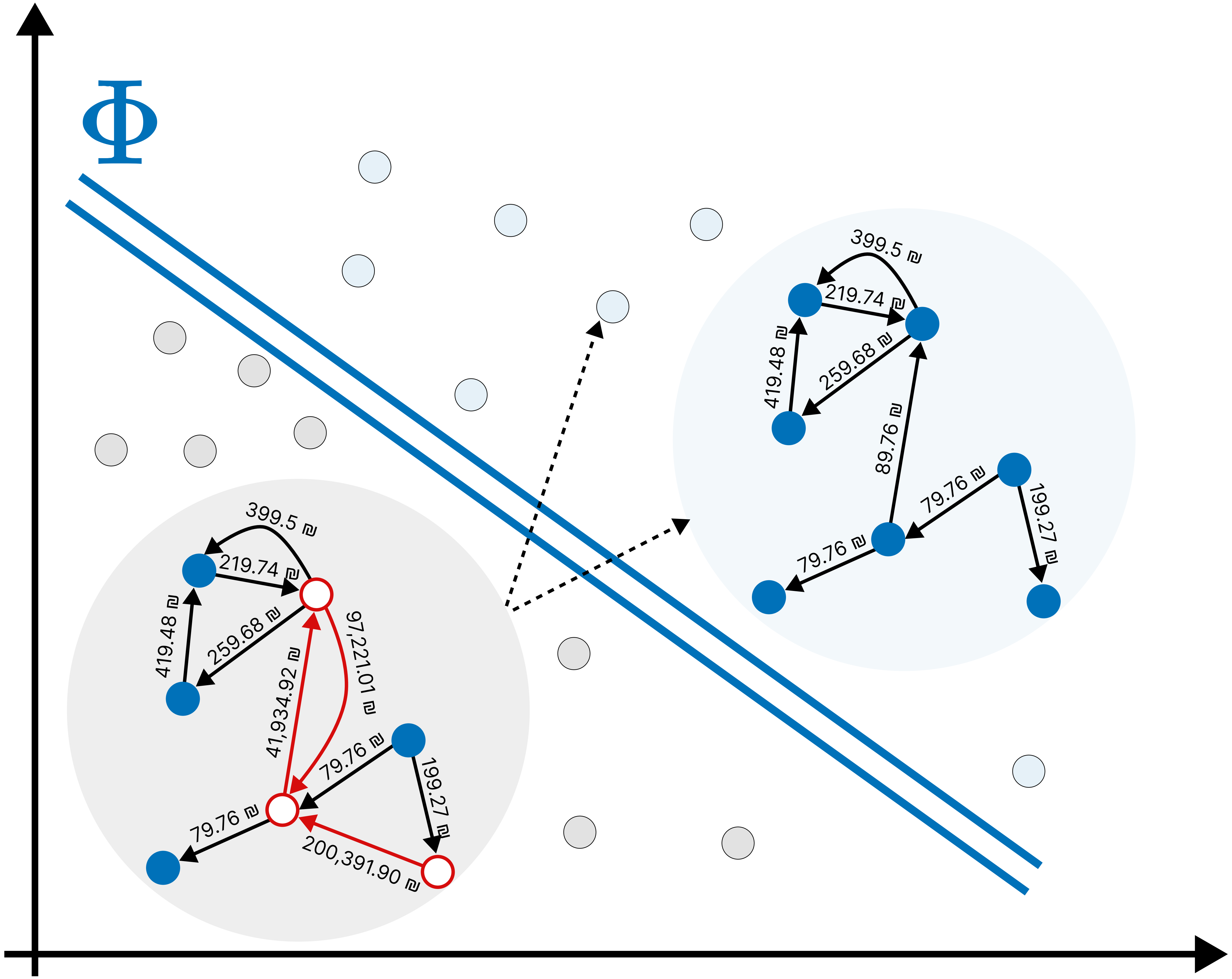
Robust Stochastic Graph Generator for Counterfactual Explanations

Bardh Prenkaj

Presented at The 38th Annual AAAI Conference on Artificial Intelligence

Graph Counterfactual Explainability (GCE)



Problems with GCE

- SoA is generally **constrained** to the **input data** (search-based GCE) and relies on **learned perturbation masks** (learning-based GCE)
- Defaulting to factual-based explainers falters when dual classes clash (e.g., **acyclic vs cyclic graphs**)
- Crossing the decision boundary isn't enough; one must be **close to** the original instance

What's been done until now...

