Appendix

1 Analysis of Time and Memory Complexity

Table 1: Generation Time and Memory Complexity Comparison.

Generator	Time Complexity	Memory Complexity
HyperCL	$O(log_2\lvert\mathcal{V}\rvert \cdot \sum_{e\in\mathcal{E}} \lvert e\rvert)$	$O(\lvert\mathcal{V}\rvert + \sum_{e\in\mathcal{E}} \lvert e\rvert)$
HyperFF	$O(\lvert\mathcal{V}\rvert \cdot \sum_{e\in\mathcal{E}} \lvert e\rvert)$	$O(\lvert\mathcal{V}\rvert + \sum_{e\in\mathcal{E}} \lvert e\rvert)$
HyperPA	$O(\sum_{e\in\mathcal{E}} log_2\binom{\lvert\mathcal{V}\rvert}{\lvert e\rvert})$	$O(\sum_{e\in\mathcal{E}} 2^{\lvert e\rvert})$
HyperLAP	$O(log_2\lvert\mathcal{V}\rvert \cdot \sum_{e\in\mathcal{E}} \lvert e\rvert)$	$O(\lvert\mathcal{V}\rvert + \sum_{e\in\mathcal{E}} \lvert e\rvert)$
THera	$O(log_2\lvert\mathcal{V}\rvert \cdot \sum_{e\in\mathcal{E}} \lvert e\rvert)$	$O(\lvert\mathcal{V}\rvert + \sum_{e\in\mathcal{E}} \lvert e\rvert)$
HyRec	$O(L(\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)^{\frac{1}{L}} + \sum_{e\in\mathcal{E}} \lvert e\rvert)$	$O((\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)^{\frac{1}{L}} + \sum_{e\in\mathcal{E}} \lvert e\rvert)$

Table 1 [1] presents the time and memory complexity for all models, including HyRec. The complexities of the competitors are detailed in [1], while the time and memory complexity of HyRec are formally proven in Theorem 1 and Theorem 2, respectively. For the theorems, we assume that the total Kronecker power K can be evenly divided into L units.

Theorem 1 (Time Complexity of Hypergraph Generation in HyRec). *Given L unit number, time complexity of hypergraph generation in HyRec is* $O\left(L(|\mathcal{V}||\mathcal{E}|)^{\frac{1}{L}} + \sum_{e\in\mathcal{E}}|e|\right)$.

Proof. Let the initiator matrix $\Theta \in \mathbb{R}^{N_1\times M_1}$, where $N_1 = log_K|\mathcal{V}|$ and $M_1 = log_K|\mathcal{E}|$, represent the initiator matrix for generating a K-order hypergraph to reproduce $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The time complexity of computing the k-th power of the Kronecker product of Θ is $O(N_1^k M_1^k)$, as each element of A is iterated over all elements of B when computing $A \otimes B$. In addition, the time complexity of independently sampling each element is proportional to the size of the Kronecker probabilistic matrix. Under the assumption that the total Kronecker power K can be evenly divided into L units, the time complexity for generating the hypergraph, after dividing it into L units, is $O\left(L\cdot(N_1^{\frac{K}{L}} M_1^{\frac{K}{L}})\right) = O\left(L\cdot(|\mathcal{V}|^{\frac{1}{L}}|\mathcal{E}|^{\frac{1}{L}})\right)$, reducing the computation significantly by working with smaller Kronecker powers.

Then, let $\hat{\Theta}^i$ represent the sampled Kronecker graph for i-th unit, and let $f_{NZ}(\cdot)$ represent the number of non-zero entries in the matrix. We assume that $E[f_{NZ}(\hat{\Theta}^i)] = (\sum_{e\in\mathcal{E}}|e|)^{\frac{1}{L}}$. The time complexity for computing $\bigotimes_{i=1}^{L}\hat{\Theta}^i$ depends on the number of non-zero elements in each unit $\hat{\Theta}^i$. Thus, the time complexity is $O(E[\prod_{i=1}^{L} f_{NZ}(\hat{\Theta}^i)])$. Since the sampling at each unit is independent, this can be simplified to $O(\prod_{i=1}^{L} E[f_{NZ}(\hat{\Theta}^i)]) = O(\sum_{e\in\mathcal{E}}|e|)$. Therefore, the total time complexity is $O\left(L(|\mathcal{V}||\mathcal{E}|)^{\frac{1}{L}} + \sum_{e\in\mathcal{E}}|e|\right)$. ∎

Theorem 2 (Memory Complexity of Hypergraph Generation in HyRec). *Given L unit number, memory complexity of hypergraph generation in HyRec is* $O((|\mathcal{V}||\mathcal{E}|)^{\frac{1}{L}} + \sum_{e\in\mathcal{E}}|e|)$.

