Cluster-wise Graph Transformer with Dual-granularity Kernelized Attention

Siyuan Huang · Yunchong Song · Jiayue Zhou · Zhouhan Lin

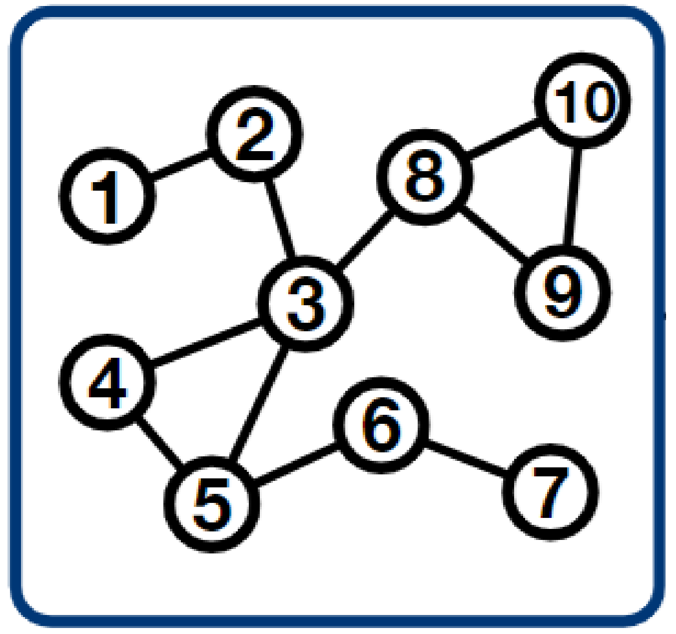
上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

NEURAL INFORMATION PROCESSING SYSTEMS

How to pool a graph step by step?

How to pool a graph step by step?

An illustrative example of **Node Clustering Pooling**



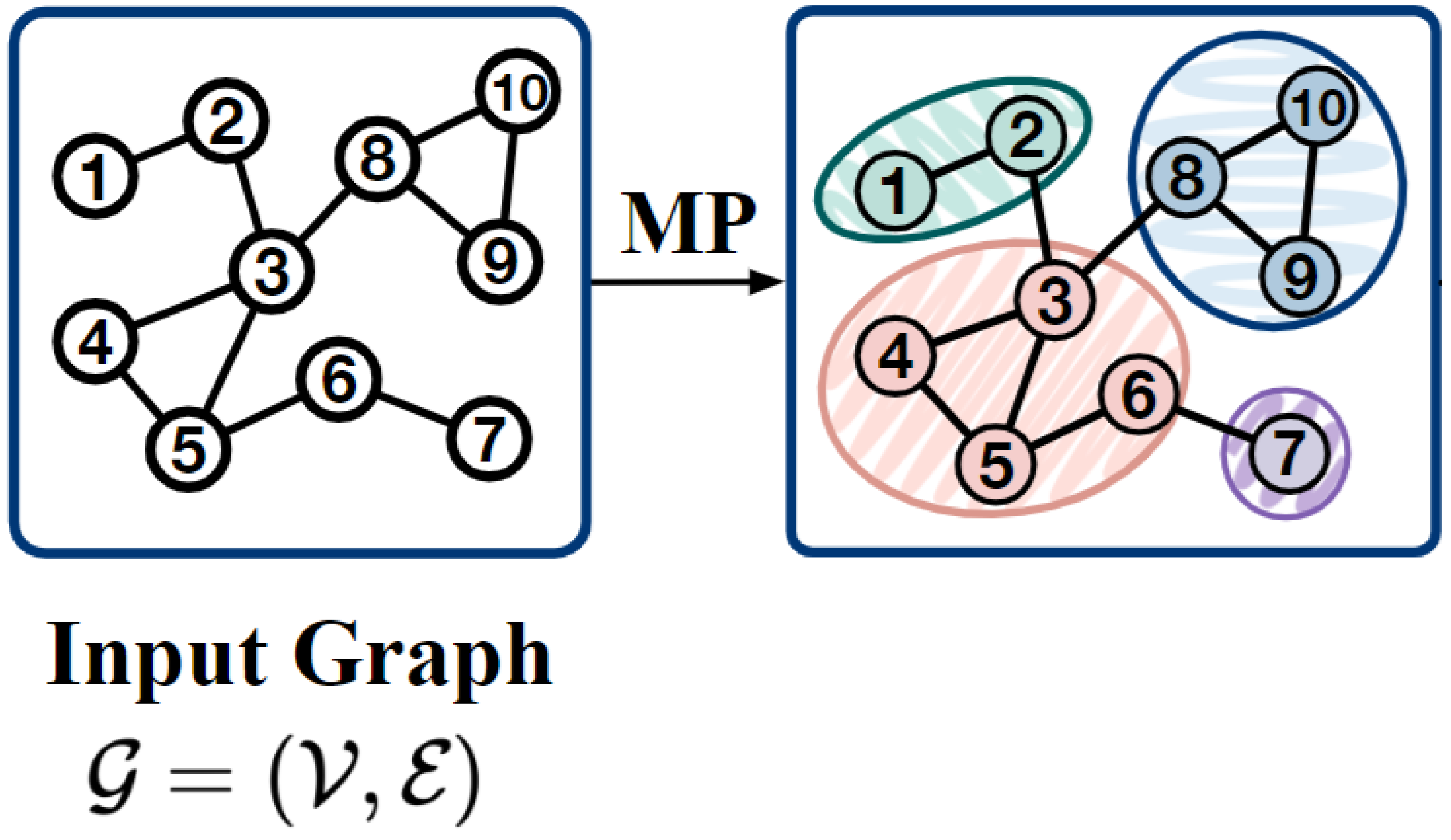
Input Graph
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., & Tao, D. (2023). Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. *arXiv preprint arXiv:2204.07321*.

How to pool a graph step by step?

An illustrative example of **Node Clustering Pooling**

MP: Message Propagation

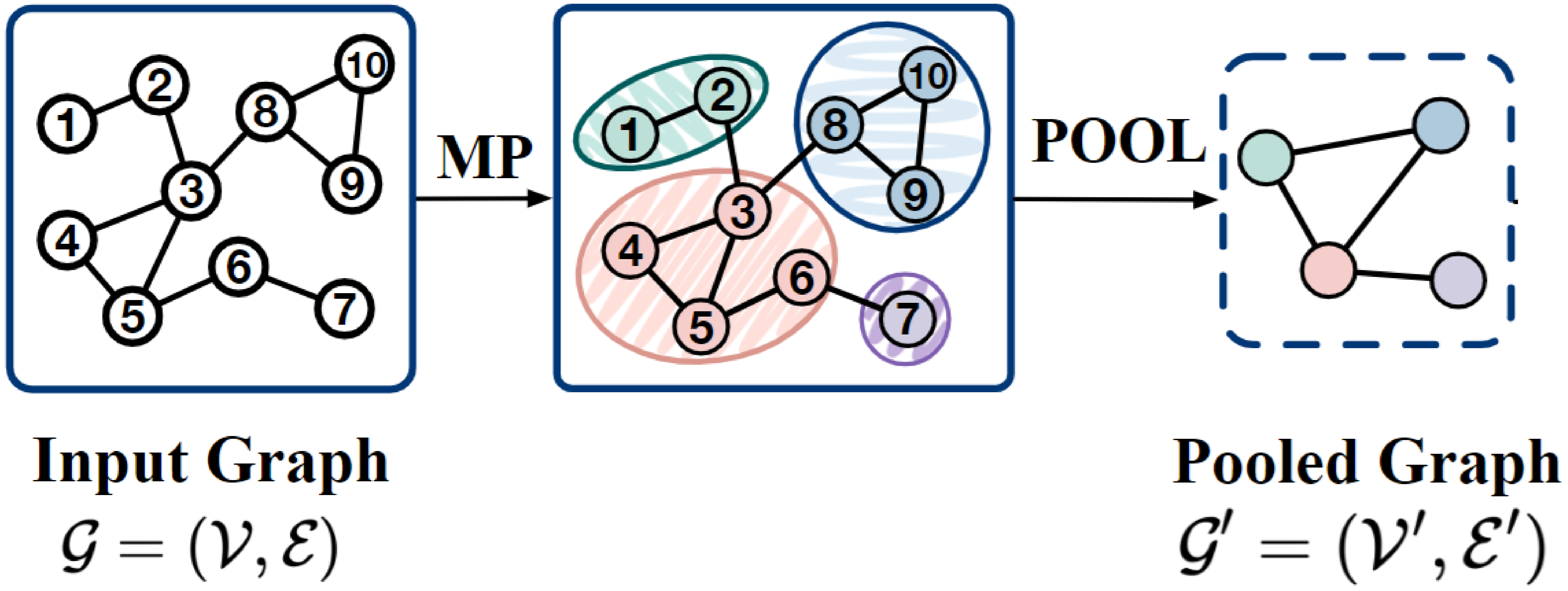


Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., & Tao, D. (2023). Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. *arXiv preprint arXiv:2204.07321*.

How to pool a graph step by step?