- Learning-based GCE [1-5]:
  1) generate masks of relevant features given a graph G;
  2) combine this mask with G to derive G';
  3) feed G' to the oracle Φ and update the mask

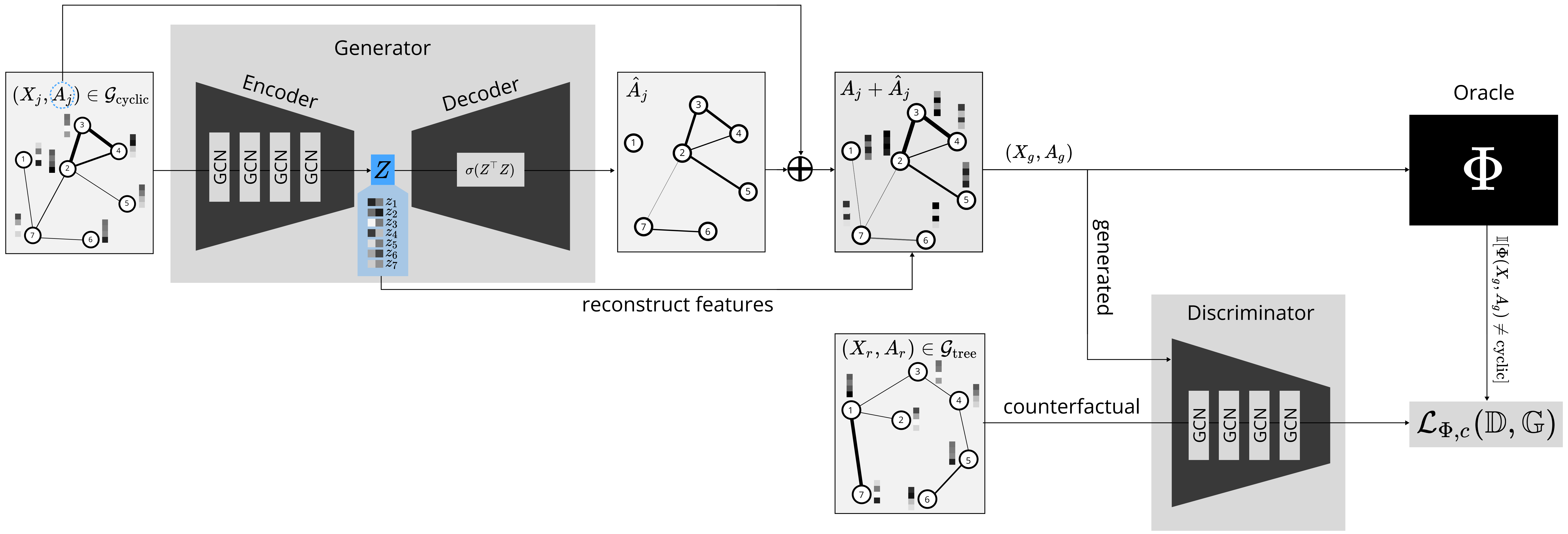
- CLEAR [5] uses a VAE to encode graphs into a latent representation which, at inference, is used to generate complete stochastic graphs

- G-CounteRGAN [6,7] relies on 2D convolutions on the adjacency matrix of graphs

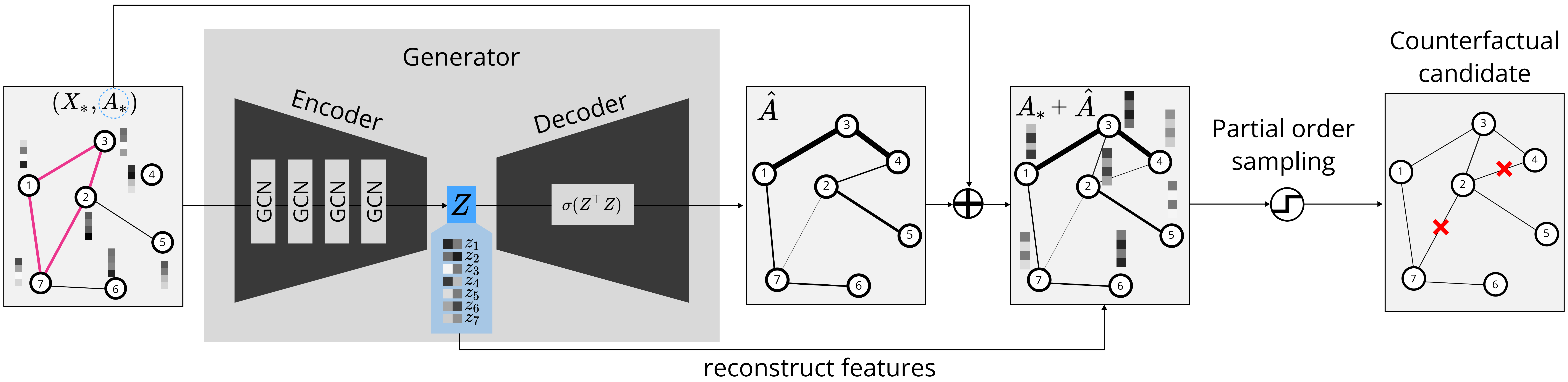
Intuition

- Using a generative approach possibly a GAN allows having brand new in-distribution counterfactuals examples
- We’ll exploit the generator to engender counterfactual candidates
- Use the discriminator to guide the generator in learning how to cross the decision boundary

A closer look at RSGG-CE



RSGG-CE (inference)



RSGG-CE (inference)

Algorithm 1: Partial order sampling to produce a counterfactual.