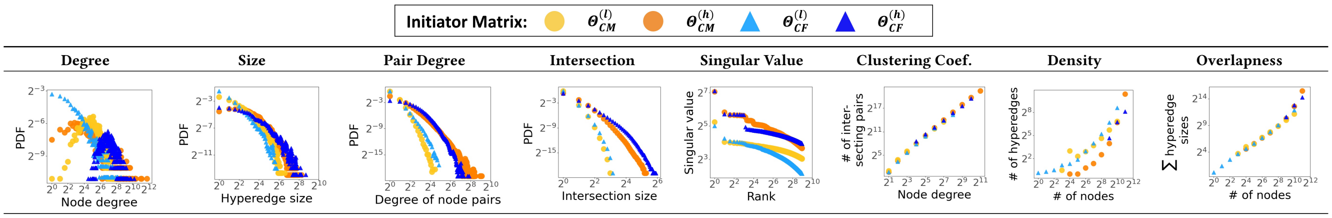
Proof. At each unit, memory is required to compute the Kronecker power of Θ, with the memory usage proportional to the size of the resulting Kronecker power. Similar to the time complexity analysis, this results in a memory complexity of $O((|\mathcal{V}||\mathcal{E}|)^{\frac{1}{L}})$. Additionally, when computing the Kronecker product of the sampled matrices from each unit, only the non-zero elements of the Kronecker graphs need to be stored, leading to a memory complexity of $O(\sum_{e\in\mathcal{E}}|e|)$. Therefore, the total memory complexity is $O((|\mathcal{V}||\mathcal{E}|)^{\frac{1}{L}} + \sum_{e\in\mathcal{E}}|e|)$. ∎

Table 2: Time Complexity of SingFit (Algorithm 1).

SingFit (Algorithm 1)	**Time Complexity**
Generation (Line 7)	$O(L(\lvert\mathcal{V}\rvert\lvert\mathcal{E}\rvert)^{\frac{1}{L}})$
Computing Singular Values (Line 8)	$O(L \cdot min(\lvert\mathcal{E}\rvert^2\lvert\mathcal{V}\rvert, \lvert\mathcal{E}\rvert\lvert\mathcal{V}\rvert^2)^{\frac{1}{L}})$
Computing Degree Distributions (Line 9)	$O(L \cdot (\sum_{e\in\mathcal{E}} \lvert e\rvert)^{\frac{1}{L}} + \lvert\mathcal{V}\rvert)$
Computing Size Distributions (Line 10)	$O(L \cdot (\sum_{e\in\mathcal{E}} \lvert e\rvert)^{\frac{1}{L}} + \lvert\mathcal{E}\rvert)$

[1] We would like to note that in all experiments, we use at least two units (i.e., $L \geq 2$).

Table 3: Properties of Hypergraphs Derived from Four Different Cases of Initiator Matrices.



SingFit is described in Algorithm 1 and implemented using PyTorch on an NVIDIA GeForce RTX 2080 Ti GPU. We analyze the time complexity line by line (Table 2):