An illustrative example of **Node Clustering Pooling**

MP: Message Propagation

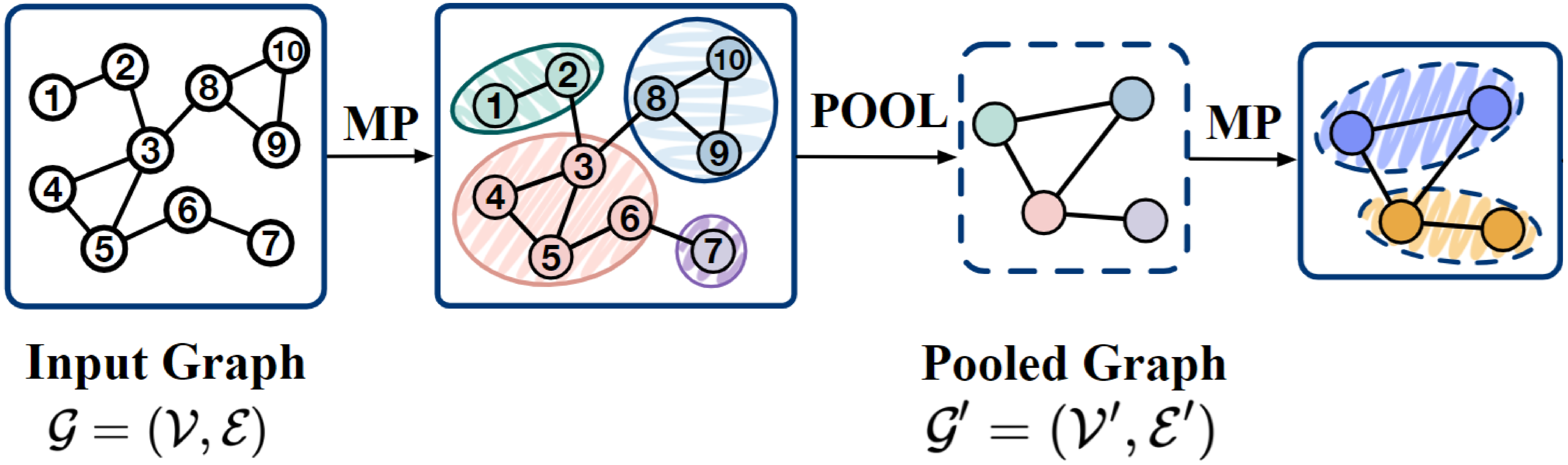


Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., & Tao, D. (2023). Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. *arXiv preprint arXiv:2204.07321*.

How to pool a graph step by step?

An illustrative example of **Node Clustering Pooling**

MP: Message Propagation

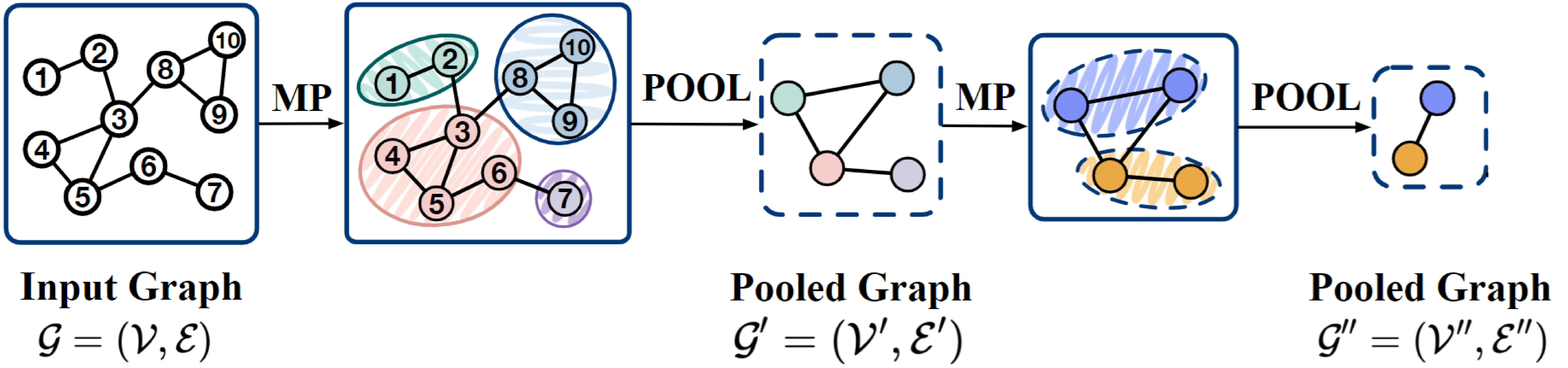


Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., & Tao, D. (2023). Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. *arXiv preprint arXiv:2204.07321*.

How to pool a graph step by step?

An illustrative example of **Node Clustering Pooling**

MP: Message Propagation

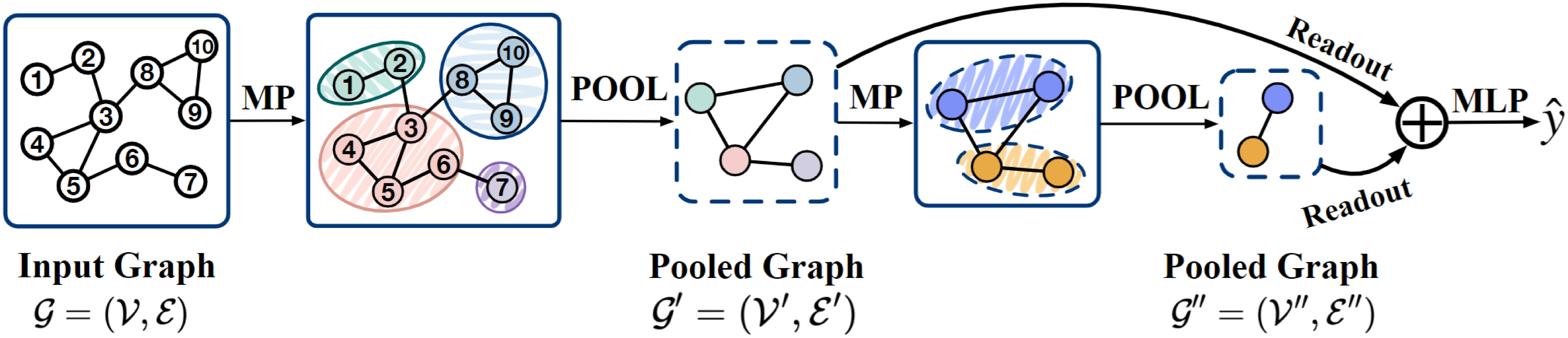


Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., & Tao, D. (2023). Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. *arXiv preprint arXiv:2204.07321*.

How to pool a graph step by step?

An illustrative example of **Node Clustering Pooling**

MP: Message Propagation

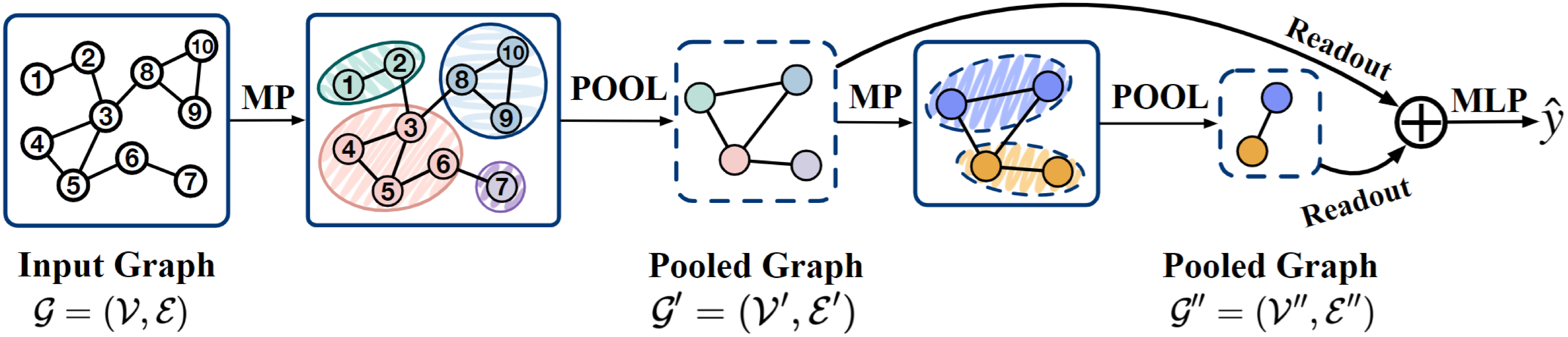


Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., & Tao, D. (2023). Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. *arXiv preprint arXiv:2204.07321*.

How to pool a graph step by step?

An illustrative example of **Node Clustering Pooling**

MP: Message Propagation



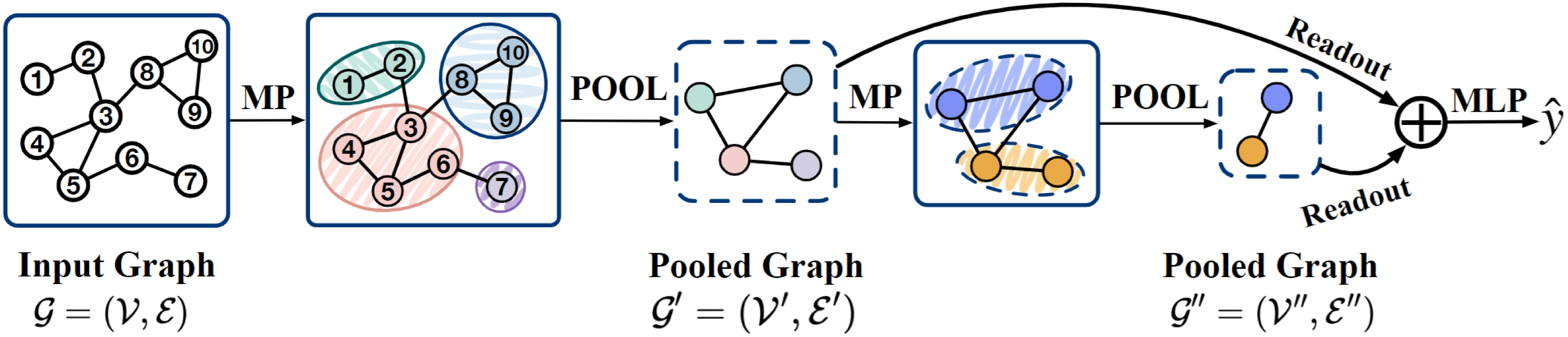
Cluster Assignment + Graph Coarsening

Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., & Tao, D. (2023). Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. *arXiv preprint arXiv:2204.07321*.

How to pool a graph step by step?