Require: $G_* = (X_*, A_*), \mathbb{G} : \mathcal{G} \to \mathcal{G}, \Phi,$

$\hat{X}_*, A_* + \hat{A}_* = \mathbb{G}(X_*, A_*)$
$X_g, A_g \leftarrow \hat{X}_*, A_* + \hat{A}_*$
$\mathcal{P} \leftarrow \texttt{partial\_order}(A_*)$
$A' \leftarrow 0^{n \times n}$
for $\mathbb{O} \in \mathcal{P}$ **do**
    for $e = (u, v) \in \mathbb{O}.\mathcal{E}$ **do**
        $A'[u, v] \leftarrow \texttt{sample}(e, A_g[u, v])$
        if $\mathbb{O}.o \wedge \Phi(X_g, A') \neq \Phi(X_*, A_*)$ **then**
            return (X_g, A')
        end if
    end for
end for
return (X_*, A_*)

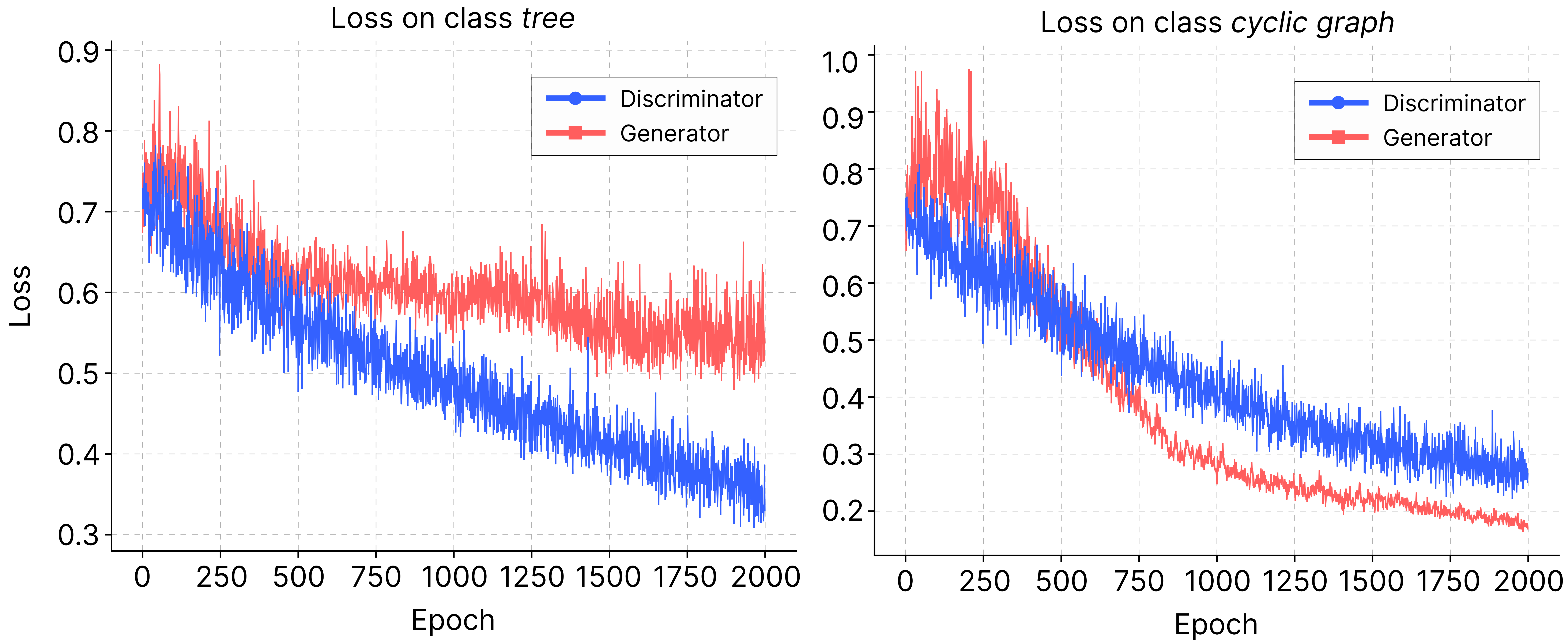
Algorithm 2: Example of `partial_order`

Require: $A \in \mathbb{R}^{n \times n}$

$E \leftarrow \texttt{positive\_edges}(A)$ ▷ *Get the set of edges from the adjacency matrix A*
$\neg E \leftarrow \texttt{negative\_edges}(A)$ ▷ *Get the set of non-existing edges from the adjacency matrix A*
$\mathcal{P} \leftarrow \{(\mathcal{E}{=}E, o{=}0), (\mathcal{E}{=}\neg E, o{=}1)\}$ ▷ *Build the partial order of the existing and non-existing edges with group tuples consisting of edge set $\mathcal{E}$, and oracle verification guard o.*
return $\mathcal{P}$

Pretty good actually when you have dual classes.