- Line 7: The time complexity for computing the Kronecker power of Θ and sampling the binary incidence matrix is $O(L(|\mathcal{V}||\mathcal{E}|)^{\frac{1}{L}})$, as derived in the proof of Theorem 1.
- Line 8: Singular value computation is performed using the PyTorch library, which relies on the Jacobi eigenvalue solver in cuSolver. For an arbitrary matrix of size $m \times n$, where $m \geq n$, the complexity is $O(mn^2)$. Assuming the full Kronecker power is divided evenly into L units, each of size $|\mathcal{V}|^{\frac{1}{L}} \times |\mathcal{E}|^{\frac{1}{L}}$, the complexity of computing singular values across all units is $O(L \cdot \min(|\mathcal{E}|^2|\mathcal{V}|, |\mathcal{E}||\mathcal{V}|^2)^{\frac{1}{L}})$. To obtain the singular values for the full Kronecker power, the algorithm computes the Kronecker product of singular values from all units, which is proportional to the number of singular values of the full Kronecker power, i.e., $O(min(|\mathcal{V}|, |\mathcal{E}|))$. Therefore, the total complexity is $O(L \cdot \min(|\mathcal{E}|^2|\mathcal{V}|, |\mathcal{E}||\mathcal{V}|^2)^{\frac{1}{L}} + min(|\mathcal{V}|, |\mathcal{E}|))$.
- Line 9: To compute the degree distribution, the algorithm first calculates the node degree vector (where the i-th element represents the degree of the i-th node) by iterating over the non-zero elements in the sampled unit matrices. Assuming $E[f_{NZ}(\hat{\Theta}^{S_i})] = (\sum_{e \in \mathcal{E}} |e|)^{\frac{1}{L}}$, where $f_{NZ}(\cdot)$ represents the number of non-zero elements in the matrix, this step has a time complexity of $O(L \cdot (\sum_{e \in \mathcal{E}} |e|)^{\frac{1}{L}}$. Next, the Kronecker product of the L node degree vectors is computed, which takes $O(|\mathcal{V}|)$. Therefore, the total time complexity for computing degree distribution is $O(L \cdot (\sum_{e \in \mathcal{E}} |e|)^{\frac{1}{L}} + |\mathcal{V}|)$.
- Line 10: Computing the size distribution follows the same process as for the degree distribution. The Kronecker product of the size vectors corresponds to the number of edges, with a complexity of $O(|\mathcal{E}|)$. Thus, the total complexity for computing the size distribution is $O(L \cdot (\sum_{e \in \mathcal{E}} |e|)^{\frac{1}{L}} + |\mathcal{E}|)$.

2 Simple Examples of Initiator Matrix Analysis

We explore how different settings of the initiator matrix influence the properties of its Kronecker hypergraph through four cases. Specifically, we compare two structures: a community structure (CM), where nodes mainly interact within the same community, and a core-fringe structure (CF), where core nodes exist and fringe nodes primarily interact with core nodes. In both structures, we explore two cases where the probabilities in the initiator matrix differ, with one being slightly lower than the other. Thus, we analyze the following four initiator matrices: $\Theta^{(l)}_{CM}$, $\Theta^{(h)}_{CM}$, $\Theta^{(l)}_{CF}$ and $\Theta^{(h)}_{CF}$:

$$\mathbf{\Theta}^{(\mathbf{l})}_{\mathbf{CM}} = \begin{bmatrix} 0.7 & 0.7 & 0.0001 & 0.0001 \\ 0.7 & 0.7 & 0.0001 & 0.0001 \\ 0.0001 & 0.0001 & 0.7 & 0.7 \end{bmatrix}.$$

$$\mathbf{\Theta}^{(\mathbf{l})}_{\mathbf{CF}} = \begin{bmatrix} 0.7 & 0.7 & 0.7 & 0.7 \\ 0.7 & 0.5 & 0.7 & 0.0001 \\ 0.5 & 0.5 & 0.0001 & 0.0001 \end{bmatrix}.$$

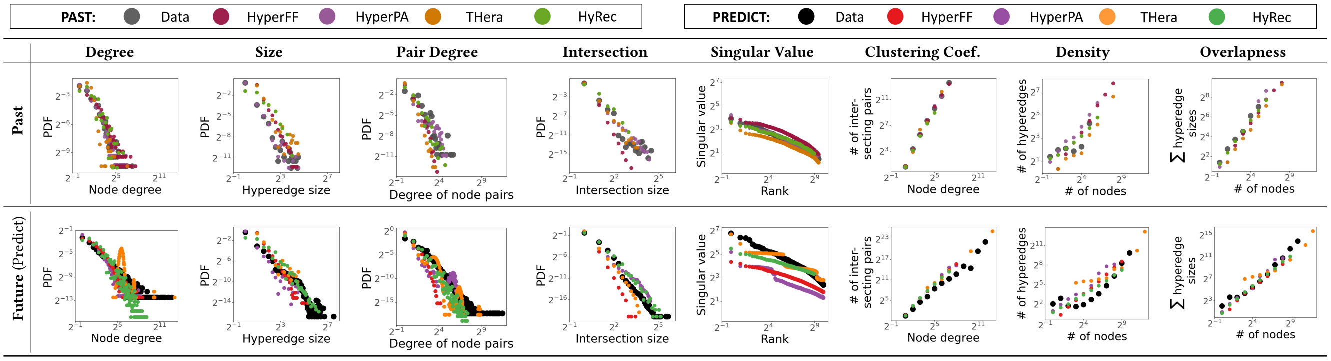
$$\mathbf{\Theta}^{(\mathbf{h})}_{\mathbf{CM}} = \begin{bmatrix} 0.9 & 0.9 & 0.0001 & 0.0001 \\ 0.9 & 0.9 & 0.0001 & 0.0001 \\ 0.0001 & 0.0001 & 0.9 & 0.9 \end{bmatrix}.$$