An illustrative example of **Node Clustering Pooling**

MP: Message Propagation



Cluster Assignment + Graph Coarsening

compressing each cluster into a single embedding

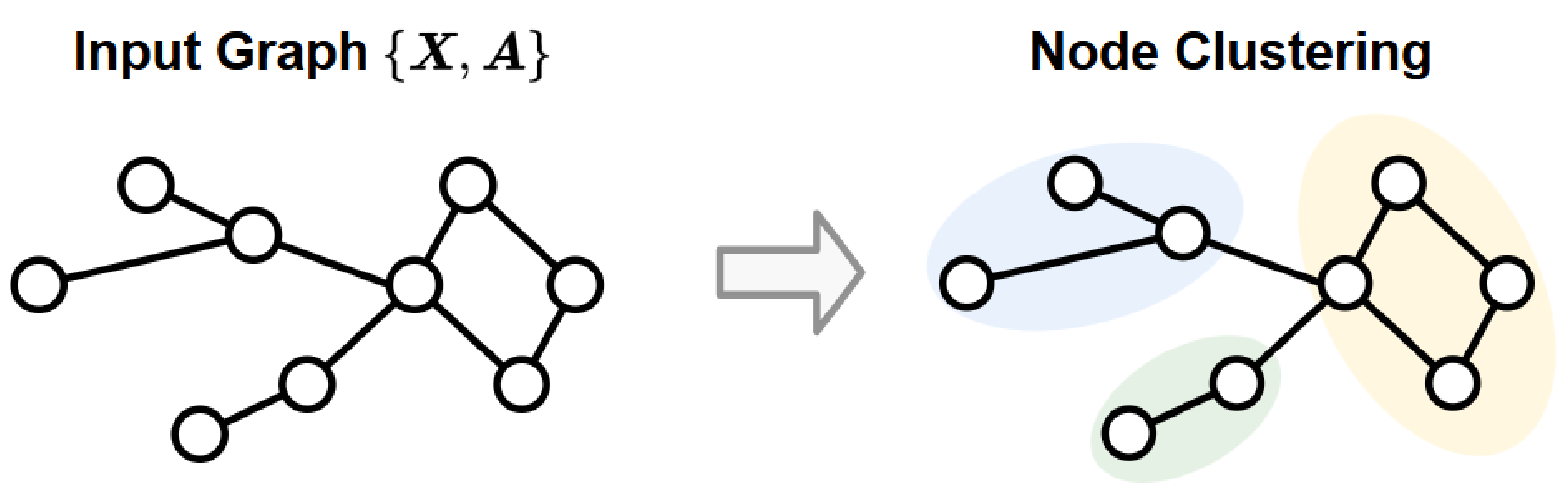
Liu, C., Zhan, Y., Wu, J., Li, C., Du, B., Hu, W., Liu, T., & Tao, D. (2023). Graph Pooling for Graph Neural Networks: Progress, Challenges, and Opportunities. *arXiv preprint arXiv:2204.07321*.

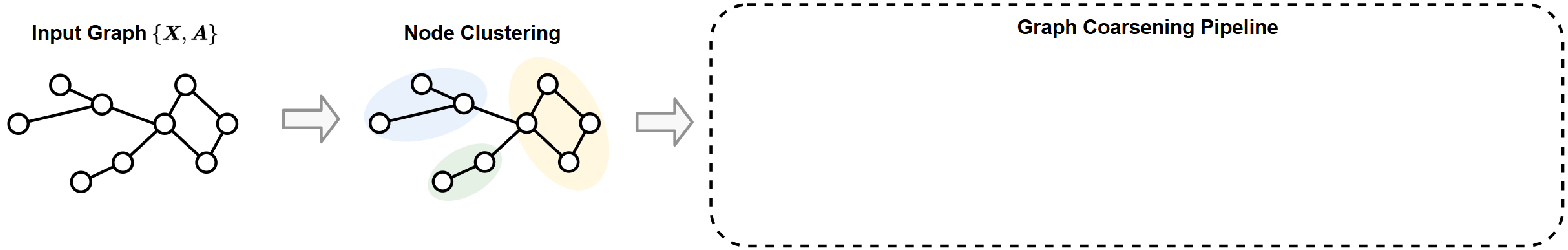
Graph Coarsening Pipeline leads to some problems when performing MP:

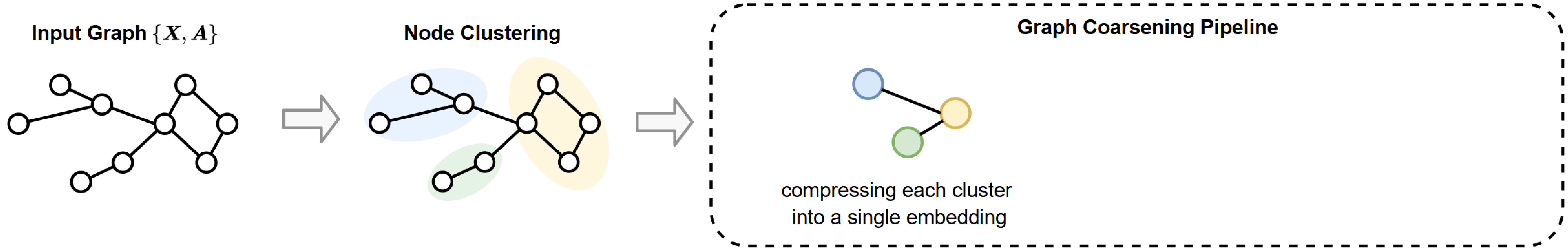
- overly homogeneous cluster representations
- loss of node-level information

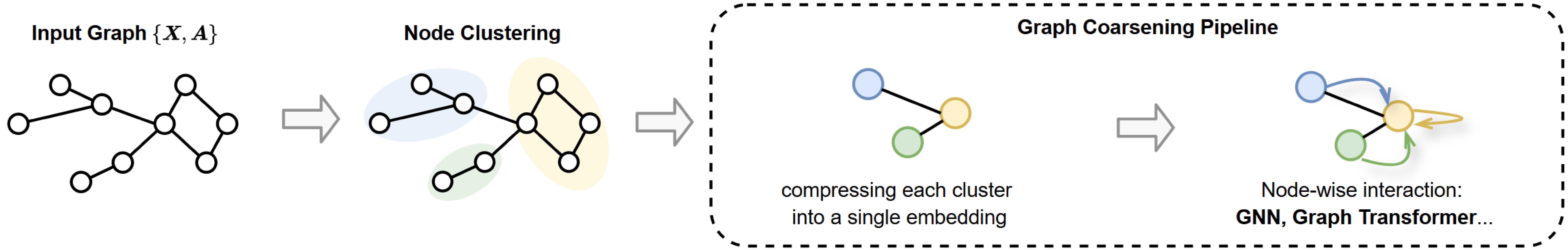
➢ Can we avoid reducing each cluster to a single node?

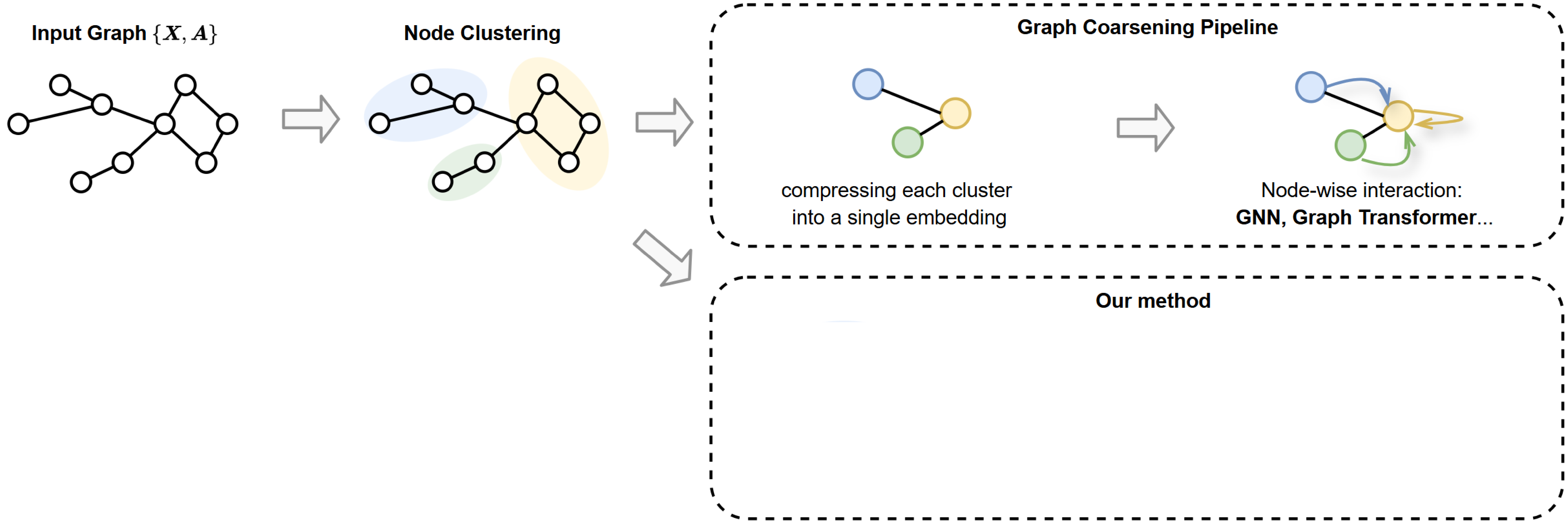
➢ We can envision the graph as a network of interconnected node sets.


