Supplementary Materials

Anonymous Author(s)
Affiliation
Address
email

A Proof of Theorem 2, Proposition 3 and Corollary 4

A.1 Proof of Theorem 2

We first restate Theorem 2 here.

Theorem 2. *Given two graphs $G = (\mathcal{V}_G, \mathcal{E}_G), H = (\mathcal{V}_H, \mathcal{E}_H)$, and let $\mathbf{u} = (u_1, ..., u_k) \in \mathcal{V}_G^k, \mathbf{v} = (v_1, ..., v_k) \in \mathcal{V}_H^k$ where k is a fixed value to be the target tuples we want to learn from G and H respectively. Then, the following statements are equivalent for any i, j satisfying $i \geq 0, j > 0, i + j = k$:*

- *A i-j-$\mathrm{NL_E}$ model gives $\mathbf{u}$ and $\mathbf{v}$ the same representation.*
- $(\mathbf{u}, G) \simeq (\mathbf{v}, H)$.

Proof. To prove the theorem, we need to first use the following fact about higher-order GNNs:

Lemma A.1. *A k-GNN with most expressive GNNs distinguishes any non-isomorphic k-tuples, and assigns isomorphic k-tuples with the same representation.*

With Lemma A.1 we are now able to prove Theorem 2.

$1 \to 2$: Suppose a i-j-$\mathrm{NL_E}$ gives $\mathbf{u}$ and $\mathbf{v}$ the same representation. From Lemma A.1 it is obvious that we have

$$\left(\mathbf{u}_{(i+1):}, G_{(\mathbf{u}_{:i})}\right) \simeq \left(\mathbf{v}_{(i+1):}, H_{(\mathbf{v}_{:j})}\right).$$

From the definition of node labeling, we know that there exists an isomorphism π from G to H satisfying

$$\pi(u_i) = v_i \text{ for all } i \in \{i+1, ..., k\}, L(w \mid \mathbf{u}_{:i}, G) = L(\pi(w) \mid \mathbf{v}_{:i}, H) \text{ for all } w \in \mathcal{V}_G.$$

Since $L(\pi(w) \mid \mathbf{v}_{:i}, H)$ for all $w \in \mathcal{V}_G \Rightarrow \pi(\mathbf{u}_{:i}) = \mathbf{v}_{:i}$, we have

$$\begin{aligned}
&\left(\mathbf{u}_{(i+1):}, G_{(\mathbf{u}_{:i})}\right) \simeq \left(\mathbf{v}_{(i+1):}, H_{(\mathbf{v}_{:j})}\right) \\
&\Rightarrow \exists \text{ an isomorphism } \pi : \mathcal{V}_G \to \mathcal{V}_H, \pi(\mathbf{u}_{:i}) = (\mathbf{v}_{:i}), \pi(\mathbf{u}_{(i+1):}) = (\mathbf{v}_{(i+1):}) \\
&\Rightarrow \exists \text{ an isomorphism } \pi : \mathcal{V}_G \to \mathcal{V}_H, \pi(\mathbf{u}) = (\mathbf{v}) \\
&\Rightarrow (\mathbf{u}, G) \simeq (\mathbf{v}, H).
\end{aligned}$$

$2 \to 1$: With Lemma A.1 we can deduce

$$\begin{aligned}
&(\mathbf{u}, G) \simeq (\mathbf{v}, H) \\
\implies &\exists \text{ an isomorphism } \pi : \mathcal{V}_G \to \mathcal{V}_H, \pi(\mathbf{u}_{:i}) = \mathbf{v}_{:i} \\
\implies &\forall w \in \mathcal{V}_G, L(w \mid \mathbf{u}_{:i}, G) = L(w \mid \mathbf{v}_{:i}, H) \\
\implies &\left(\mathbf{u}_{(i+1):}, G_{(\mathbf{u}_{:i})}\right) \simeq \left(\mathbf{v}_{(i+1):}, H_{(\mathbf{v}_{:j})}\right) \\
\implies &i\text{-}j\text{-}\mathrm{NL_E} \text{ gives } \mathbf{u} \text{ and } \mathbf{v} \text{ the same representation.}
\end{aligned}$$

Since the above equations hold for all $i \geq 0, j > 0, i + j = k$, Theorem 2 is proved. □

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

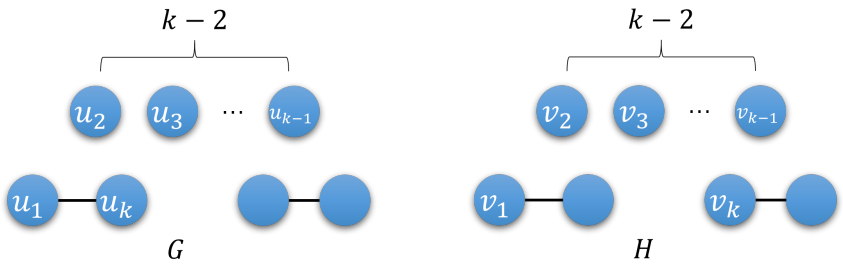

Figure 1: Example of non-isomorphic tuples.

A.2 Proof of Proposition 3

We first restate Proposition 3 here.

Proposition 3. *Suppose $i, p \geq 0, j, q > 0$ are integers. A p-q-NL$_{\mathrm{E}}$ can distinguish any non-isomorphic node tuples that a i-j-NL$_{\mathrm{E}}$ can distinguish, if and only if $p+q \geq i+j$.*

Proof. The proof is straightforward. Since the discriminative power of i-j-NL$_{\mathrm{E}}$ is strictly captured by the $(i+j)$-order isomorphism, the proof is also built on this fact.

2→1: We first prove that $p+q \geq i+j \Rightarrow p$-$q$-NL$_{\mathrm{E}}$ distinguishes any non-isomorphic node tuples that a i-j-NL$_{\mathrm{E}}$ distinguishes. Given $G=(\mathcal{V}_G, \mathcal{E}_G), H=(\mathcal{V}_H, \mathcal{E}_H)$ and $\mathbf{u}=(u_1,...,u_k) \in \mathcal{V}_G^k, \mathbf{v}=(v_1,...,v_k) \in \mathcal{V}_H^k)$, we first prove the situation where $p+q=i+j+1$.

- if $k=i+j$: Suppose $\mathbf{u}$ and $\mathbf{v}$ get different representations by i-j-NL$_{\mathrm{E}}$. By pooling the representations, p-q-NL$_{\mathrm{E}}$ computes the representation of $\mathbf{u}$ as $\left\{\left\{\boldsymbol{h}_{(u_1,...,u_k,w)} \mid w \in \mathcal{V}\right\}\right\}_G$. Then,

$$
\begin{aligned}
& \mathbf{u} \text{ and } \mathbf{v} \text{ get different representations by } i\text{-}j\text{-NL}_{\mathrm{E}} \\
\Longrightarrow\ & (\mathbf{u}, G) \ncong (\mathbf{v}, H) \\
\Longrightarrow\ & \forall w_1 \in \mathcal{V}_G, w_2 \in \mathcal{V}_H, ((u_1,...,u_k,w_1), G \ncong (v_1,...,v_k,w_2)) \\
\Longrightarrow\ & \forall w_1 \in \mathcal{V}_G, w_2 \in \mathcal{V}_H, \boldsymbol{h}_{(u_1,...,u_k,w_1)} \neq \boldsymbol{h}_{(u_1,...,u_k,w_2)} \\
\Longrightarrow\ & \left\{\left\{\boldsymbol{h}_{(u_1,...,u_k,w)} \mid w \in \mathcal{V}_G\right\}\right\} \neq \left\{\left\{\boldsymbol{h}_{(v_1,...,v_k,w)} \mid w \in \mathcal{V}_H\right\}\right\} \\
\Longrightarrow\ & p\text{-}q\text{-NL}_{\mathrm{E}} \text{ distinguishes } \mathbf{u}, \mathbf{v}.
\end{aligned}
$$

- if $k \neq i+j$: From the above deduction we can see that for any pair of $(i+j)$-tuples, if i-j-NL$_{\mathrm{E}}$ assigns different representations for them, then so does p-q-NL$_{\mathrm{E}}$. Obviously, this leads to the fact that p-q-NL$_{\mathrm{E}}$ is always more expressive than i-j-NL$_{\mathrm{E}}$.