$$\mathbf{\Theta}^{(\mathbf{h})}_{\mathbf{CF}} = \begin{bmatrix} 0.9 & 0.9 & 0.9 & 0.9 \\ 0.9 & 0.7 & 0.9 & 0.0001 \\ 0.7 & 0.7 & 0.0001 & 0.0001 \end{bmatrix}.$$

Table 3 compares properties derived from four different initiator matrices. There are clear differences between $\Theta^{(l)}_{CM}$ ($\Theta^{(l)}_{CF}$) and $\Theta^{(h)}_{CM}$ ($\Theta^{(h)}_{CF}$), particularly as the probability increases. Higher probabilities lead to higher node degrees, larger hyperedge sizes, greater node pair degrees, and larger hyperedge intersections, but lower egonet density. When comparing the community and core-fringe structures (i.e., $\Theta^{(h)}_{CM}$ vs. $\Theta^{(h)}_{CF}$), notable differences are evident in the distributions of node degrees, density, and overlap. In the core-fringe structure $\Theta^{(h)}_{CF}$, most nodes with high degrees and the distribution centers around a peak with a rapid decline on both sides. By contrast, the community structure shows a more varied distribution of node degrees, with most nodes having relatively lower degrees. Regarding density and overlap, $\Theta^{(h)}_{CF}$ generates larger egonets, as indicated by a few data points on the right side of the distribution. This suggests that most nodes are highly

Table 4: HyRec Effectively Extrapolates Real-world Hypergraphs, Even for Extended Growth Periods.

(a) **HyRec accurately predicts fourfold hypergraph growth.** After fitting to a past snapshot of the NDC-substances dataset with 1/4 of the total nodes, HyRec (green) accurately predicts the properties of the future snapshot (the full dataset).



(b) **HyRec excels in extrapolation with minimal input parameters, even for long-term predictions.** HyRec performs the **best overall** in extrapolation as the node count quadruples, requiring far fewer parameters than the second-best model (THera). The best, second-best, and third-best performance are highlighted in blue, green, and yellow, respectively.

	# Input Parameters		Average Ranking (Across 11 Hypergraph Datasets)									
	Min	**Max**	**Degree**	**Size**	**Pair Deg.**	**Intersect.**	**Singular Value**	**Clustering Coef.**	**Density**	**Overlapness**	**Effective Diam.**	**Average**
HyperFF	**2**	**2**	**1.909**	*2.273*	3.182	2.727	*2.545*	*2.091*	*2.364*	2.636	2.818	2.505
HyperPA	143	515,085	3.182	2.636	*2.545*	2.909	4.000	3.273	3.273	3.364	2.636	3.091
THera	86	199,803	2.545	**2.000**	2.727	**2.000**	**1.727**	3.182	**2.182**	*2.455*	**2.000**	*2.313*
HyRec	*12*	*396*	*2.364*	3.091	**1.545**	*2.364*	**1.727**	**1.455**	**2.182**	**1.545**	*2.545*	**2.091**

interconnected, with core nodes acting as bridges that connect a significant portion of the hypergraph. However, when the probability decreases in the core-fringe structure (from $\Theta_{CF}^{(h)}$ to $\Theta_{CF}^{(l)}$), node degrees become more varied, and egonet sizes also vary, with most nodes having lower degrees and more diverse egonets.