Input Graph $\{X, A\}$
Node Clustering

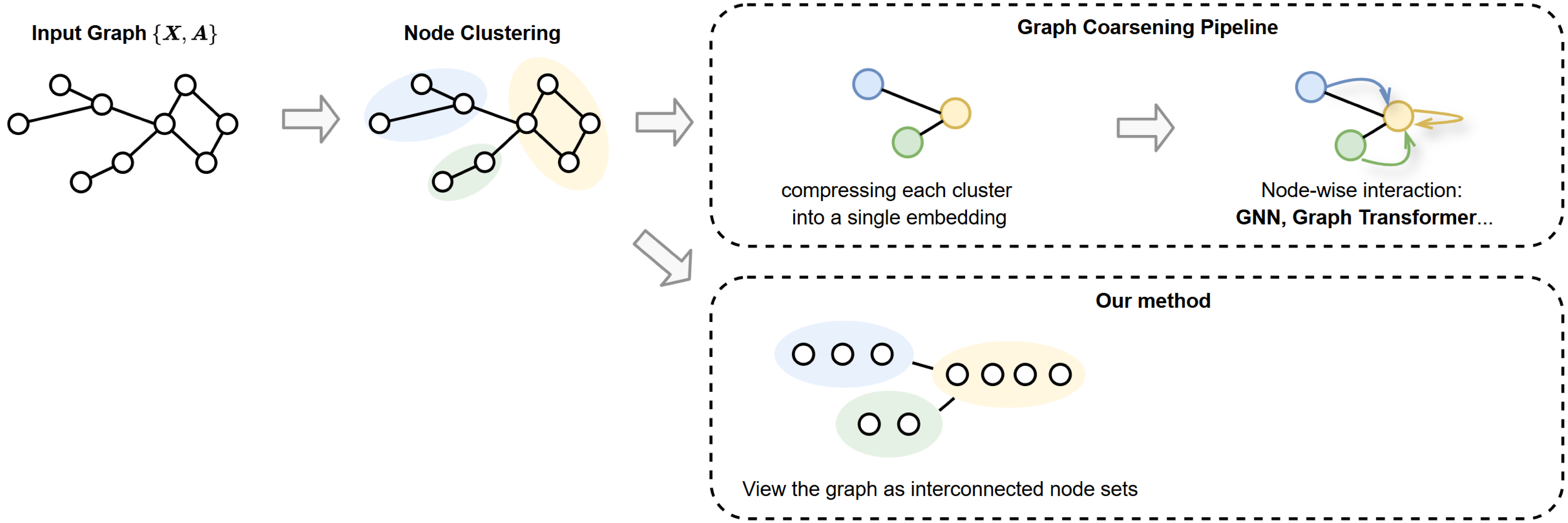

Input Graph $\{\boldsymbol{X}, \boldsymbol{A}\}$
Node Clustering
Graph Coarsening Pipeline


Input Graph $\{X, A\}$
Node Clustering
Graph Coarsening Pipeline
compressing each cluster
into a single embedding

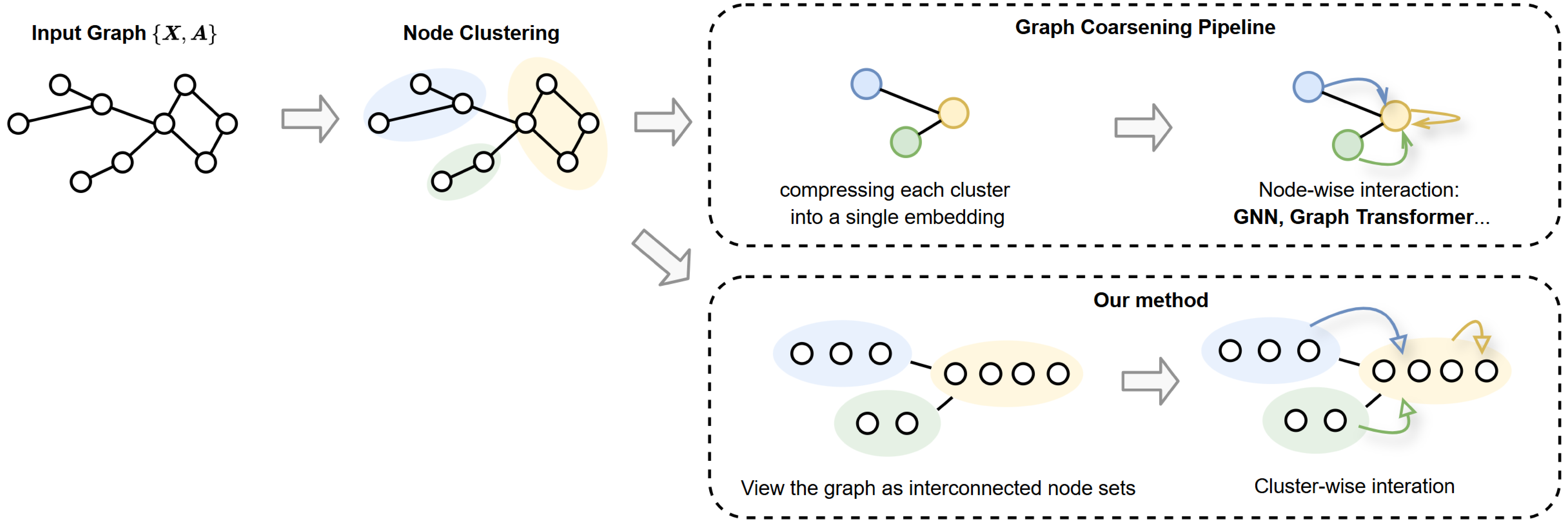

Input Graph $\{X, A\}$
Node Clustering
Graph Coarsening Pipeline
compressing each cluster
into a single embedding
Node-wise interaction:
GNN, Graph Transformer...



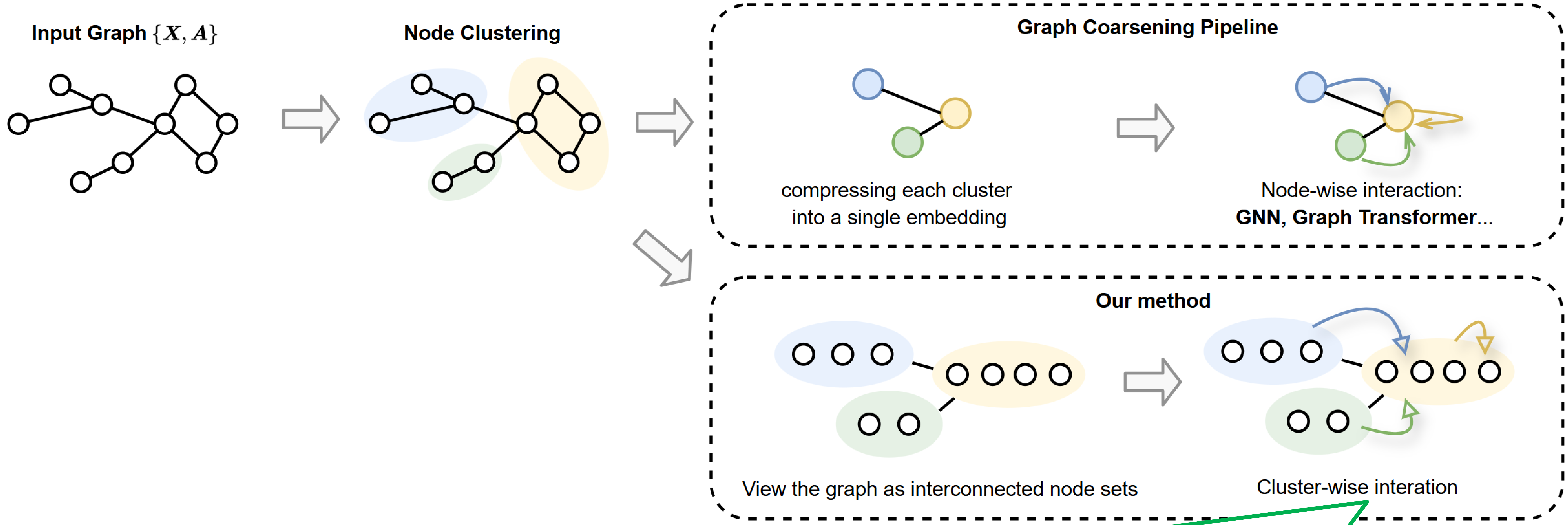
Input Graph $\{X, A\}$
Node Clustering
Graph Coarsening Pipeline
compressing each cluster
into a single embedding
Node-wise interaction:
GNN, Graph Transformer...
Our method



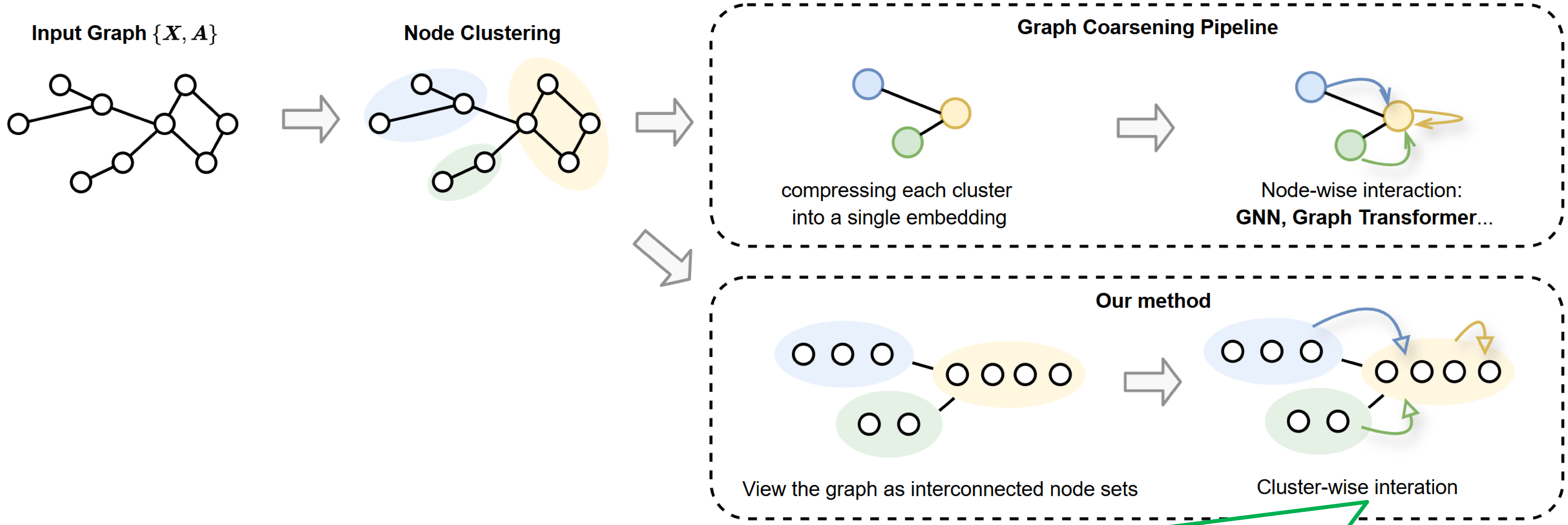
Input Graph $\{X, A\}$
Node Clustering
Graph Coarsening Pipeline
compressing each cluster
into a single embedding
Node-wise interaction:
GNN, Graph Transformer...
Our method
View the graph as interconnected node sets



Input Graph $\{X, A\}$
Node Clustering
Graph Coarsening Pipeline
compressing each cluster
into a single embedding
Node-wise interaction:
GNN, Graph Transformer...
Our method
View the graph as interconnected node sets
Cluster-wise interation



We need a method that captures information at both the node and cluster levels.