Note that when $p+q=i+j$ the result naturally holds. Then by proof by induction we have proved all situations for $p+q \geq i+j$.

¬2→ ¬1: Similar as before, we first assume $p+q=i+j-1$. We build graphs $G=(\mathcal{V}_G, \mathcal{E}_G), H=(\mathcal{V}_H, \mathcal{E}_H)$ and $\mathbf{u} \in \mathcal{V}_G^k, \mathbf{v} \in \mathcal{V}_H^k$ such that i-j-NL$_{\mathrm{E}}$ assigns different representations for $\mathbf{u}, \mathbf{v}$. p-q-NL$_{\mathrm{E}}$ also computes the representation of $\mathbf{u}$ by aggregation as $\left(\boldsymbol{h}_{(u_1,...,u_{k-1})}, \boldsymbol{h}_{(u_2,...,u_k)}\right)$. The graphs are constructed as in Figure 1. It is easy to see that $((u_1,...,u_{k-1}), G) \simeq ((v_1,...,v_{k-1}), H)$ and $((u_2,...,u_k), G) \simeq ((v_2,...,v_k), H)$ but clearly $(\mathbf{u}, G)$ and $(\mathbf{v}, H)$ are not isomorphic.

With proof by induction, the direction ¬2→ ¬1 is proved. □

A.3 Proof of Corollary 4

We first restate Corollary 4 here.

Corollary 4. *Given the conditions in Theorem 2 hold. Then, we have*

$$G_{(\mathbf{u})} \simeq H_{(\mathbf{v})} \iff (\mathbf{u}, G) \simeq (\mathbf{v}, H),$$

where $G_{(\mathbf{u})}, H_{(\mathbf{v})}$ are node labeling induced graphs by labeling $\mathbf{u}$ and $\mathbf{v}$ respectively.

Proof. Corollary 4 is a special situation of the proof step 1→2 in the proof of Theorem 2.

$$\begin{aligned}
& G_{(\mathbf{u})} \simeq H_{(\mathbf{v})} \\
\iff\ & \exists \text{ isomorphism } \pi, \forall w \in \mathcal{V}_G :\ L(w \mid \mathbf{u}, G) = L(\pi(w) \mid \mathbf{v}, H) \\
\iff\ & \exists \text{ isomorphism } \pi(\mathbf{u}) = \mathbf{v} \\
\iff\ & (\mathbf{u}, G) \simeq (\mathbf{v}, H).
\end{aligned}$$

□

A.4 Proof of Lemma A.1

Proof. Proving Lemma A.1 is equivalent to showing that for any graphs G, H, the target tuples $\mathbf{u}, \mathbf{v}$ and G^k, H^k, $(\mathbf{u}, G) \simeq (\mathbf{v}, H) \iff (\mathbf{u}, G^k) \simeq (\mathbf{v}, H^k)$.

1→2: There exists an isomorphism π from G to H, such that $\pi(\mathbf{u}) = \mathbf{v}$. We assume a k-order permutation π^k which satisfies $\pi^k(w_1, ..., w_k) = (\pi(w_1), ..., \pi(w_k))$. Then we prove that π^k is a valid isomorphism from G^k to H^k. For all $\mathbf{w} \in \mathcal{V}_G^k$, we have

$$\begin{aligned}
& \pi^k(\mathcal{N}(\mathbf{w})) \\
=& \left\{\left\{(\pi^k(i, ..., w_k), ..., \pi^k(w_1, ..., i)) \mid i \in \mathcal{V}_G\right\}\right\} \\
=& \{\{((\pi(i), ..., \pi(w_k)), ..., (\pi(w_1), ..., \pi(i))) \mid i \in \mathcal{V}_G\}\} \\
=& \mathcal{N}(\pi(\mathbf{w})).
\end{aligned}$$

Note that π is structure-preserving: this indicates that for all $(w_1, ..., w_k)$, the label of the corresponding $(\pi(w_1), ..., \pi(w_k))$ must also be the same. Since we also have $\pi(\mathbf{u}) = \mathbf{v}$, we have $(\mathbf{u}, G^k) \simeq (\mathbf{v}, H^k)$.

2→1:

$$\begin{aligned}
& (\mathbf{u}, G^k) \simeq (\mathbf{v}, H^k) \\
\implies\ & \exists \pi^k : \mathcal{V}_G^k \to \mathcal{V}_H^k, \forall \mathbf{w} \in \mathcal{V}_G^k : \pi^k(\mathbf{u}) = \mathbf{v}, \pi^k(\mathcal{N}(\mathbf{w})) = \mathcal{N}(\pi^k(\mathbf{w})).
\end{aligned}$$

From $\pi^k(\mathcal{N}(\mathbf{w})) = \mathcal{N}(\pi^k(\mathbf{w}))$ we can further deduce

$$\begin{aligned}
& \pi^k(\mathcal{N}(\mathbf{w})) = \mathcal{N}(\pi^k(\mathbf{w})) \\
\implies\ & \left\{\left\{(\pi^k(i, ..., w_k), ..., \pi^k(w_1, ..., i)) \mid i \in \mathcal{V}_G\right\}\right\} = \\
& \left\{\left\{\left((j, \pi^k(\mathbf{w})_2, ..., \pi^k(\mathbf{w})_k), ..., (\pi^k(\mathbf{w})_1, ...j)\right) \mid j \in \mathcal{V}_H\right\}\right\}.
\end{aligned}$$

Now we show that π^k must can be expressed by another $\pi : \mathcal{V}_G \to \mathcal{V}_H$ such that $\pi^k(w_1, ..., w_k) = (\pi(w_1), ..., \pi(w_k))$. First from the above equation we can directly observe that $\pi^k(w_1, ..., w_{k-1}, i)_{:k-1} = \pi^k(w_1, ..., w_{k-1}, w_k)_{:k-1}$ for all $w_1, ..., w_k, i \in \mathcal{V}_G$. This directly indicates that we can break π^k into two parts: $\pi^k(w_1, ..., w_k) = (\pi^{k-1}(w_1, ..., w_{k-1}), \pi_k(w_k))$ where $\pi_k : \mathcal{V}_G \to \mathcal{V}_H$. The same way, by substituting the result into $\pi^k(w_1, ..., i, w_k)_{:k-2} = \pi^k(w_1, ..., w_{k-1}, w_k)_{:k-2}$ and continuing we can finally obtain $\pi^k(w_1, ..., w_k) = (\pi_1(w_1), ..., \pi_k(w_k))$. Next, we need to show that $\pi_1, ..., \pi_k$ are all equivalent. By substituting into $\pi^k(\mathcal{N}(\mathbf{w})) = \mathcal{N}(\pi^k(\mathbf{w}))$ we have

$$\begin{aligned}
& \left\{\left\{(\pi^k(i, ..., w_k), ..., \pi^k(w_1, ..., i)) \mid i \in \mathcal{V}_G\right\}\right\} = \\
& \left\{\left\{\left((j, \pi^k(\mathbf{w})_2, ..., \pi^k(\mathbf{w})_k), ..., (\pi^k(\mathbf{w})_1, ...j)\right) \mid j \in \mathcal{V}_H\right\}\right\} \\
\implies\ & \{\{((\pi_1(i), ..., \pi_k(w_k)), ..., (\pi_1(w_1), ..., \pi_k(i))) \mid i \in \mathcal{V}_G\}\} = \\
& \{\{((j, ..., \pi_k(w_k)), ..., (\pi_1(w_1), ..., j)) \mid j \in \mathcal{V}_H\}\}.
\end{aligned}$$

