Supplementary Material of GraphRPM

Injective analysis

In this section, we will briefly analyze why we can get different representations from different subgraph structures. According to GIN [1], GNN currently used for graph-level tasks can be divided into two steps as shown below.

$$h_v^{(k)} = AGGREGATE^{(k)}(h_v^{(k-1)}, COMBINE^{(k)}(N_v^{(k-1)})) \tag{1}$$

$$h_G = READOUT(\{h_v^k | v \in G\}) \tag{2}$$

where $N_v^{(k)}$ denotes the k-layer neighborhood information of node v. If the neighborhood combine, aggregation, and graph-level readout functions are injective, the resulting GNN is as powerful as Weisfeiler-Lehman (WL) isomorphism test [2] that works well in general but with a few exceptions such as regular graphs [3]. WL test is sufficient for our isomorphism discrimination.

For node v_j in a directed graph G, the feature combination of all its incoming edges e_{ij} and the starting nodes v_i of these incoming edges form a multiset $\{(h_i, f_{e_{ij}})\}$. Certain popular injective set functions such as mean aggregators are not injective functions over multisets, while sum aggregators can represent injective [4]. Thanks to the universal approximation theorem [5, 6], we can use MLPs over summation of representations to model sum aggregators as in GIN [1]. Therefore, recall our modified EGIN convolve and readout equations:

$$\begin{aligned} h_v^{(k)} = MLP_n^{(k)}((1+\epsilon^{(k)})h_v^{(k-1)} + \\ \sum\nolimits_{e \in IN(v)} MLP_0(h_{S_e}^{(k-1)} || f_e^{(k-1)})) \end{aligned} \tag{3}$$

$$f_e^{(k)} = MLP_e^{(k)}(f_e^{(k-1)} || (h_{S_e}^{(k-1)}) + h_{E_e}^{(k-1)}) \tag{4}$$

$$h_G = MLP^{(R)} \sum\nolimits_{v \in G} h_v^{(K)} \tag{5}$$

Equation (4) and the right part of equation (3) guarantee the injective property of the combine function in equation(1), while the left part of equation (4) and equation (5) guarantee the injective property of aggregate function and readout function respectively.

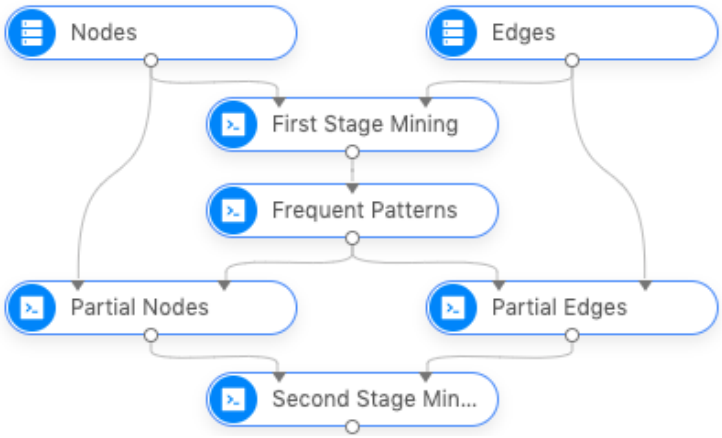

Figure 1: Two-staged mining task

Production deployment

We use a proprietary visual programming R&D platform that integrates deployment, release, and monitoring to run our two-staged mining task. As shown in Figure 1, we construct a data flow graph with nodes denoting computing components and edges representing data communication. The two input components are the node table and the edge table. We use a mining component to complete the first stage of subgraph mining, and then extract the new node table and edge table according to top-ranked patterns. Finally, we use another mining component to complete the second stage of subgraph mining.

References

[1] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[2] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9):12–16, 1968.

[3] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[4] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[5] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[6] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.