We need a method that captures information at both the node and cluster levels.

Node-to-Cluster Attention Mechanism

How to design the Node-to-Cluster Attention Mechanism?

How to design the Node-to-Cluster Attention Mechanism?

After node clustering, each node possesses two tiers of information:

a) individual node feature

b) collective feature of its cluster

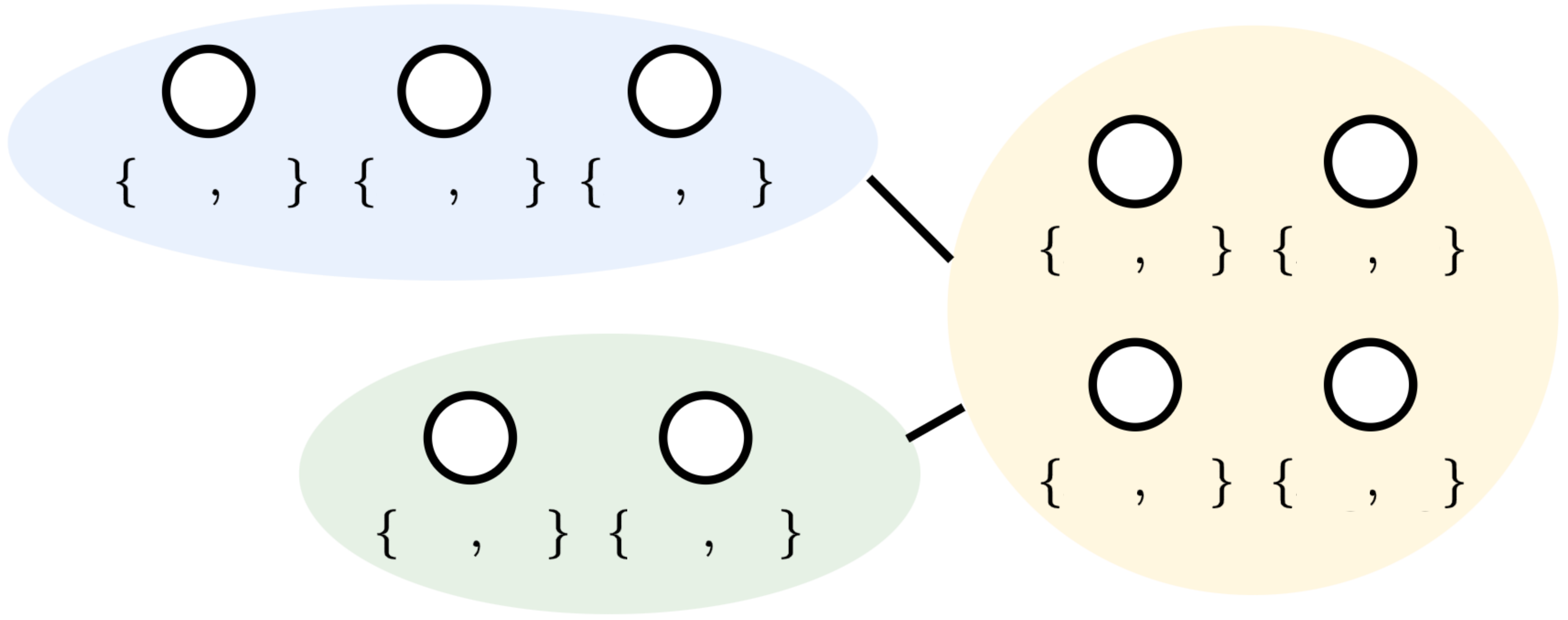
How to design the Node-to-Cluster Attention Mechanism?

After node clustering, each node possesses two tiers of information:

a) individual node feature

b) collective feature of its cluster

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters		
Keys of nodes		



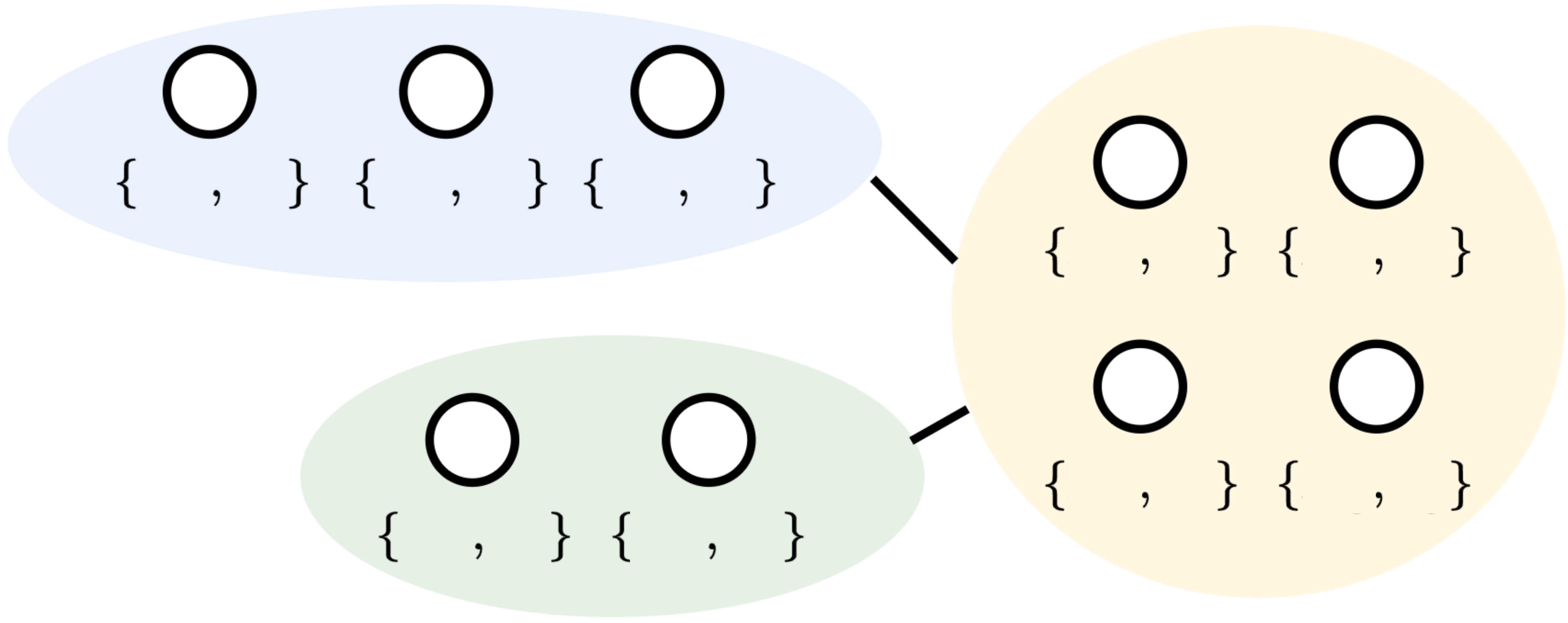
How to design the Node-to-Cluster Attention Mechanism?

After node clustering, each node possesses two tiers of information:

a) individual node feature, denoted by node-level key: $k_t \in \mathcal{X}_N$

b) collective feature of its cluster

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters		
Keys of nodes		



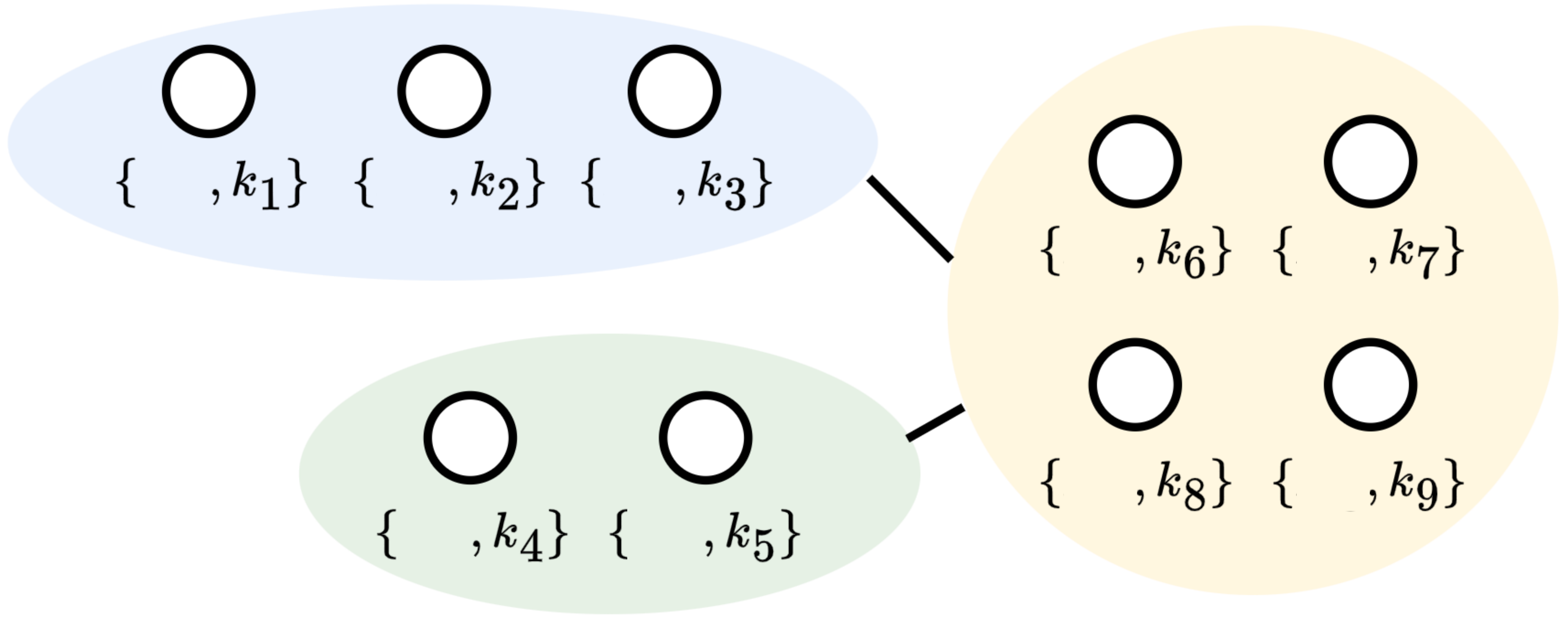
How to design the Node-to-Cluster Attention Mechanism?