Therefore, for all $i \in \mathcal{V}_G$, there must exists a corresponding $j \in \mathcal{V}_H$ such that $\pi_1(i) = \pi_2(i) = ... = \pi_k(i) = j$. As a result, we proved that there exists $\pi : \mathcal{V}_G \to \mathcal{V}_H$ such that $\pi^k(w_1, ..., w_k) =$

$(\pi(w_1), ..., \pi(w_k))$. Next we need to show that π is an isomorphism from G to H. This is easily done by noticing that for all $w_1, ..., w_k \in \mathcal{V}_G$, the structures between $w_1, ..., w_k$ and $\pi(w_1), ..., \pi(w_k)$ must be the same, otherwise $(w_1, ..., w_k)$ and $(\pi(w_1), ..., \pi(w_k))$ would have different labels. Therefore, for any w_1, w_2 we have $(w_1, w_2) \in \mathcal{E}_G \iff (\pi(w_1), \pi(w_2)) \in \mathcal{E}_H$, and π is an isomorphism from G to H. Together, we have $(\mathbf{u}, G) \simeq (\mathbf{v}, H)$. □

B Proof of Theorem 5 and Proposition 6, 7, 8

B.1 Proof of Theorem 5

We first restate Theorem 5 here.

Theorem 5. *Given two graphs $G = (\mathcal{V}_G, \mathcal{E}_G), H = (\mathcal{V}_H, \mathcal{E}_H)$, and let $\mathbf{u} = (u_1, ..., u_k) \in \mathcal{V}_G^k, \mathbf{v} = (v_1, ..., v_k) \in \mathcal{V}_H^k$ where k is a fixed value to be the target tuples we want to learn from G and H respectively. Then, for any i, j satisfying $i > 0, j > 0, i + j = k$, the expressive power of $(i-1)$-$(j+1)$-$\mathrm{NL_{MP}}$ is strictly higher than i-j-$\mathrm{NL_{MP}}$, that is:*

- *There is a i-j-$\mathrm{NL_{MP}}$ that distinguishes $\mathbf{u}$ and $\mathbf{v} \Longrightarrow$ There is a $(i-1)$-$(j+1)$-$\mathrm{NL_{MP}}$ that distinguishes $\mathbf{u}$ and $\mathbf{v}$.*
- *There exists $G, H, \mathbf{u}, \mathbf{v}$ such that a $(i-1)$-$(j+1)$-$\mathrm{NL_{MP}}$ distinguishes $\mathbf{u}$ and $\mathbf{v}$ but i-j-$\mathrm{NL_{MP}}$ cannot.*

Proof. We prove the two results separately. To prove the first result, we first recall the procedure of 0-k-$\mathrm{NL_{MP}}$ (or k-FWL). Given $G = (\mathcal{V}, \mathcal{E})$, at each layer it evaluates

$$\mathrm{Col}^{(l+1)}(u_1, ..., u_k) = \mathrm{Hash}\Big(\mathrm{Col}^{(l)}(u_1, ..., u_k), \Big\{\Big\{\Big(\mathrm{Col}^{(l)}(v, u_2, ..., u_k), \mathrm{Col}^{(l)}(u_1, v, ..., u_k), ..., \mathrm{Col}^{(l)}(u_1, ..., u_{k-1}, v)\Big) \mid v \in \mathcal{V}\Big\}\Big\}\Big).$$

If we consider a fragment of the above procedure where we replace the first i colors in each neighbor with the initial colors

$$\mathrm{Col}^{(l+1)}(u_1, ..., u_k) = \mathrm{Hash}\Big(\mathrm{Col}^{(l)}(u_1, ..., u_k), \Big\{\Big\{\Big(\mathrm{Col}^{(0)}(v, u_2, ..., u_k), ..., \mathrm{Col}^{(0)}(u_1, ..., u_{i-1}, v, u_{i+1}, ..., u_k), \mathrm{Col}^{(l)}(u_1, ..., u_i, v, u_{i+2}, ..., u_k), ..., \mathrm{Col}^{(l)}(u_1, ..., u_{k-1}, v)\Big) \mid v \in \mathcal{V}_G\Big\}\Big\}\Big).$$

Clearly, this variant is less expressive than the original one. We can rewrite this variant to make it strictly corresponded to i-$(k-i)$-$\mathrm{NL_{MP}}$. First, note that the first i variables in $\mathrm{Col}^{(l)}$ and $\mathrm{Col}^{(l+1)}$ are the same. This inspires that we can rewrite the equations when we fix the first i variables $u_1, ..., u_i$ as

$$f^{(l+1)}(u_{i+1}, ..., u_k) = \phi\left(f^{(l)}(u_{i+1}, ..., u_k), \left\{\left\{\left(f^{(l)}(v, ..., u_k), ..., f^{(l)}(u_{i+1}, ..., v)\right) \mid v \in \mathcal{V}\right\}\right\}\right),$$

where we initialize $f^{(0)}$ as $f^{(0)}(u_{i+1}, ..., u_k) = \mathrm{Col}^{(0)}(u_1, ..., u_k)$. The computation of $f^{(l)}$ is exactly the same as i-$(k-i)$-$\mathrm{NL_{MP}}$ when we apply node labeling on $u_1, ..., u_k$. Since i-j-$\mathrm{NL_{MP}}$ is clearly corresponded to a more "small" fragment of the above equations compared with $(i-1)$-$(j+1)$-$\mathrm{NL_{MP}}$, we have proved the first result.

To prove the second result, we need to utilize the results from Grohe and Otto [3]. Cai et al. [1] designed a construction of a series of pairs of non-isomorphic graph CFI(k) such that $(k-1)$-FWL fails to distinguish them. Further more, Grohe and Otto [3] proposed a variant of CFI graphs and showed that there are graphs such that $(k-1)$-FWL cannot distinguish them but k-FWL can. Our proof is based on this result. Suppose G, H are non-isomorphic graphs than cannot by distinguished by j-FWL but are distinguished by $(j+1)$-FWL. We first obtain the following property of FWL

Lemma B.1. *If two graphs $G_1 = (\mathcal{V}_{G_1}, \mathcal{E}_{G_1}), H_1 = (\mathcal{V}_{H_1}, \mathcal{E}_{H_1})$ are not distinguished by k-FWL, $G_2 = (\mathcal{V}_{G_2}, \mathcal{E}_{G_2}), H_2 = (\mathcal{V}_{H_2}, \mathcal{E}_{H_2})$ are also not distinguished by k-FWL, then we let $G = (\mathcal{V}_G, \mathcal{E}_G)$ where $\mathcal{V}_G = \mathcal{V}_{G_1} \cup \mathcal{V}_{G_2}, \mathcal{E}_G = \mathcal{E}_{G_1} \cup \mathcal{E}_{G_2}$ and $H = (\mathcal{V}_H, \mathcal{E}_H)$ where $\mathcal{V}_H = \mathcal{V}_{H_1} \cup \mathcal{V}_{H_2}, \mathcal{E}_H = \mathcal{E}_{H_1} \cup \mathcal{E}_{H_2}$. G and H are still not distinguished by k-FWL.*

It is easy to prove Lemma B.1 by induction. Suppose φ is a FOC_{k+1} formula. Then,

- If φ is of the form $\varphi := C$, i.e. node colors or edges, then obviously φ produces the same results on G, H.
- If $\varphi := \varphi' \wedge \varphi''$ and φ', φ'' produce the same results on G, H, then obviously φ also produces the same results on G, H.
- if $\varphi := \neg\varphi'$, the situation is the same as before.
- if $\varphi := \exists^N \varphi'$, and φ' produces the same results on G, H. We let m, n to be the number of distinct groundings of φ' on G_1, G_2 respectively. Then, the number of distinct groundings of φ' on H_1, H_2 are also m, n. (otherwise a formula $\varphi'' = (\exists^m \varphi') \wedge \neg(\exists^{m+1} \varphi')$ differs on G_1, H_1 or $\varphi'' = (\exists^n \varphi') \wedge \neg(\exists^{n+1} \varphi')$ differs on G_2, H_2.) Therefore, on both G, H φ still produces the same results.