Lessons learned through RSGG-CE

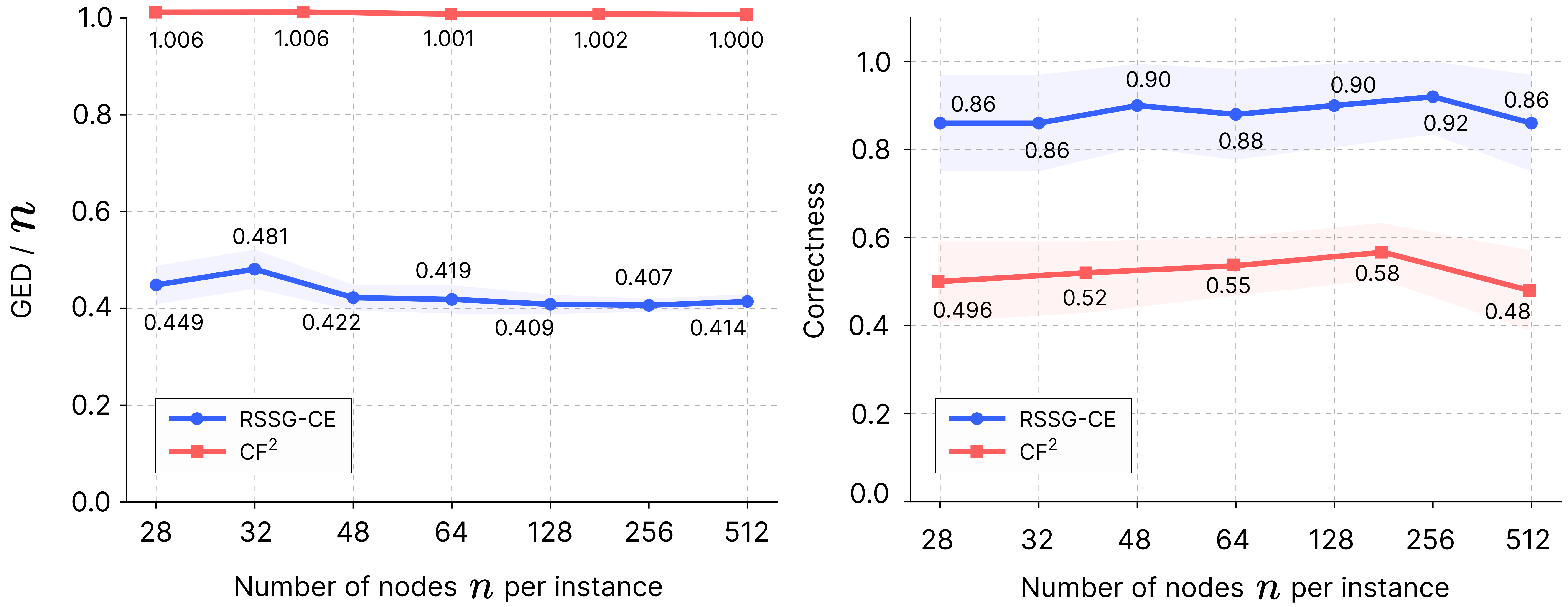


Loss on class *tree*
Loss
Epoch
Discriminator
Generator
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
250
500
750
1000
1250
1500
1750
2000
Loss on class *cyclic graph*
Epoch
Discriminator
Generator
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
250
500
750
1000
1250
1500
1750
2000

