D OTHER SUPPLEMENTARY MATERIALS

D.1 Task Extension: Using Edge Multiplicities

Throughout this work we do not assume that the projected graph has edge multiplicities. Relying on edge multiplicities addresses a simpler version of the problem which might limit its applicability. That said, some applications may come with edge multiplicity information, and it is important to understand what is possible in this more tractable case. Here we provide an effective unsupervised method as a foundation for further work.

From Sec. 2, the multiplicity of an edge (u, v) is the number of hyperedges containing both u and v. It is not hard to show that knowledge of the edge multiplicities does not suffice to allow perfect reconstruction, and so we still must choose from among a set of available cliques to form hyperedges. In doing this with multiplicity information, we need to ensure that the cliques we select add up to the given edges multiplicities. We do this by repeatedly finding maximal cliques, removing them, and reducing the multiplicities of their edges by 1. We find that an effective heuristic is to select maximal cliques that have large size and small average edge multiplicities (combining these for example using a weighted sum).

Table 7 gives the performance on the datasets we study. We can see that with edge multiplicities our unsupervised baseline outperforms all the methods not using edge multiplicities on most datasets, showing the power of this additional information. The performance, however, is still far from perfect, and we leave the study of this interesting extension to future work.

DBLP	Enron	P.School	H.School
82.75 (+1.56)	19.79 (+3.77)	10.46 (-32.60)	19.30 (-35.56)
Foursquare	**Hosts-Virus**	**Directors**	**Crimes**
83.91 (+10.35)	67.86 (+19.01)	100.0 (+0.00)	80.47 (+1.20)

Table 7: Performance of the proposed baseline using available edge multiplicities. In parenthesis reported the increment against the best-performed method not using edge multiplicities (cr. Table 3).

D.2 Storage Comparison

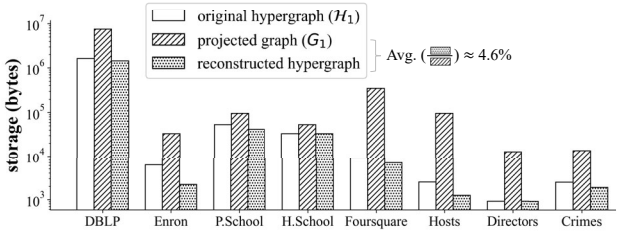

Figure 14: Comparing the storage of original hypergraphs, projected graphs and reconstructed hypergraphs. Each hypergraph/graph is stored as a nested array with `int64` node indexes. Over the 8 datasets on average, a reconstructed hypergraph takes only 4.6% the storage of a projected graph.

As mentioned in Introduction, a side bonus of having a reconstructed hypergraph versus a projected graph is that the former typically requires much less storage space. As a mini-study, we compare the storage of each hypergraph, its projected graph, and its reconstruction generated by SHyRe-count. We use the unified data structure of a nested array to represent the list of edges/hyperedges. Each node is indexed by an int64. Fig.14 visualizes the results. We see that the reconstructions take 1 to 2 orders of magnitude less storage space than the projected graphs and are closest to the originals.

D.3 Visualizing the Reconstruction

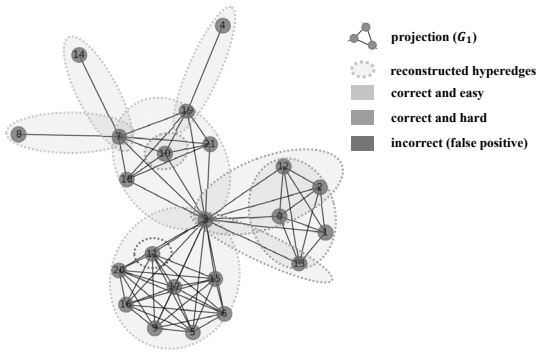

Figure 15: A small part of the hypergraph reconstructed by SHyRe-motif on DBLP dataset. Projection G_1 is drawn in black. The shaded ellipsis are the reconstructed hyperedges. Those with green dashed edges are difficult to reconstruct in the absence of the training data. Notice that $\{3, 12, 2, 0, 13, 1\}$ is a maximal clique (in this local graph).

Fig.15 visualizes a portion of the actual reconstructed hypergraph by SHyRe-count on DBLP dataset. We include more explanation and analysis in the caption.