Therefore, any FOC_{k+1} formula, if produces the same results on G_1, H_1 and G_2, H_2, also produces the same results on G, H. As a result k-FWL cannot distinguishes G, H.

With Lemma B.1 we can construct counterexamples for i-j-$\mathrm{NL}_{\mathrm{MP}}$ and $(i-1)$-$(j+1)$-$\mathrm{NL}_{\mathrm{MP}}$. Suppose G, H cannot be distinguished by j-FWL but are distinguished by $(j+1)$-FWL. We add k isolated nodes $u_1, ...u_k$ to G and $v_1, ...v_k$ to H, obtaining G', H'. Furthermore, the labels of $u_1, ..., u_k$ are distinct and different from the rest of the nodes, and we let the label of v_l to be the same with u_l for $l \in [k]$. We then apply i-j-$\mathrm{NL}_{\mathrm{MP}}$ and $(i-1)$-$(j+1)$-$\mathrm{NL}_{\mathrm{MP}}$ to learn the representation of $\mathbf{u} = (u_1, ..., u_k) \in \mathcal{V}_G^k$ and $\mathbf{v} = (v_1, ..., v_k) \in \mathcal{V}_H^k$. Apparently from Lemma B.1 we know that i-j-$\mathrm{NL}_{\mathrm{MP}}$ cannot distinguish them. Since a $(j+1)$-FWL can distinguish G and H, it also distinguishes G' and H'. This indicates that there is a FOC_{j+1} formula φ such that $G' \vDash \varphi$ and $H' \nvDash \varphi$. By letting $\psi(x_1, ..., x_j) := \varphi()$ we can see that $\psi(u_{i+1}, ..., u_k) \neq \psi(v_{i+1}, ..., v_k)$. Therefore, the $j+1$-FWL also assign different colors to $(u_{i+1}, ..., u_k)$ and $(v_{i+1}, ..., v_k)$. As a result $(i-1)$-$(j+1)$-$\mathrm{NL}_{\mathrm{MP}}$ distinguishes them. □

B.2 Proof of Proposition 6

To prove Proposition 6 is to construct a counterexample for i-j-$\mathrm{NL}_{\mathrm{MP}}$ and p-q-$\mathrm{NL}_{\mathrm{MP}}$. Specially, from the proof of Theorem 5 we know that we only need to construct a counterexample for k-1-$\mathrm{NL}_{\mathrm{MP}}$ and 0-k-$\mathrm{NL}_{\mathrm{MP}}$ for any k. We first introduce a series of graph pairs proposed by Grohe and Otto [3].

Let $\mathcal{K} = (\mathcal{V}, \mathcal{E})$ be the complete graph on k nodes. We further assume the nodes in $\mathcal{K}$ are $v_1, ..., v_k$. We define the construction of a structure $\mathcal{X}(\mathcal{K})$, which is the ***CFI-companions*** of $\mathcal{K}$, as follows. It is convenient to call the nodes from $\mathcal{X}(\mathcal{K})$ *vertices*, to distinguish them from the nodes of $\mathcal{K}$.

For every $v \in \mathcal{V}$, the graph $\mathcal{X}(\mathcal{K})$ has a vertex v^S, where S is a subset of $\mathcal{E}(v)$ of even cardinality. We use $\mathcal{E}(v)$ to denote the edges connected with v. For every edge $e \in \mathcal{E}$, the graph $\mathcal{X}$ has two vertices e^0, e^1. Vertices of the form v^S are called node vertices and vertices e^i edge vertices. Formally, the set of vertices in $\mathcal{X}(\mathcal{K})$ is

$$\{v^S \mid v \in \mathcal{V}, S \subseteq \mathcal{E}(v) \text{ such that } |S| \equiv 0 \bmod 2\} \cup \{e^0, e^1 \mid e \in \mathcal{E}\}. \tag{1}$$

The edges of $\mathcal{X}(\mathcal{K})$ link node vertices and edge vertices according to

$$(v^S, e_i) \text{ is an edge if } \begin{cases} i = 1 \text{ and } e \in S \\ i = 0 \text{ and } e \notin S \end{cases} \tag{2}$$

In $\mathcal{X}(\mathcal{K})$, the vertices of the form v^S are colored C_v, and the vertices of the form e^i are colored C_e. After defining $\mathcal{X}(\mathcal{K})$, we also define its variant $\widehat{\mathcal{X}}(\mathcal{K})$ whose node set and edge set are the same except that for the node v_1 we take nodes v_1^S from the subsets of $\mathcal{E}(v_1)$ of odd cardinality. It is proved by Grohe and Otto [3] that $\mathcal{X}(\mathcal{K}), \widehat{\mathcal{X}}(\mathcal{K})$ are distinguished by $(k-1)$-FWL but not $(k-2)$-FWL.

Before we proceed, we would like to provide a $k = 3$ example to illustrate $\mathcal{X}(\mathcal{K})$ and $\widehat{\mathcal{X}}(\mathcal{K})$ in Figure 2. The graphs $\mathcal{X}(\mathcal{K})$ and $\widehat{\mathcal{X}}(\mathcal{K})$ are distinguished by 2-FWL but not 1-WL.

Now we proceed to prove Proposition 6. We first restate it here.

Proposition 6. *Suppose $i, p \geq 0, j, q > 0$ are integers. There is a p-q-$\mathrm{NL_{MP}}$ that distinguishes any non-isomorphic node tuples that a i-j-$\mathrm{NL_{MP}}$ can distinguish if and only if $p + q \geq i + j$ and $q \geq j$. Otherwise, there exists non-isomorphic node tuples that a p-q-$\mathrm{NL_{MP}}$ cannot distinguish but a i-j-$\mathrm{NL_{MP}}$ can distinguish.*

Proof. We first show that there are graphs distinguished by $(k-2)$-1-$\mathrm{NL_{MP}}$ but not 0-$(k-2)$-$\mathrm{NL_{MP}}$. We construct CFI graphs $\mathcal{X}(\mathcal{K})$ and $\widehat{\mathcal{X}}(\mathcal{K})$. From Grohe and Otto [3] we know that $\mathcal{X}(\mathcal{K})$ and $\widehat{\mathcal{X}}(\mathcal{K})$ cannot be distinguished by 0-$(k-2)$-$\mathrm{NL_{MP}}$, whose expressive power is bounded by $(k-2)$-FWL.

Next we show that $\mathcal{X}(\mathcal{K})$ and $\widehat{\mathcal{X}}(\mathcal{K})$ can be distinguished by a $(k-2)$-1-$\mathrm{NL_{MP}}$. To do so we need to introduce the concept of *pebble games*[3, 1]. The readers are welcome to check [3] for a more detailed introduction. Given two structures $\mathcal{A}, \mathcal{B}$, the *bijective k-pebble game* is played by two players by placing k pairs of pebbles on a pair of structures $\mathcal{A}, \mathcal{B}$. The rounds of the game are as follows. Player **I** picks up one of his pebbles, and player **II** picks up her corresponding pebble. Then player **II** chooses a bijection f between $\mathcal{A}$ and $\mathcal{B}$ (if no such bijection exists player **II** immediately loses). Then player **I** places his pebble on an element a of $\mathcal{A}$, and player **II** places her pebble on $f(a)$. After each round there is a subset $p \subseteq \mathcal{A} \times \mathcal{B}$ consisting of the at most k pairs of elements corresponding to the pebbles placed. Player **II** wins a play if every position p is a local isomorphism.