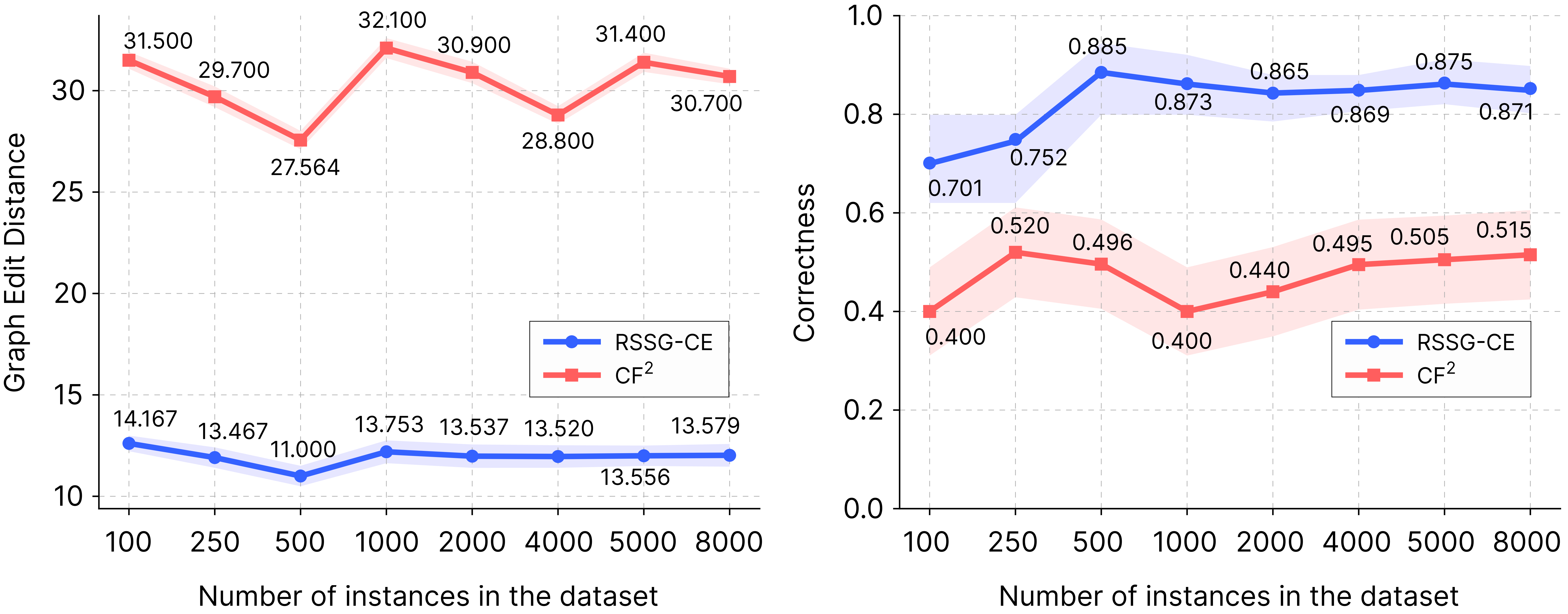
RSGG-CE has a gain of 66.98% and 19.65% in correctness.

		Methods				
		MEG †	CF^2 †	CLEAR ‡	G-CounteRGAN ‡	**RSGG-CE ‡**
TC	Runtime (s) ↓	272.110	4.811	25.151	632.542	**0.083**
	GED ↓	159.700	27.564	61.686	182.414	**11.000**
	Oracle Calls ↓	**0.000**	**0.000**	4341.600	1321.000	121.660
	Correctness ↑	0.530	0.496	0.504	0.504	**0.885**
	Sparsity ↓	2.510	0.496	1.110	3.283	**0.199**
	Fidelity ↑	0.530	0.496	0.504	0.504	**0.885**
	Oracle Acc. ↑	1.000	1.000	1.000	1.000	1.000
ASD	Runtime (s) ↓	×	**15.313**	275.884	969.255	80.000
	GED ↓	×	655.661	1479.114	3183.729	**234.853**
	Oracle Calls ↓	×	**0.000**	5339.455	1182.818	794.805
	Correctness ↑	×	0.463	0.554	0.529	**0.603**
	Sparsity ↓	×	0.850	1.917	4.125	**0.304**
	Fidelity ↑	×	**0.287**	0.319	0.265	**0.287**
	Oracle Acc. ↑	×	0.773	0.773	0.773	0.773