After node clustering, each node possesses two tiers of information:

a) individual node feature, denoted by node-level key: $k_t \in \mathcal{X}_N$

b) collective feature of its cluster

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters		
Keys of nodes		$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$



How to design the Node-to-Cluster Attention Mechanism?

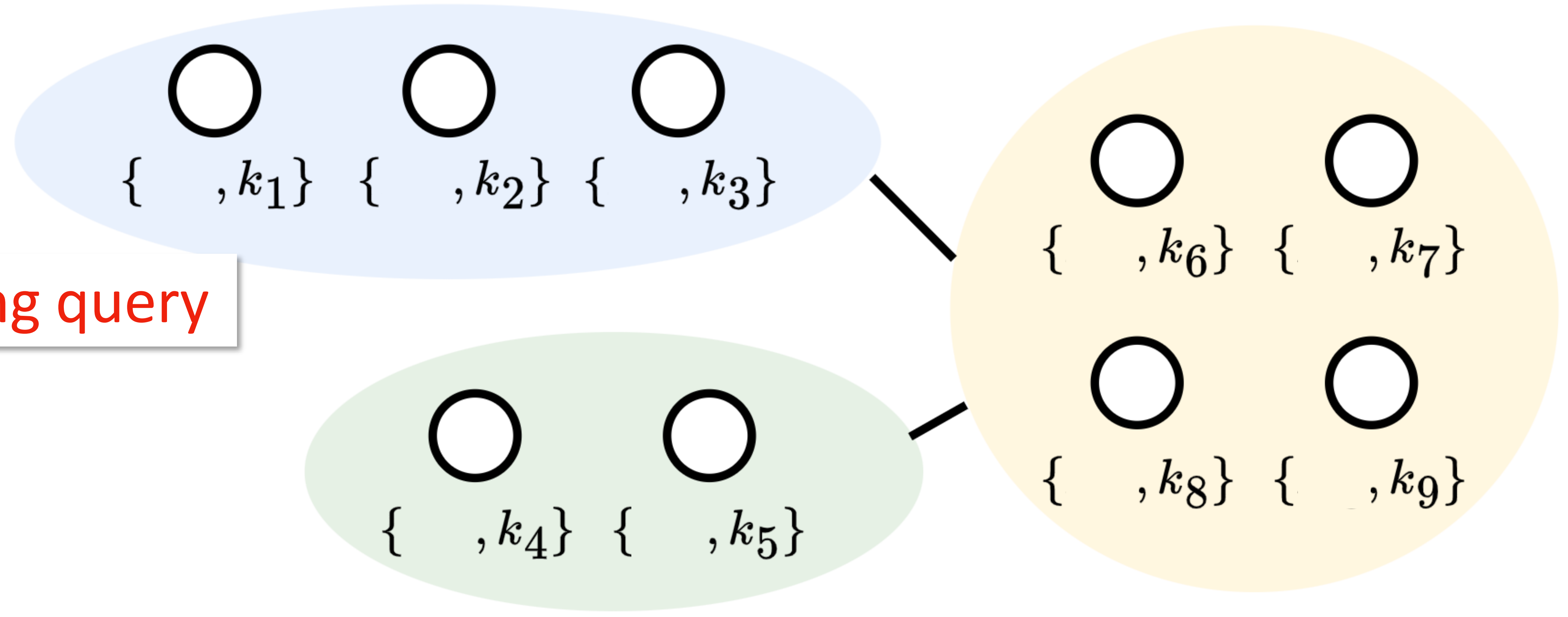
After node clustering, each node possesses two tiers of information:

a) individual node feature, denoted by node-level key: $k_t \in \mathcal{X}_N$

b) collective feature of its cluster

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters		q_1, q_2, q_3
Keys of nodes		$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$

corresponding query



How to design the Node-to-Cluster Attention Mechanism?

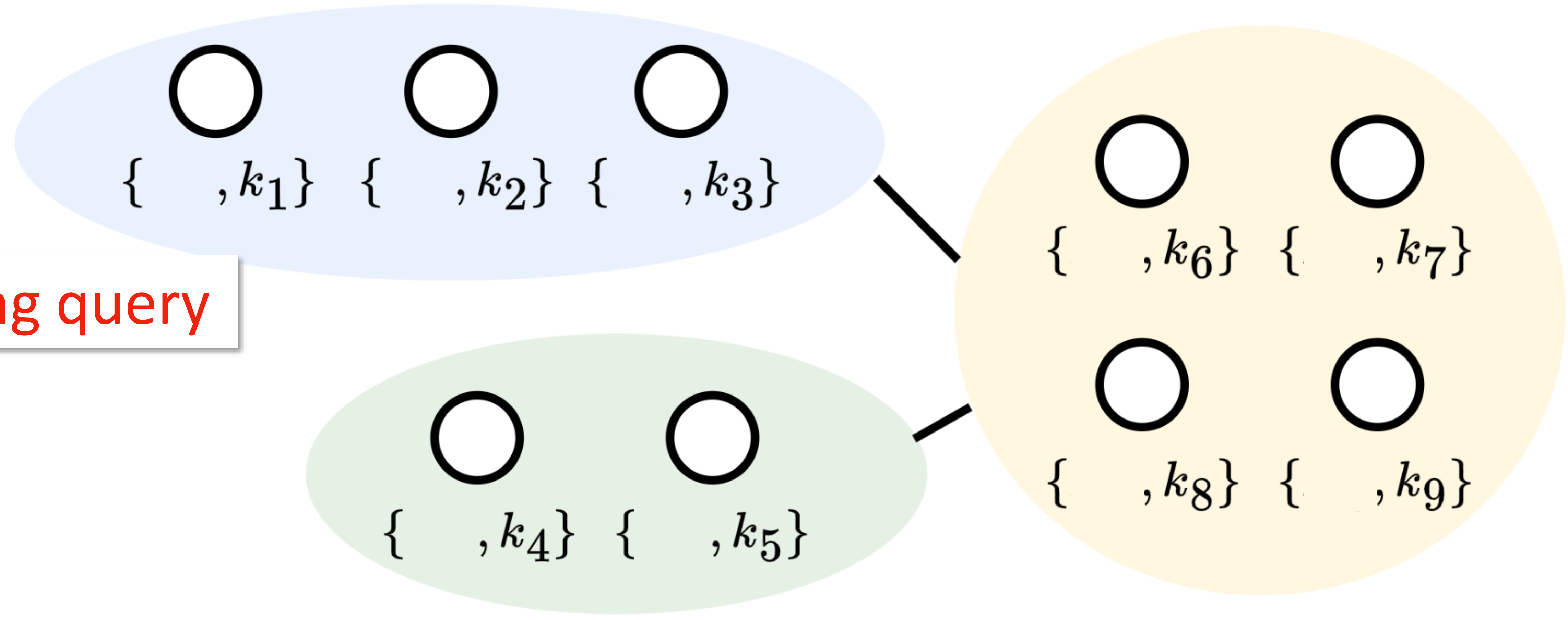
After node clustering, each node possesses two tiers of information:

a) individual node feature, denoted by node-level key: $k_t \in \mathcal{X}_N$

b) collective feature of its cluster, denoted by cluster-level key: $K_j \in \mathcal{X}_C$

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters		q_1, q_2, q_3
Keys of nodes		$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$

corresponding query



How to design the Node-to-Cluster Attention Mechanism?

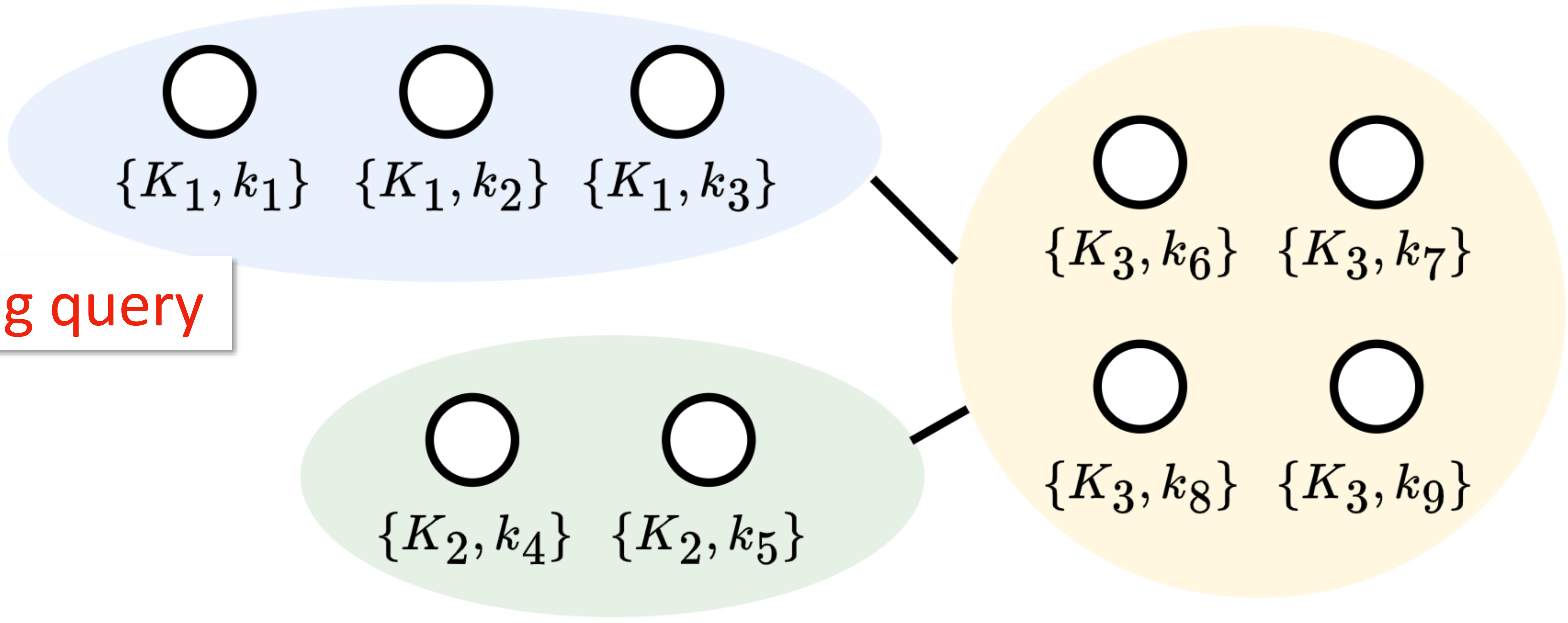
After node clustering, each node possesses two tiers of information:

a) individual node feature, denoted by node-level key: $k_t \in \mathcal{X}_N$

b) collective feature of its cluster, denoted by cluster-level key: $K_j \in \mathcal{X}_C$

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters		q_1, q_2, q_3
Keys of nodes	K_1, K_2, K_3	$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$

corresponding query



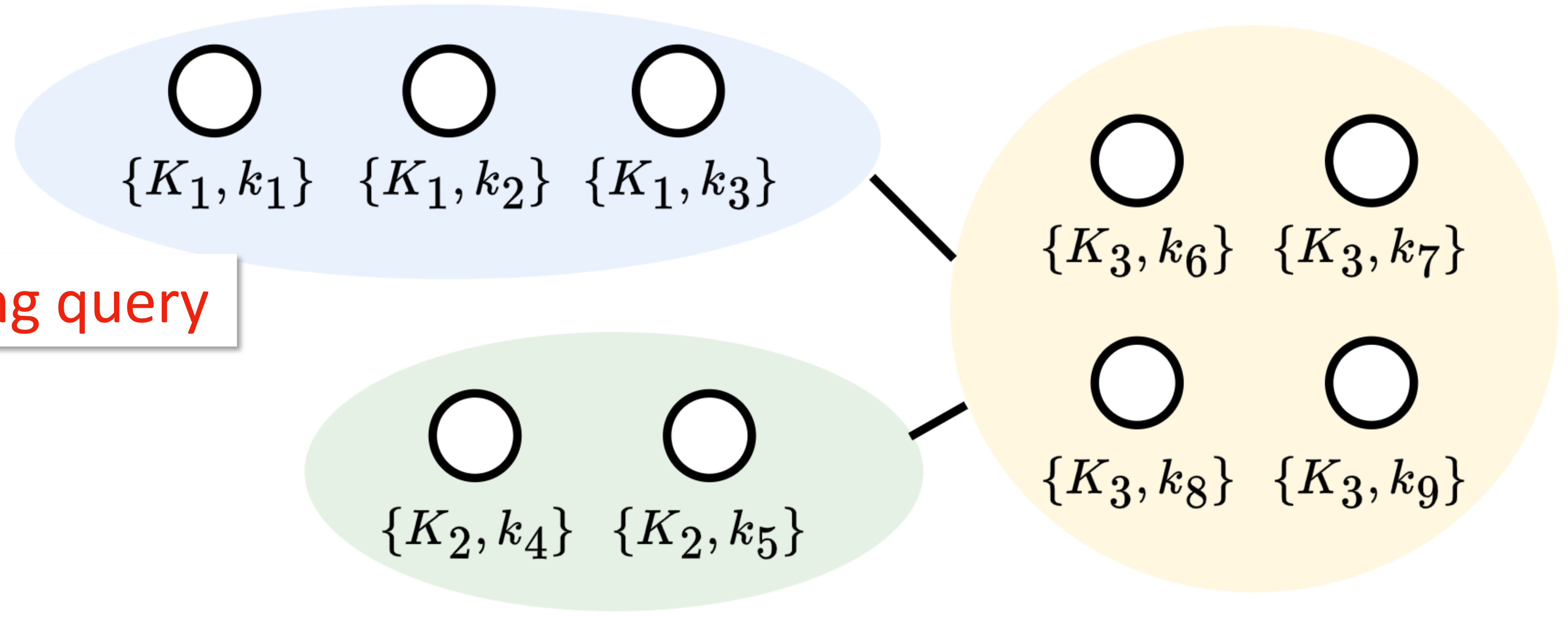
How to design the Node-to-Cluster Attention Mechanism?

After node clustering, each node possesses two tiers of information:

a) individual node feature, denoted by node-level key: $k_t \in \mathcal{X}_N$

b) collective feature of its cluster, denoted by cluster-level key: $K_j \in \mathcal{X}_C$

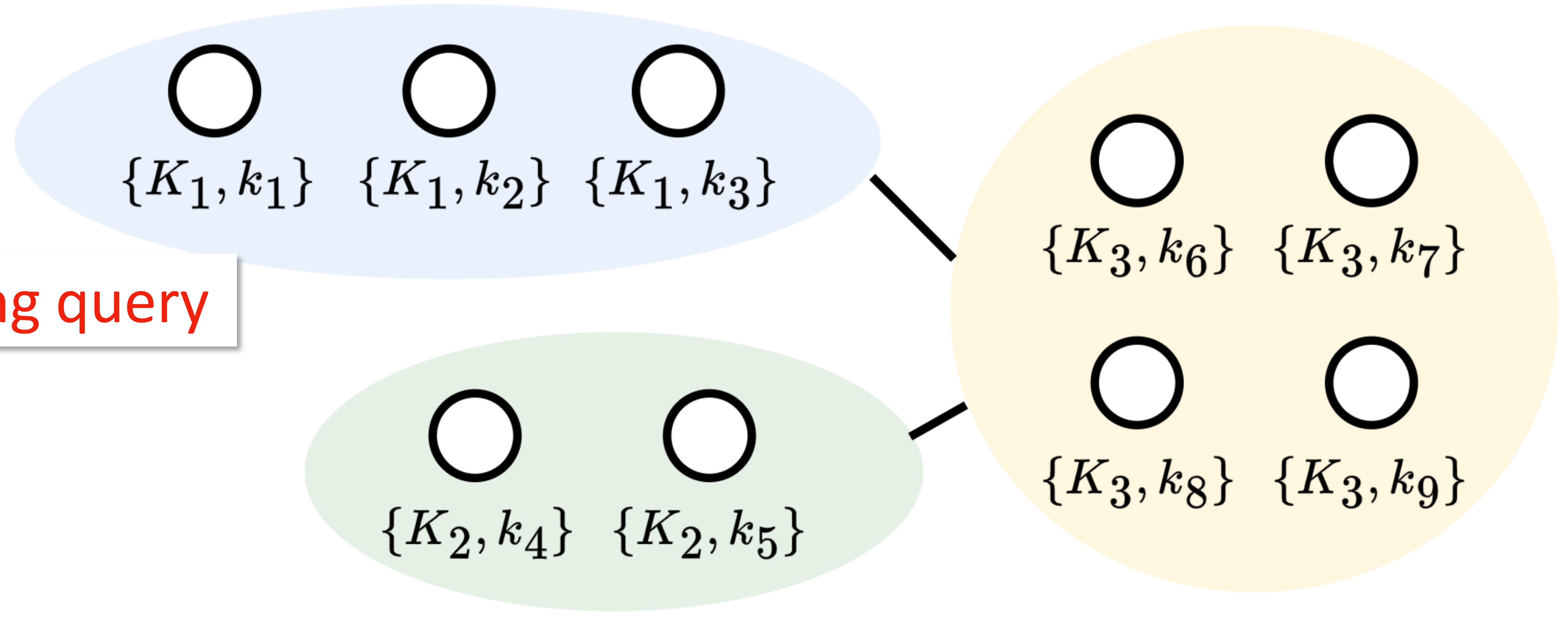
Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters	Q_1, Q_2, Q_3	q_1, q_2, q_3 corresponding query
Keys of nodes	K_1, K_2, K_3	$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$



How to design the Node-to-Cluster Attention Mechanism?

Having obtained the bi-level queries and keys,
we consider how to measure their similarity.