Theorem B.2. *(*Cai et al. [1]*) $\mathcal{A} \equiv^k_C \mathcal{B}$ if and only if player **II** has a winning strategy for the bijective k-pebble game on $\mathcal{A}, \mathcal{B}$.*

Theorem B.2 indicates that $(k-1)$-FWL can distinguish $\mathcal{A}, \mathcal{B}$ if and only if player **II** has a winning strategy for the bijective k-pebble game on $\mathcal{A}, \mathcal{B}$. However, although it justifies $(k-1)$-FWL, it has nothing to do with the $(k-2)$-1-$\mathrm{NL_{MP}}$ here. We propose a variant of the bijective k-pebble game, namely restricted bijective k-i-pebble game, described as follows. Given two structures $\mathcal{A}, \mathcal{B}$, the *restricted bijective k-i-pebble game* is also played by two players by placing k pairs of pebbles on a pair of structures $\mathcal{A}, \mathcal{B}$. The rounds of the game are as follows. Player **I** picks up one of his pebbles, and player **II** picks up her corresponding pebble. Then player **II** chooses a bijection f between $\mathcal{A}$ and $\mathcal{B}$ (if no such bijection exists player **II** immediately loses). Then player **I** places his pebble on an element a of $\mathcal{A}$, and player **II** places her pebble on $f(a)$. The difference is, i pairs of the pebbles are static, as when the players place these pebbles on the elements of $\mathcal{A}, \mathcal{B}$, they can no longer pick and replace them on other elements. Then, we have

Theorem B.3. *There is a $(k-2)$-1-$\mathrm{NL_{MP}}$ that distinguishes the $(k-2)$-tuples $\mathbf{u}, \mathbf{v}$ from $\mathcal{A}, \mathcal{B}$ if and only if player **II** has a winning strategy for the restricted bijective k-(k-2)-pebble game on $\mathcal{A}, \mathcal{B}$ which initially places $(k-2)$ static pebbles on $\mathbf{u}, \mathbf{v}$, if $\mathcal{A}, \mathcal{B}$ are connected graphs.*

Theorem B.3 is proved in the next section. With Theorem B.3 we can now prove that the graphs $\mathcal{X}(\mathcal{K})$ and $\widehat{\mathcal{X}}(\mathcal{K})$ can be distinguished by $(k-2)$-1-$\mathrm{NL_{MP}}$. The prove steps are exactly the same as in [3], as the steps in [3] naturally follow the constraints of the bijective k-$(k-2)$-pebble game.

We give a winning strategy for player **I** in the bijective k-$(k-2)$-pebble game. In the first $k-1$ rounds of the game, player **I** picks his $k-1$ pebbles on $v_2^\emptyset, ..., v_k^\emptyset$ and suppose $p(v_i^\emptyset) = v_i^{S_i}$ is the corresponding position for some sets S_i. That is,

$$p = \{v_2^\emptyset v_2^{S_2}, ..., v_k^\emptyset v_k^{S_k}\}.$$

We now assume that the pebbles at $v_3^\emptyset, ..., v_k^\emptyset$ are static. Therefore $v_3^\emptyset, ..., v_k^\emptyset$ compose all $k-2$ static pebbles and the pebble at $v_2^\emptyset$ is still movable. In the next round of the game player **I** starts by selecting this pebble, and places it on e_{12}^0. In the next round, player **I** starts by selecting the pebble on e_{12}^0 and places it on $v_1^\emptyset$. It is proved by Grohe and Otto [3] that player **I** wins at this time.

□

B.3 Proof of Proposition 7

We first restate Proposition 7 here.

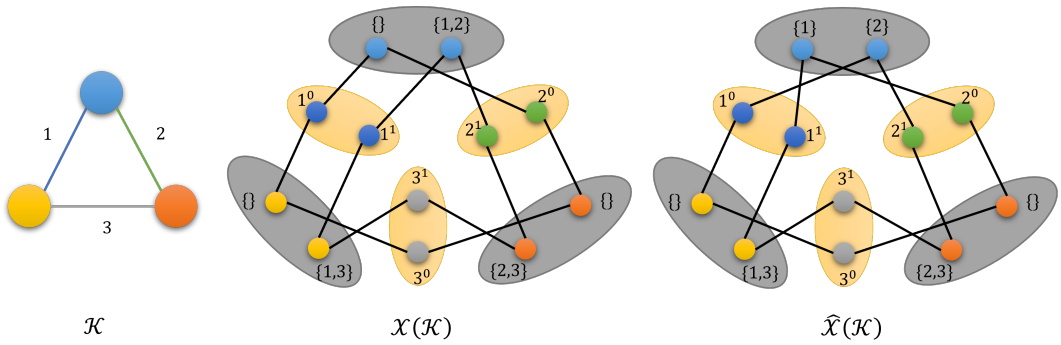

Figure 2: Example of non-isomorphic graphs.

Proposition 7. *For any k-tuples $\mathbf{u}, \mathbf{v}$ and $i+j=k$: There is a i-j-$\mathrm{NL}_{\mathrm{MP}}$ that distinguishes $\mathbf{u}$ and $\mathbf{v}$ $\iff$ There is a $\mathrm{FOC}_{k+1,i}$ formula that distinguishes $\mathbf{u}$ and $\mathbf{v}$.*

Proof. To prove Proposition 7 we can first use the results from [1].

Theorem B.4. (Cai et al. [1]) *Let G, H be a pair of colored graphs and let $\mathbf{u} \in \mathcal{V}_G^k, \mathbf{v} \in \mathcal{V}_H^k$ be k-tuples. The following are equivalent:*

- *k-FWL assigns the same color for $\mathbf{u}, \mathbf{v}$.*
- *All FOC_{k+1} produce the same result for $\mathbf{u}, \mathbf{v}$.*

The above equivalence can be extend to node labeling situations. Formally, given $G=(\mathcal{V},\mathcal{E})$ and $\mathbf{u}=(u_1,...,u+k) \in \mathcal{V}^k$, and let $i+j=k$. Then, applying a i-j-$\mathrm{NL}_{\mathrm{MP}}$ is essentially first assigning additional predicates $\mathrm{IsTarget}_l(\cdot)$ for $l \in [i]$. Further more, we have $\mathrm{IsTarget}_l(v) = \mathbf{1}_{v=u_l}$. Then, it applies a j-MPNN on the augmented graph $G_{(\mathbf{u}_{:i})}$ whose expressive power is bounded by j-FWL, and thus is be described by FOC_{j+1}.

Now we need to prove the equivalence between FOC_{j+1} on the augment graphs $G_{(\mathbf{u}_{:i})}$ and $\mathrm{FOC}_{k+1,i}$ on the original graph G. Specially, we want to prove

1. Given any FOC_{j+1} formula φ on $G_{(\mathbf{u}_{:i})}$, there is a corresponding $\mathrm{FOC}_{k+1,i}$ formula ψ on G that expresses φ.
2. Given any $\mathrm{FOC}_{k+1,i}$ formula ψ on G, there exists $\mathbf{u}$ such that a FOC_{j+1} formula φ on $G_{(\mathbf{u}_{:i})}$ expresses ψ.

1. It is simple to create the corresponding ψ. We only need to replace all unary predicates $C(v)$ with an invented one $C'(x_1,x_2,...,x_i,v)$, and replace all binary predicates $E'(v,w)$ with an invented one $E'(x_1,x_2,...,x_i,v,w)$. We define

$$C'(x_1,x_2,...,x_i,v) := \begin{cases} \mathbf{1}_{x_l=v}, & \text{if } C(v) := \mathrm{IsTarget}_l(v), \\ C(v), & \text{else.} \end{cases}$$
$$E'(x_1,x_2,...,x_i,v,w) := E(v,w).$$

Therefore, we effectively remove all emergence of the additional predicates $\mathrm{IsTarget}_l(\cdot)$.