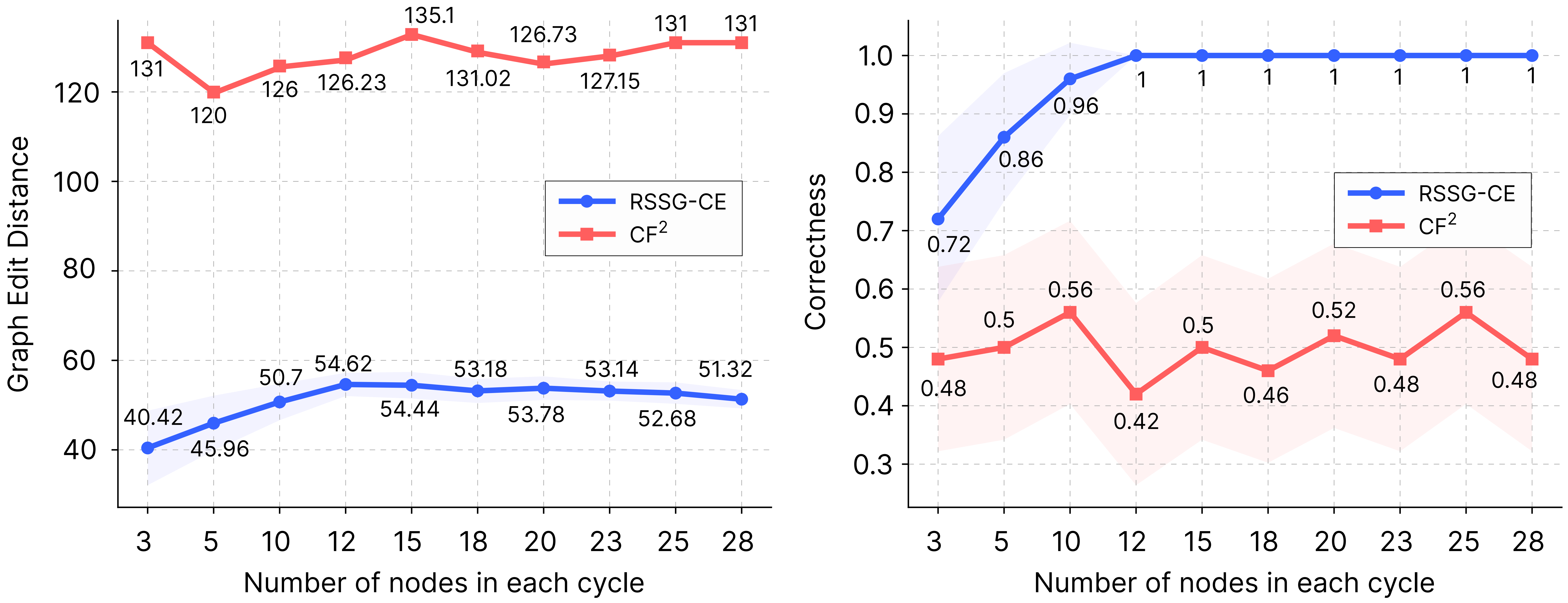
We don't care about larger graphs. Results depend only on dataset complexity



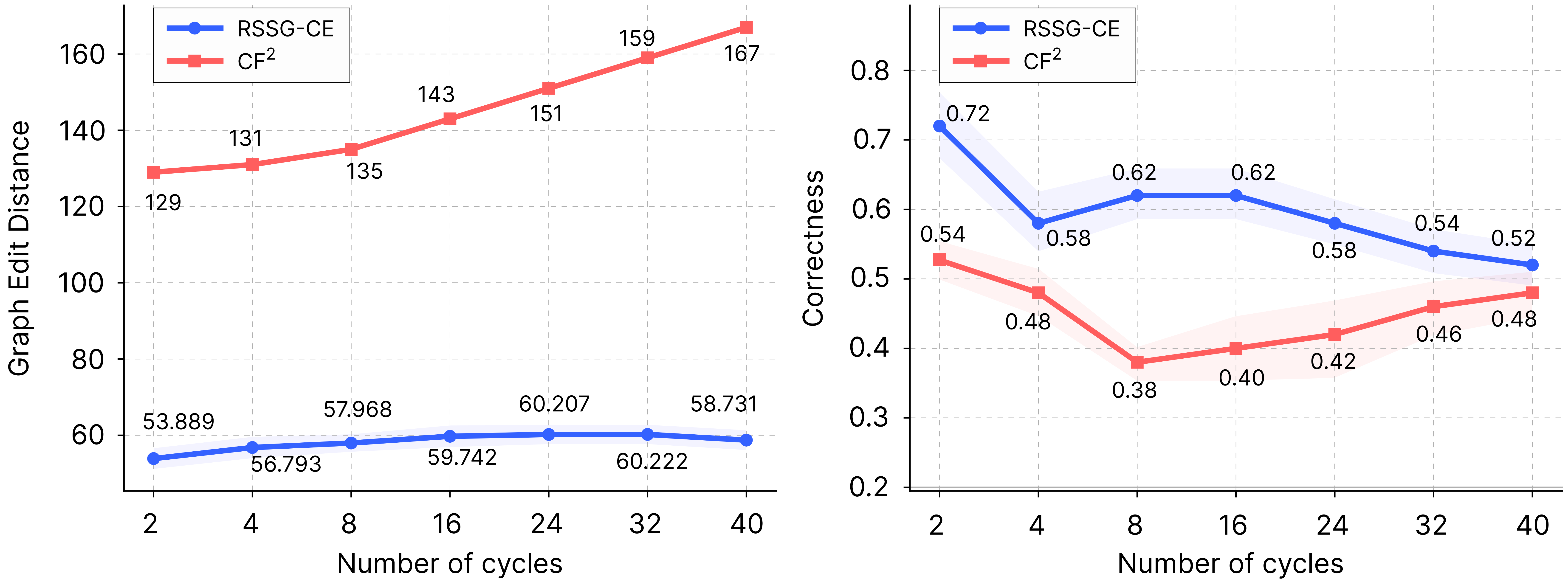
Performance stabilizes when the number of instances is greater than 500.