3 Longer-term Extrapolation

In this section, we extend the extrapolation task described in Section 7.3 to account for longer growth periods. The task involves forecasting hypergraph growth by fitting HyRec to an initial snapshot and predicting its future state as it expands. For this experiment, we fit HyRec to a snapshot containing hyperedges when only 1/4 of the total nodes have appeared. We then predict the properties of the full hypergraph, where the node count has quadrupled. This differs from the previous setup in Section 7.3, where HyRec was fit to 50% of the nodes. We apply the same evaluation protocol to HyperPA, HyperFF, and THera, but exclude HyperLAP and HyperCL, as they are not suitable for extrapolation beyond the input data. The results, shown in Table 4, demonstrate HyRec 's strength in predicting hypergraph evolution, especially over long-term growth. In Table 4 (a), a visual comparison in the NDC-substances dataset confirms that HyRec accurately captures both the past hypergraph and its future properties. Across 11 datasets, Table 4 (b) shows that HyRec consistently performs the **best overall** in predicting hypergraph growth, accurately modeling key properties as the node count increases fourfold. Remarkably, HyRec accomplishes this while requiring two orders of magnitude fewer input parameters (in terms of scalars) than the second-best model, THera.

4 Applicability to Non-Power-Law Hypergraphs

Although HyRec is inspired by recursive structures (a.k.a. fractals) that naturally reproduce power-law distributions, it demonstrates both theoretical flexibility and decent empirical performance for non-power-law hypergraphs. *Theoretically*, HyRec is not restricted to power-law distributions. By generating multinomial distributions, it can approximate a wide range of patterns, including Gaussian degree distributions when all initiator matrix entries are set to the same value. *Empirically*, we evaluated HyRec on four datasets: modelnet_40 (non-power-law hyperedge sizes), imdb (non-power-law node degrees), and two random hypergraphs with normal degree distributions and uniform hyperedge size distributions. **Fitting Performance:** As shown in Table5, HyRec ranked third on average, outperforming Thera, HyperFF, and HyperPA. While HyperLAP and HyperCL surpass HyRec, it is worth noting that both require significantly more input parameters. Additionally, HyperCL performs better on random-like datasets due to its reliance on ground-truth input distributions. **Extrapolation Performance:** As shown in Table6, HyRec ranks second on average, surpassing most baselines except Thera. It effectively captured dynamic size distributions, such as the emergence of larger hyperedges in future snapshots, whereas all other models—except for HyperFF—generated hyperedge sizes solely based on past snapshot distributions. Thera, however, outperformed HyRec on static size distributions due to its fixed assumptions. These results highlight HyRec 's adaptability and effectiveness in handling non-power-law hypergraphs while maintaining its scalability and requiring minimal input parameters.

Table 5: Comparison of fitting performance on non-power-law hypergraphs. HyRec ranks third on average, outperforming Thera, HyperFF, and HyperPA, while requiring fewer input parameters than HyperLAP and HyperCL. The best, second-best, and third-best performance are highlighted in blue, green, and yellow, respectively.

	# Input Parameters		Average Ranking (Across 11 Hypergraph Datasets)									
	Min	Max	Degree	Size	Pair Deg.	Intersect.	Singular Value	Clustering Coef.	Density	Overlapness	Effective Diam.	Average
HyperCL	5,954	236,268	1.750	1.000	2.500	3.000	2.000	2.000	2.500	2.250	2.750	2.194
HyperFF	2	2	4.500	5.750	4.750	3.500	5.250	5.000	5.000	4.500	3.750	4.667
HyperPA	5,954	236,268	6.000	4.750	4.000	5.000	5.750	5.750	5.750	6.000	5.250	5.361
HyperLAP	5,954	236,268	1.750	1.000	2.500	1.500	1.500	1.500	1.250	1.500	3.250	1.750
THera	2,019	235,267	4.000	2.500	4.000	4.250	3.750	3.250	3.500	3.750	3.000	3.556
HyRec	15	48	3.000	4.500	3.000	3.750	2.750	3.500	3.000	3.000	2.750	3.250

Table 6: Comparison of extrapolation performance on non-power-law hypergraphs. HyRec ranks second on average, effectively capturing dynamic size distributions, while Thera outperforms it on static distributions due to its fixed assumptions. The best, second-best, and third-best performance are highlighted in blue, green, and yellow, respectively.