2. Suppose the target tuple is fixed $\mathbf{u}$. This indicates that all predicates in ψ share the same first i variables $\mathbf{u}_{:i}$. The same way, we replace predicates in ψ with our invented ones. Note that a predicate $P(u_1,...,u_k)$ is defined by any permutation-invariant functions over the subgraph induced by $u_1,...,u_k$. Therefore we initialize $P'(u_{i+1},...,u_k) = P(u_1,...,u_k)$ for all P. Since all predicates in ψ share the same first i variables $\mathbf{u}_{:i}$, we can replace all emergence of predicates in ψ and eliminate the first i variables in this way. We have constructed the corresponding FOC_{j+1} formula φ on $G_{\mathbf{u}_{:i}}$. □

B.4 Proof of Proposition 8

We first restate Proposition 8 here.

Proposition 8 *Given a graph $G = (\mathcal{V}_G, \mathcal{E}_G)$, we run 1-WL to assign node colors for G, denoted as $\mathrm{Col}(\cdot)$. The color of a path $P = (w_1, ..., w_d)$ is obtained by hashing the corresponding node color sequence $\mathrm{Col}(P) = \mathrm{Hash}(\mathrm{Col}(w_1), ..., \mathrm{Col}(w_d))$. Then, the correlation between two nodes (u, v), as discussed above, is expressed by all paths between u, v,*

$$\mathrm{Corr}(u, v) = \mathrm{Hash}\left(\{\{\mathrm{Col}(P) \mid P \in \mathrm{Paths}(u, v)\}\}\right).$$

Given two connected graphs $G = (\mathcal{V}_G, \mathcal{E}_G)$ and $H = (\mathcal{V}_H, \mathcal{E}_H)$. Let $\mathbf{u} = (u_1, ..., u_k) \in \mathcal{V}_G^k, \mathbf{v} = (v_1, ..., v_k) \in \mathcal{V}_H^k$ be the target node tuples. If there exists $1 \le p, q \le k$ such that $\mathrm{Corr}(u_p, u_q) \neq \mathrm{Corr}(v_p, v_q)$, then any i-j-$\mathrm{NL}_{\mathrm{MP}}$ with $i + j \ge k$ with sufficient injective layers always distinguishes $\mathbf{u}$ and $\mathbf{v}$.

Proof. Since the weakest i-j-$\mathrm{NL}_{\mathrm{MP}}$ variant given fixed $k = i + j$ is the $(k-1)$-1-$\mathrm{NL}_{\mathrm{MP}}$, we only prove the result for $(k-1)$-1-$\mathrm{NL}_{\mathrm{MP}}$.

Suppose $\mathrm{Corr}(u_p, u_q) \neq \mathrm{Corr}(v_p, v_q)$. Without loss of generality we may assume that u_p, v_p are with node labeling. By the definition of Corr we know that there are at least one type of color of paths $C = \mathrm{Hash}(C_1, ..., C_d)$ such that the number of C-colored paths between (u_p, u_q) are different from that between (v_p, v_q). We manually construct a i-j-$\mathrm{NL}_{\mathrm{MP}}$ model that distinguishes u_q, v_q from $G_{(\mathbf{u})}, H_{(\mathbf{v})}$. First, we add sufficient injective layers to the model, but these layers are not aware of the additional node labels. In other words, we compute the 1-WL color Col in this way. Then, we let the next layers to detect the paths as

$$\mathrm{Layer}^{(1)}(x) = (\mathrm{Col}(x), \mathbf{1}(x \text{ is the } p\text{-th node labeling}) \cdot \mathbf{1}(\mathrm{Col}(x) = C_1)),$$

where $\mathbf{1}$ is the indicator function. Similarly, we define the l layer for $l \le d$ as

$$\begin{aligned}
&\mathrm{ColLayer}^{(l)}(x) = \mathrm{Layer}^{(l-1)}(x)[0] = \mathrm{Col}(x),\\
&\mathrm{PathLayer}^{(l)}(x) = \mathrm{Layer}^{(l-1)}(x)[1],\\
&\mathrm{Layer}^{(l)}(x) = \left(\mathrm{ColLayer}^{(l)}(x), \mathbf{1}\left(\mathrm{ColLayer}^{(l)}(x) = C_l\right) \cdot \sum_{y \in \mathcal{N}(x)} \left(\mathrm{PathLayer}^{(l)}(x)\right)\right).
\end{aligned}$$

Therefore, $\mathrm{Layer}^{(\mathrm{d})}(x)$ counts the number of C-colored paths between x and u_p or v_p. Thus $\mathrm{Layer}^{(\mathrm{d})}(u_q) \neq \mathrm{Layer}^{(\mathrm{d})}(v_q)$. If $q = k$, then this directly indicates that our $(k-1)$-1-$\mathrm{NL}_{\mathrm{MP}}$ distinguishes them. If $q < k$, then we further add sufficient identical layers as follows.

$$\mathrm{IdLayer}^{(0)}(x) = \mathbf{1}(x \text{ is the } q\text{-th node labeling}) \odot \mathrm{Layer}^{(\mathrm{d})}(x)[1],$$

and

$$\mathrm{IdLayer}^{(l)}(x) = \max_{y \in \mathcal{V}(x)} \mathrm{IdLayer}^{(l-1)}(y).$$

After sufficient layers, $\mathrm{IdLayer}$ gives different results on u_k and v_k. □

B.5 Proof of Theorem B.3

Proof. Let G, H be a pair of connected graphs and let $\mathbf{u}, \mathbf{v}$ be the target k-tuples. Let $G_{(\mathbf{u})}, H_{(\mathbf{v})}$ be the corresponding node labeling induced graphs. Since G, H are connected, we need to show the following statements are equivalent.

1. 1-WL cannot distinguish $G_{(\mathbf{u})}, H_{(\mathbf{v})}$
2. 2-WL cannot distinguish $G_{(\mathbf{u})}, H_{(\mathbf{v})}$
3. No FOC_2 formula distinguishes $G_{(\mathbf{u})}, H_{(\mathbf{v})}$
4. Player **II** has a winning strategy for the $(k+2)$-k pebble game on G, H which initially places the k pairs of static pebbles on $\mathbf{u}, \mathbf{v}$.

$1 \iff 2 \iff 3$: $2 \iff 3$ is proved as a special case in Cai et al. [1]. Since G, H are connected, $1 \iff 2$ also holds.

$2 \Rightarrow 4$: Suppose after r iterations the 2-WL still assigns the same color to $G_{(\mathbf{u})}, H_{(\mathbf{v})}$. We instead prove the following statement:

- After $r + k$ iterations 2-WL gives $(x_1, y_1) \in \mathcal{V}^2_{G_{(\mathbf{u})}}$ and $(x_2, y_2) \in \mathcal{V}^2_{H_{(\mathbf{v})}}$ the same color $\implies$ Player **II** has a winning strategy for the $(k+2)$-k pebble game on G, H which initially places the k pairs of static pebbles on $\mathbf{u}, \mathbf{v}$ and place the other 2 pairs of static pebbles on (x_1, y_1) and (x_2, y_2) in r moves.