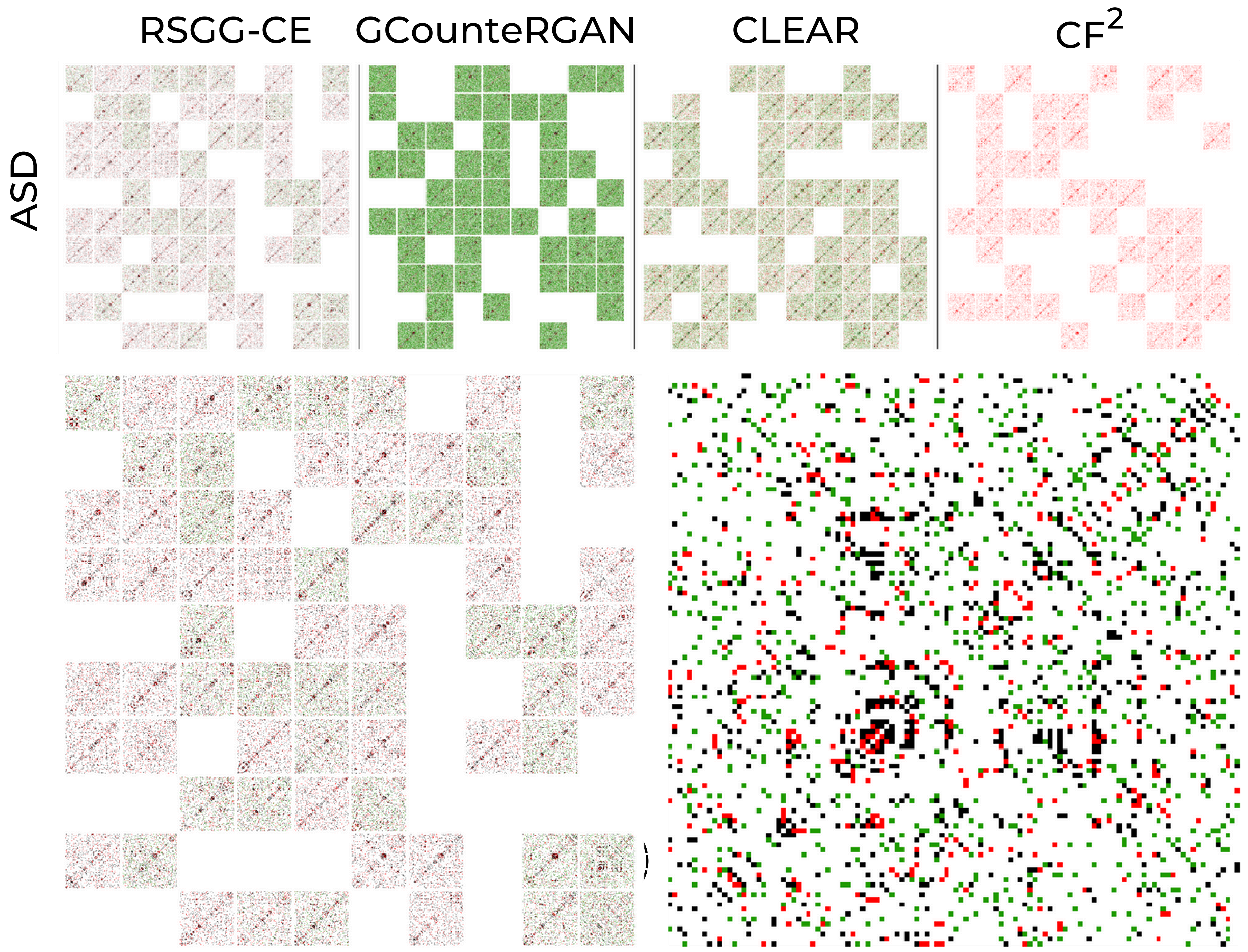
We scale perfectly when the number of nodes in a cycle increases (GED plateaus, and correctness is 1).