	# Input Parameters		Average Ranking (Across 11 Hypergraph Datasets)									
	Min	Max	Degree	Size	Pair Deg.	Intersect.	Singular Value	Clustering Coef.	Density	Overlapness	Effective Diam.	Average
HyperFF	2	2	2.500	3.000	2.750	2.500	3.250	2.000	3.000	2.500	2.250	2.639
HyperPA	2,553	11,397	3.750	3.250	3.000	3.250	3.750	3.500	3.500	4.000	4.000	3.556
THera	624	6,086	1.000	1.750	2.750	1.750	1.250	2.000	1.250	1.750	2.000	1.722
HyRec	35	81	2.750	2.000	1.500	2.500	1.750	2.500	2.250	1.750	1.750	2.083

Table 7: Correlation (0.95) between hypergraph evolutionary differences—quantified as the sum of D-statistics across five key properties—and HyRec 's extrapolation performance (where lower is better), demonstrating the increasing challenge of extrapolation as temporal changes intensify.

Dataset	Hypergraph Evolutionary Difference (Sum of D-stat.)	Extrapolation Performance (Diff.)
threads-ubuntu	0.0678	1.2588
threads-math	0.0909	1.2709
coauth-geology	0.1426	0.5263
NDC-substances	0.5751	0.5176
tags-ubuntu	0.6464	0.5889
tags-math	0.8438	0.5774
NDC-classes	1.1385	0.8571
email-Enron	1.3418	0.9139
email-Eu	1.3658	0.7888
contact-high	1.7674	1.1302
contact-primary	2.1324	1.2627

5 Extrapolation Performance Under Temporal Evolution and Structural Changes

To gain deeper insights into the experimental results presented in Section 7.3, we investigate how HyRec 's extrapolation performance is affected by datasets exhibiting significant temporal evolution or abrupt structural changes. Our evaluation across eleven real-world datasets reveals that greater evolutionary changes in hypergraphs lead to increased extrapolation difficulty. To quantify these changes, we compute the difference between the initial half of each hypergraph (used for fitting) and the target hypergraph (used for extrapolation) using the Kolmogorov-Smirnov (D-statistic) across five key distributions: degree, hyperedge size, pair degree, intersection size, and singular values. We then assess our model's extrapolation performance by averaging differences over nine structural properties before ranking (refer to Section 7.1). As shown in Table 7, we observed a strong correlation of 0.95 between hypergraph evolutionary differences and extrapolation performance (where lower is better), indicating that datasets with greater temporal changes pose a more significant challenge. The only exceptions were the two threads datasets, which exhibited unique structural dynamics that deviated from the trend. These findings highlight that while HyRec effectively handles dynamic hypergraphs, its extrapolation performance can vary based on the degree of temporal evolution and abrupt structural changes in the dataset.

6 Discussion

A limitation of HyRec is its difficulty in reproducing size distributions when the maximum hyperedge size is constrained and deviates from a heavy-tailed distribution. For instance, in the tags dataset, the number of tags assigned to a post (i.e., the size of the hyperedge) is often limited to four. Other methods (except hyperFF) can easily replicate any size distribution by generating hypergraphs directly from the input ground-truth size distribution. In such cases, HyRec may struggle to match the size distribution as precisely as these methods. However, a key strength of HyRec lies in its flexibility. Unlike other methods (again, except hyperFF) that require the hyperedge size distribution as input for each generation, HyRec operates independently of such input, allowing it to generate dynamic, evolving size distributions over time. In contrast, methods relying on static input distributions lack this adaptability, making them less effective when extrapolating to changing distributions. Although HyRec may face limitations when the distribution deviates from a heavy-tailed structure, it is important to note that many real-world hypergraphs exhibit heavy-tailed distributions, as documented in numerous studies (see Section 2.2 and our findings in Section 4). Additionally, HyRec outperforms nearly all other methods in fitting 11 real-world hypergraphs and surpasses all methods in extrapolating these hypergraphs. In contrast, HyperFF, which also does not require input size distributions, performs poorly.

For future work, we can explore extending HyRec to handle weighted, attributed, or labeled hypergraphs, which would enable it to model more complex and nuanced real-world structures. Additionally, investigating the effectiveness of the initiator matrices in capturing key features across different hypergraphs could be valuable. This could involve tasks such as graph classification, where the initiator matrices are expected to provide meaningful insights into the distinct structural properties across various domains.

References

[1] Sunwoo Kim, Fanchen Bu, Minyoung Choe, Jaemin Yoo, and Kijung Shin. 2023. How Transitive Are Real-World Group Interactions?–Measurement and Reproduction. In *KDD*.