We assume W^r to be the color assignment of 2-WL at iteration $r + k$. Clearly player **I** can only chooses one of the pebbles on x_1, y_1. Without loss of generality suppose he picks up x_1. The player **II** answers with the bijective mapping that maps node pairs with the same W^{r-1} color, that is, $f(t_1) \in \{t_2 \mid W^{r-1}(t_1, y_1) = W^{r-1}(t_2, y_2)\}$. Note that such mapping must exist because

$$W^r(x, y) = \text{Hash}\left(W^{r-1}(x, y), \left\{\left\{W^{r-1}(x, z) \mid z \in \mathcal{V}_G\right\}\right\}, \left\{\left\{W^{r-1}(z, y) \mid z \in \mathcal{V}_G\right\}\right\}\right)$$

is the same for (x_1, y_1) and (x_2, y_2). No matter which node player **I** places his pebble on, player **II** places her pebble on the corresponding node. Player **II** has not yet lost: the structure between (t_1, y_1) and (t_2, y_2) must be the same, otherwise they have different W^{r-1} colors. The structure between $(t_1, \mathbf{u})$ and $(t_2, \mathbf{v})$ are also the same: This is because that at the start we added unique labels to the k-tuples $\mathbf{u}$ and $\mathbf{v}$. Therefore, after the first k rounds of 2-WL iterations if the subgraphs induced by $t_1, y_1, \mathbf{u}$ from $G_{(\mathbf{u})}$ and $t_2, y_2, \mathbf{v}$ from $H_{(\mathbf{v})}$ are different, (t_1, y_1) and (t_2, y_2) will also have different 2-WL colors. Now, since (t_1, y_1) and (t_2, y_2) have the same W^{r-1} colors, by induction on r we proved the above statement.

By further induction on r in the statement, we can prove $2 \Rightarrow 4$ because we have showed that for any node pairs the results of 2-WL and the pebble game are always consistent.

$\neg 3 \Rightarrow \neg 4$: Suppose for some FOC_2 formula φ, $G_{(\mathbf{u})} \models \varphi$ and $H_{(\mathbf{v})} \not\models \varphi$. If φ is a conjunction then $G_{(\mathbf{u})}, H_{(\mathbf{v})}$ must differs on at least one of the conjuncts, so we may assume φ is of the form $\exists^N x\psi$. Without loss of generality we assume the quantifier depth of φ is r. Note that there are total 2 free pairs of pebbles that can be placed to nodes, which exactly corresponds to the number of free variables in FOC_2 formula. Player **I** takes a pebble, corresponding to the variable x in φ. Player **II** must respond with a bijective mapping f. Since $G_{(\mathbf{u})} \models \varphi$ and $H_{(\mathbf{v})} \not\models \varphi$, we know that there are at least N nodes satisfying ψ in $G_{(\mathbf{u})}$ but less than N nodes satisfying ψ in $H_{(\mathbf{v})}$. Player **I** then picks the node w in $G_{(\mathbf{u})}$ such that $\psi(w)$ is true but $\psi(f(w))$ is false. By induction we can see that at quantifier depth 0 player **II** loses the game. □

C The algorithm complexities of i-j-NL_{MP}

In this section we derive the algorithm complexities of i-j-NL_{MP}.

C.1 The case of $j \geq 2$

We first study the situation where we assume $j \geq 2$. Suppose we are given a graph $G = (\mathcal{V}, \mathcal{E})$ with N nodes and M edges. Our model is a i-j-NL_{MP}, and we let $k = i + j$. Suppose we want to evaluate every k-tuple $\mathbf{u} \in \mathcal{V}^k$.

Obviously there are total N^k target tuples in the graph. For learning one tuple $\mathbf{u} \in \mathcal{V}^k$, we need to first assign node labeling to $\mathbf{u}_{:i}$, then apply a j-MPNN on $G_{(\mathbf{u}_{:i})}$. The numbers of nodes and edges in $G_{(\mathbf{u}_{:i})}$ are still N, M respectively. Since we assume the j-MPNN simulate the j-FWL, applying the j-MPNN on $G_{(\mathbf{u}_{:i})}$ and simultaneously learning representations all j-tuples requires $\mathcal{O}(N^j)$ space and $\mathcal{O}(N^{j+1})$ time. After this procedure, we actually learns all k-tuples with the same first i nodes $\mathbf{u}_{:i}$, so we only need to apply this procedure for N^i times to learn all representations for all k-tuples. Therefore, we need $\mathcal{O}(N^j)$ space and $\mathcal{O}(N^{i+j+1})$ time.

C.2 The case of $j = 1$

When $j = 1$ the time and space complexities of MPNN are $\mathcal{O}(M)$ and $\mathcal{O}(N)$ respectively. Therefore, the total process for computing all k-tuple takes $\mathcal{O}(N)$ space and $\mathcal{O}(MN^{i+j-1})$ time. As a result, the conclusion in Section 4.1 still holds.

D The Weisfeiler-Lehman Algorithms

In this section we briefly introduce the k-FWL graph isomorphism tests.

D.1 1-WL

The 1-WL test is also known as color refinement and shares similar message passing process with node-level GNNs. To begin with, each node v is assigned with a color c_v. The 1-WL test can then be summarized as follows:

Algorithm 1 1-WL test

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: the input graph; c_v for $v \in \mathcal{V}$: initial colors;
Output: the final colors;
1: $c_v^0 \leftarrow c_v$ for all $v \in \mathcal{V}$;
2: **repeat**
3: $\quad \forall v \in \mathcal{V}, c_v^{l+1} \leftarrow \mathrm{hash}(c_v^l, \{\{c_w^l \mid w \in N(v)\}\})$;
4: **until** $\forall v \in \mathcal{V}, c_v^{l+1} = c_v^l$;
5: **return** c_v^l for every $v \in \mathcal{V}$;

Here, $N(v)$ is the set of the neighbors of v in $\mathcal{G}$. The critical part is the hash function hash. It needs to be injective in order to fully express the discriminative power of the 1-WL test.

D.2 k-FWL

The k-FWL test is summarized as follows.

Algorithm 2 k-FWL test

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: the input graph; $c_{\mathbf{u}}$ for $\mathbf{u} \in \mathcal{V}^k$: initial colors;
Output: the final colors;
1: $c_{\mathbf{u}}^0 \leftarrow c_{\mathbf{u}}$ for all $\mathbf{u} \in \mathcal{V}^k$;
2: **repeat**
3: $\quad \forall \mathbf{u} \in \mathcal{V}^k, c_{\mathbf{u}}^{l+1} \leftarrow \mathrm{hash}(c_{\mathbf{u}}^l, \{\{c_{\mathbf{v}}^l \mid \mathbf{v} \in N(\mathbf{u})\}\})$;
4: **until** $\forall \mathbf{u} \in \mathcal{V}, c_{\mathbf{u}}^{l+1} = c_{\mathbf{v}}^l$;
5: **return** $c_{\mathbf{u}}^l$ for every $\mathbf{u} \in \mathcal{V}^k$;

The difference between the k-FWL test and the 1-WL is that we now assign a color for each node tuple $\mathbf{u}$ instead of a single node. Generally, the k-FWL test follows the same computation procedure with the 1-WL, but we need to redefine the neighbors, i.e. $N(\mathbf{u})$, of a node tuple $\mathbf{u}$. In the k-FWL test, we set each node pair $\mathbf{u}$ to have $|\mathcal{V}|$ neighbors, with the i-th neighbor being

$$((i, u_2, ..., u_k), (u_1, i, u_3, ..., u_k), ..., (u_1, ..., u_{k-1}, i)).$$

E Different node labeling methods

In this section we discuss about different node labeling methods. Generally, the methods we discuss here all follows the constraints in Section 2:

1. $L(u \mid \mathbf{v}, G) = L(\pi(u) \mid \mathbf{w}, H)$ holds for all $u \in \mathcal{V}_G \Rightarrow \pi(\mathbf{v}) = \mathbf{w}$,
2. π is an isomorphism from $(\mathbf{v}, G)$ to $(\mathbf{w}, H) \Rightarrow \forall u \in \mathcal{V}_G, L(u \mid \mathbf{v}, G) = L(\pi(u) \mid \mathbf{w}, H)$.

E.1 0-1 node labeling

We first introduce the basic 0-1 node labeling method introduced in SEAL. We modify the original 0-1 node labeling for node tuples here.