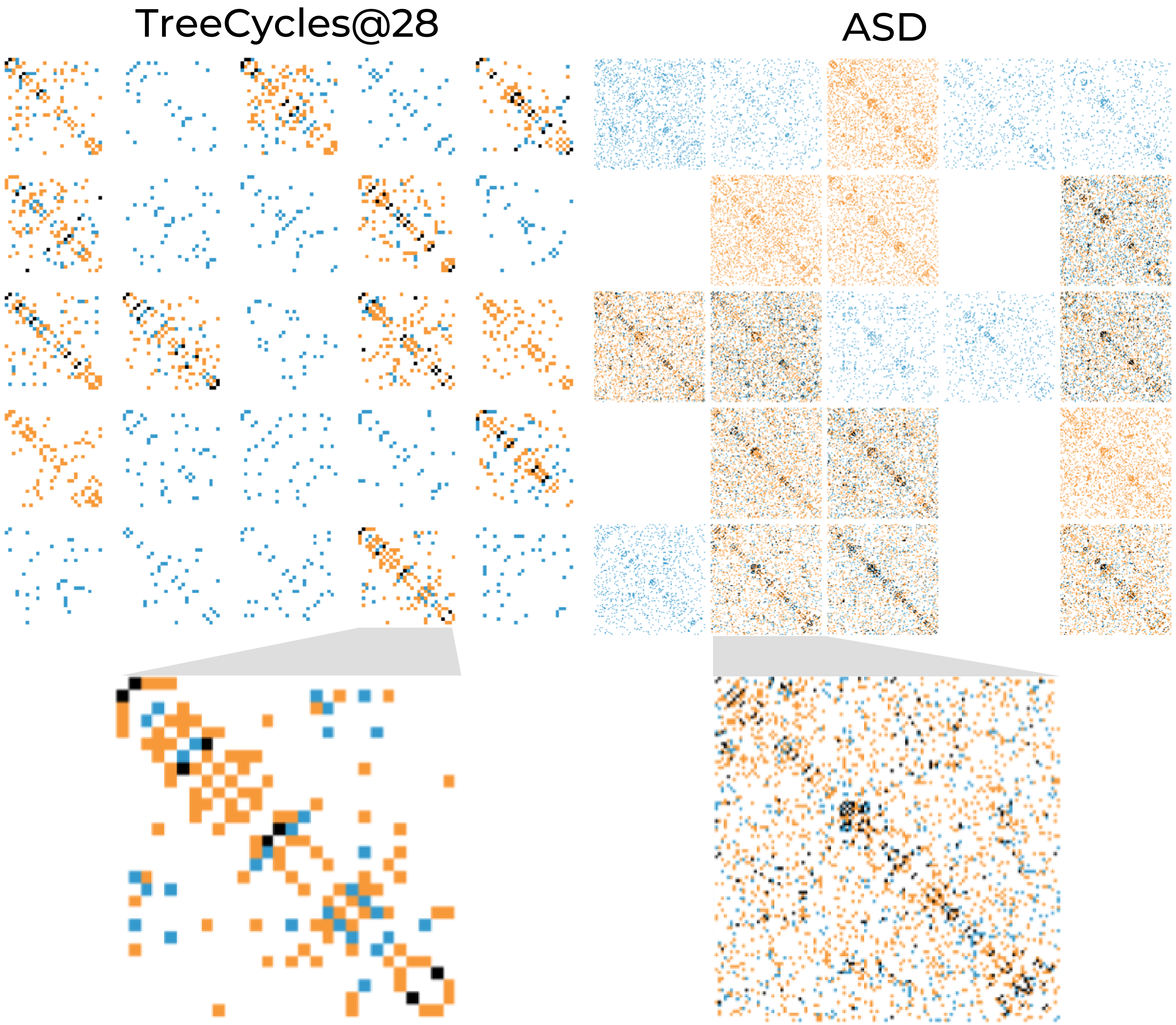
Even when the number of cycles increases, we don't need as many edge-cutting operations.



We can do **both** edge additions and removals



We perform a lot less perturbation on the graphs vs CLEAR



References

1. Abrate, C.; and Bonchi, F. 2021. *Counterfactual graphs for explainable classification of brain networks*. In **KDD'21**

2. Liu, Y.; Chen, C.; Liu, Y.; Zhang, X.; and Xie, S. 2021. *Multi-objective Explanations of GNN Predictions*. In **ICDM'21**

3. Nguyen, T. M.; Quinn, T. P.; Nguyen, T.; and Tran, T. 2022. *Explaining Black Box Drug Target Prediction through Model Agnostic Counterfactual Samples*. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**

4. Numeroso, D.; and Bacciu, D. 2021. *Meg: Generating molecular counterfactual explanations for deep graph networks*. In **IJCNN'21**

5. Ma, J.; Guo, R.; Mishra, S.; Zhang, A.; and Li, J. 2022. *CLEAR: Generative Counterfactual Explanations on Graphs*. In **NeurIPS'22**

6. Nemirovsky, D.; Thiebaut, N.; Xu, Y.; and Gupta, A. 2022. *CounteRGAN: Generating counterfactuals for real-time recourse and interpretability using residual GANs*. In **UAI'22**

7. Prado-Romero, M. A.; Prenkaj, B.; and Stilo, G. 2023. *Revisiting CounteRGAN for Counterfactual Explainability of Graphs*. In **ICLR'23 @ Tiny Paper Track**

QUICK DEMO

Food for Thought

Finding counterfactuals is mathematically equivalent to adversarially attacking a predictor, but they have different social connotations

???