Given a graph $G = (\mathcal{V}, \mathcal{E})$ and suppose $\mathbf{u} = (u_1, ..., u_k) \in \mathcal{V}^k$ be the target tuple. We define the node labeling mechanism L to be

$$L(v \mid \mathbf{u}, G) = \text{Hash}(\mathbf{1}_{v=u_1}, \mathbf{1}_{v=u_2}, ..., \mathbf{1}_{v=u_k}),$$

where $\mathbf{1}$ is the indicator function. For example in the graph G in Figure 1, we can assign labels $1, 2, ..., k$ to the nodes $u_1, u_2, ..., u_k$ respectively and assign label 0 to the rest of the nodes.

Now we show that the above 0-1 node labeling satisfies the constraints. Given $G = (\mathcal{V}_G, \mathcal{E}_G), H = (\mathcal{V}_H, \mathcal{E}_H)$ and $\mathbf{v} \in \mathcal{V}_G^k, \mathbf{w} \in \mathcal{V}_H^k$,

1. We have
$$\begin{aligned} & L(u \mid \mathbf{v}, G) = L(\pi(u) \mid \mathbf{w}, H) \text{ holds for all } u \in \mathcal{V}_G \\ \implies & L(\pi(v_i) \mid \mathbf{w}, H) = L(\pi(w_i) \mid \mathbf{w}, H) \text{ holds for all } i \in [k] \\ \implies & \pi(\mathbf{v}) = \mathbf{w}. \end{aligned}$$
2. Suppose π is an isomorphism and $\pi(\mathbf{v}) = \mathbf{w}$, then for any $u \in \mathcal{V}_G$, if u is not in $\mathbf{v}$, obviously $\pi(u)$ is also not in $\mathbf{w}$ (because $\pi(\mathbf{v}) = \mathbf{w}$)). Therefore, both u and $\pi(u)$ are assigned with label 0. If u is in $\mathbf{v}$, without loss of generality we suppose $u = v_i$. Then $\pi(\mathbf{v}) = \mathbf{w} \Rightarrow \pi(v_i) = w_i$. Note that $L(v_i \mid \mathbf{v}, G) = L(w_i \mid \mathbf{w}, G)$ for any $i \in [k]$, therefore $\forall u \in \mathcal{V}_G, L(u \mid \mathbf{v}, G) = L(\pi(u) \mid \mathbf{w}, H)$.

E.2 0-1 induced node labeling

There are also some node labeling method that can be induced by the 0-1 node labeling with message passing. In this section we take Distance Encoding (DE) [4] and Double Radius Node Labeling (DRNL) [7] as examples and show how they are induced by 0-1 node labeling with message passing.

A generalized version We introduce a k-tuple version of DRNL and DE. It's labeling function is

$$L(v \mid \mathbf{u}, G) = \text{Hash}(d(v, u_1), d(v, u_2), ..., d(v, u_k)),$$

where $d(x, y)$ is the shortest distance between x and y. Clearly, DRNL can be computed by MPNNs with 0-1 node labeling. We only need to set $\boldsymbol{h}_v^{(0)} = [\text{Ind}(v, u_1), ..., \text{Ind}(v, u_k)]$ where $\text{Ind}(v, u_i) = 0$ if $v = u_i$ and ∞ otherwise. Therefore it a valid 0-1 node labeling. We let

$$\boldsymbol{h}_v^{(l+1)} = \min\left(\boldsymbol{h}_v^{(l)}, \min\left\{\boldsymbol{h}_u^{(l)} + \mathbf{1} \mid u \in \mathcal{N}(v)\right\}\right),$$

where $\min$ is element-wise. This corresponds to the multi-source shortest path and directly corresponds to the above labeling function L.

F Additional experiments

F.1 Verifying the expressive power of i-j-NL_{MP}

We consider popular variants including 1-1-NL_{MP}, 2-1-NL_{MP} and 0-2-NL_{MP}. To eliminate unrelated factors we assume these models have sufficient injective layers. We test whether they are able to distinguish:

- Cut edges
- Cut vertices.
- Rook's 4×4 graph and Shrikhande graph

These tasks are with increasing difficulties. The problem of detecting cut vertices is shown to be more complex than detecting cut edges [6]. Note that the problem of detecting cut edges in this paper is more difficult than [6]: they assume that target nodes must be adjacent while we do not. Distinguishing Rook's 4×4 graph and Shrikhande graph is even more complex: it cannot be done by the 2-FWL test. Table 1 lists the results. We can see that the results are consistent with our theoretical findings.

Graph type	MPNN	1-1-$\mathrm{NL}_{\mathrm{MP}}$	0-2-$\mathrm{NL}_{\mathrm{MP}}$	2-1-$\mathrm{NL}_{\mathrm{MP}}$
Cut edges	$\times$	✓	✓	✓
Cut vertices	$\times$	$\times$	✓	✓
Rook's vs. Shrikhande	$\times$	$\times$	$\times$	✓

Table 1: Results of detecting different graph patterns.

F.2 Experiment configurations

The experiment configurations follow the settings in Chamberlain et al. [2]. Results are taken from [2]. For all tasks we use negative sampling to generate negative targets. At training time the message passing links are equal to the supervision links, while at test time disjoint sets of links are held out that are never seen at training time. We random generate 70-10-20 percent train-val-test splits which is the same as [2]. The number of the max hops of the paths is 5. We search for hyperparameters using the valid dataset and set learning rate to be 0.0001, dropout 0.5, feature propagation layer 2, weight decay 0. The code is implemented in PyTorch [5] and based on the implementation of Chamberlain et al. [2]. The experiments were either run on V100 or CPU. The predictor of the model is designed as $p(u, v) = \mathrm{MLP}(\boldsymbol{h}_u \odot \boldsymbol{h}_v \odot \boldsymbol{h}_{uv})$ where $\boldsymbol{h}_u, \boldsymbol{h}_v$ are node representations of u, v respectively and $\boldsymbol{h}_{uv}$ is the representations of the paths between u, v.

G The problems related to higher-order representations

The problems of learning higher-order representations with GNNs have been extensively studied. However, in fact these methods should be divided into two parts: *learning higher-order representations for predicting node-tuple properties* and *designing higher-order GNNs for learning graph representations*. Although they both learn higher-order representations, they actually study different problems which requires different techniques and solutions.

The goal of the first approaches is to overcome GNNs' inherent weakness on predicting node tuples. This includes the well-known automorphism problem, such as Figure 1. u_1, u_k, v_1, v_k always share the same node representation, so it is impossible to distinguish between them using a vanilla GNN. Therefore, we need to design more powerful GNN variants to give GNNs new abilities to consider the correlation between u_1, u_k and v_1, v_k.

The goal of the second approaches is to improve GNNs' expressive power on graph classification and node classification. Therefore, learning higher-order representations is nothing but a method for strengthening GNNs. In other words, if there are better methods for learning node-level or graph-level representations, higher-order GNNs are not required for this goal.

The difference between the two goals is that, suppose we have a powerful GNN that is able to distinguishes any non-isomorphic nodes (or graphs). Then, the second goal is accomplished: we already reached the upper bound of the expressive power. However, the first goal is far from being accomplished: the automorphism problem still remains. As a result, although higher-order GNNs play an important role in both two problems, the ultimate target and solutions are much different.

References

[1] J.-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12:389–410, 1989.

[2] B. P. Chamberlain, S. Shirobokov, E. Rossi, F. Frasca, T. Markovich, N. Y. Hammerla, M. Bronstein, and M. Hansmire. Graph neural networks for link prediction with subgraph sketching. In *ICLR*, 2023.

[3] M. Grohe and M. Otto. Pebble games and linear equations. *The Journal of Symbolic Logic*, 80: 797 – 844, 2012.

[4] P. Li, Y. Wang, H. Wang, and J. Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *NeurIPS*, 2020.

[5] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703, 2019.

[6] B. Zhang, S. Luo, L. Wang, and D. He. Rethinking the expressive power of gnns via graph biconnectivity. *ArXiv*, abs/2301.09505, 2023.

[7] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *NeurIPS*, 2018.