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters	Q_1, Q_2, Q_3	q_1, q_2, q_3 corresponding query
Keys of nodes	K_1, K_2, K_3	$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$

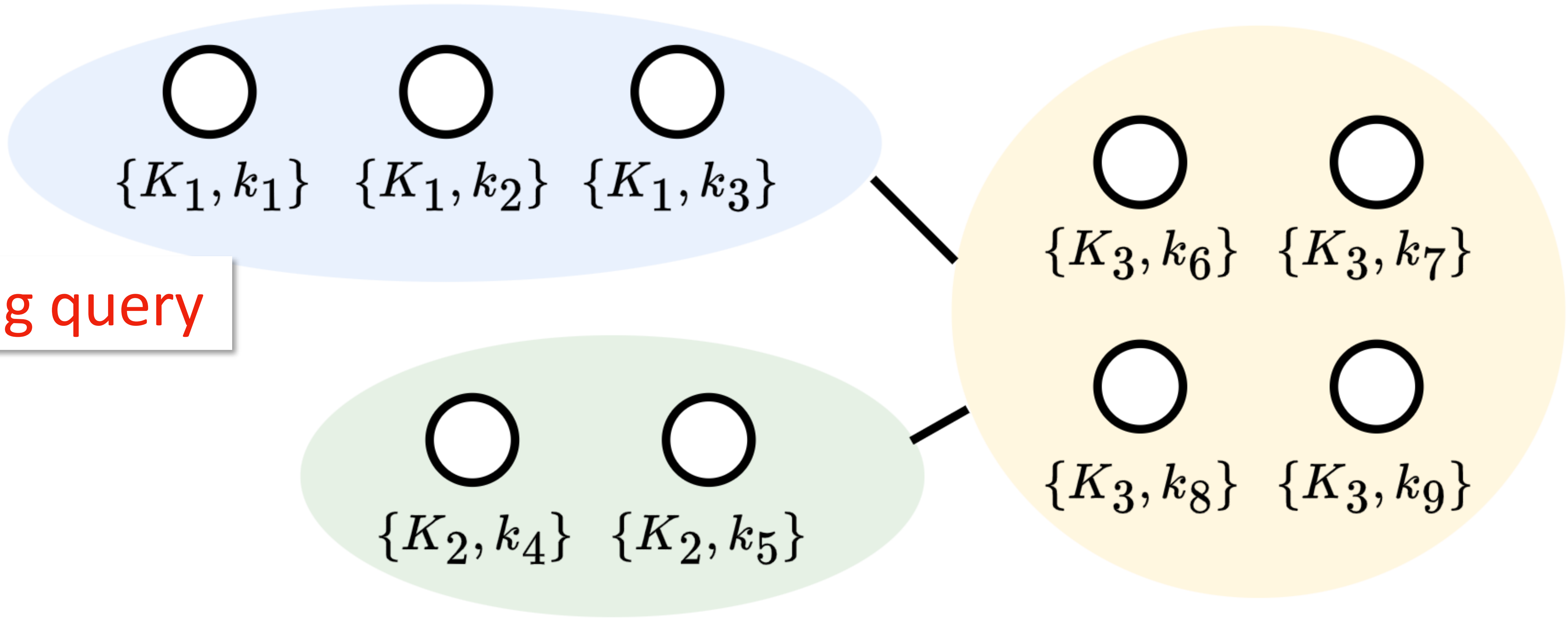


How to design the Node-to-Cluster Attention Mechanism?

Having obtained the bi-level queries and keys,
we consider how to measure their similarity.

κ_C : a valid kernel in the cluster-level space $\mathcal{X}_C$

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters	Q_1, Q_2, Q_3	q_1, q_2, q_3 corresponding query
Keys of nodes	K_1, K_2, K_3	$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$



How to design the Node-to-Cluster Attention Mechanism?

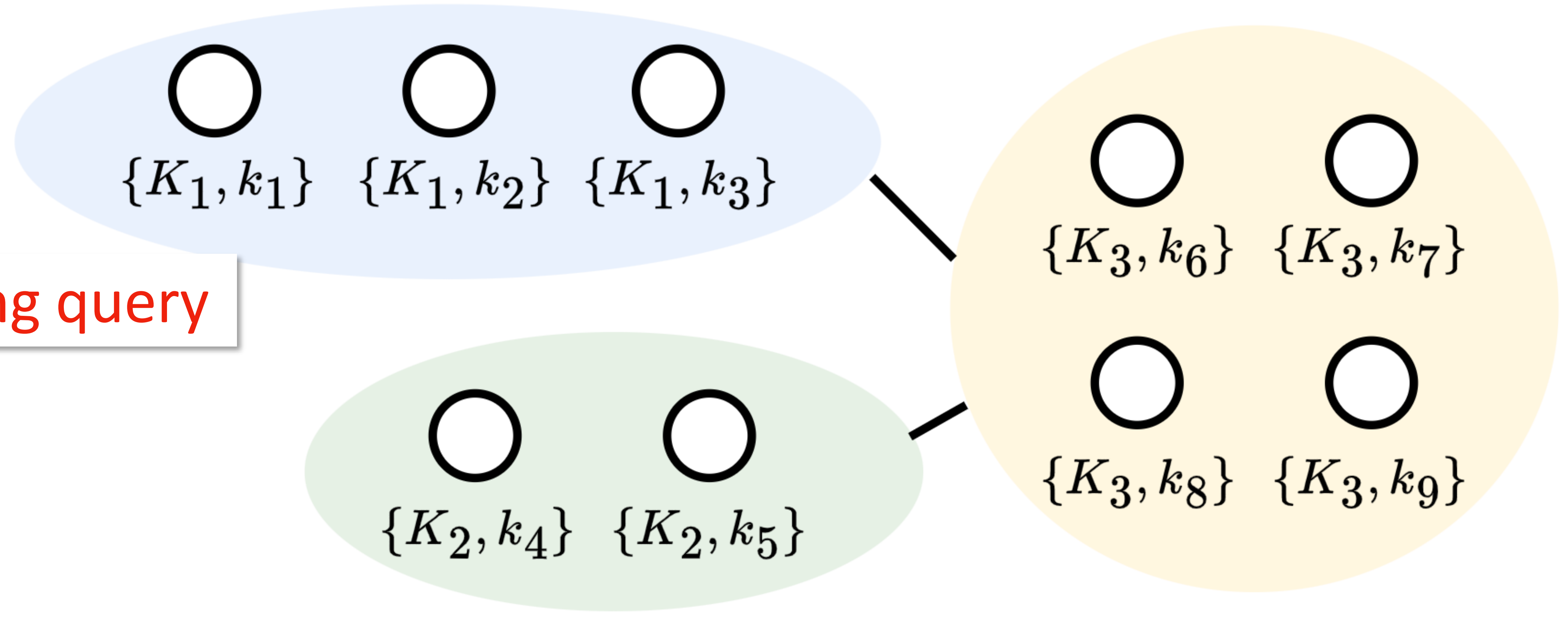
Having obtained the bi-level queries and keys,
we consider how to measure their similarity.

κ_C : a valid kernel in the cluster-level space $\mathcal{X}_C$

κ_N : a valid kernel in the cluster-level space $\mathcal{X}_N$

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters	Q_1, Q_2, Q_3	q_1, q_2, q_3
Keys of nodes	K_1, K_2, K_3	$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$

corresponding query



How to design the Node-to-Cluster Attention Mechanism?

Having obtained the bi-level queries and keys, we consider how to measure their similarity.

κ_C : a valid kernel in the cluster-level space $\mathcal{X}_C$

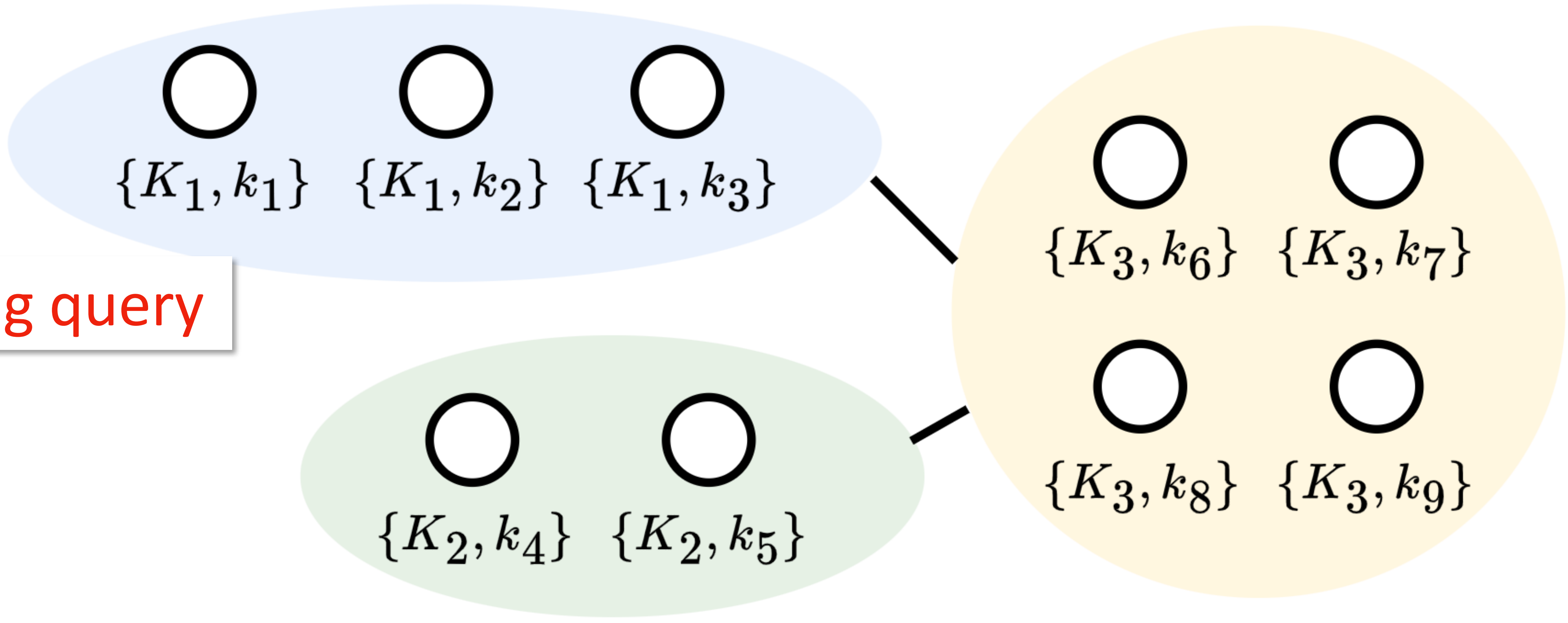
κ_N : a valid kernel in the cluster-level space $\mathcal{X}_N$

Now we have:

$$\{K_j, k_t\} \in \mathcal{X}_C \times \mathcal{X}_N$$
$$\{Q_i, q_i\} \in \mathcal{X}_C \times \mathcal{X}_N$$

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters	Q_1, Q_2, Q_3	q_1, q_2, q_3
Keys of nodes	K_1, K_2, K_3	$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$

corresponding query



How to design the Node-to-Cluster Attention Mechanism?

Having obtained the bi-level queries and keys, we consider how to measure their similarity.

κ_C : a valid kernel in the cluster-level space $\mathcal{X}_C$

κ_N : a valid kernel in the cluster-level space $\mathcal{X}_N$

Now we have:

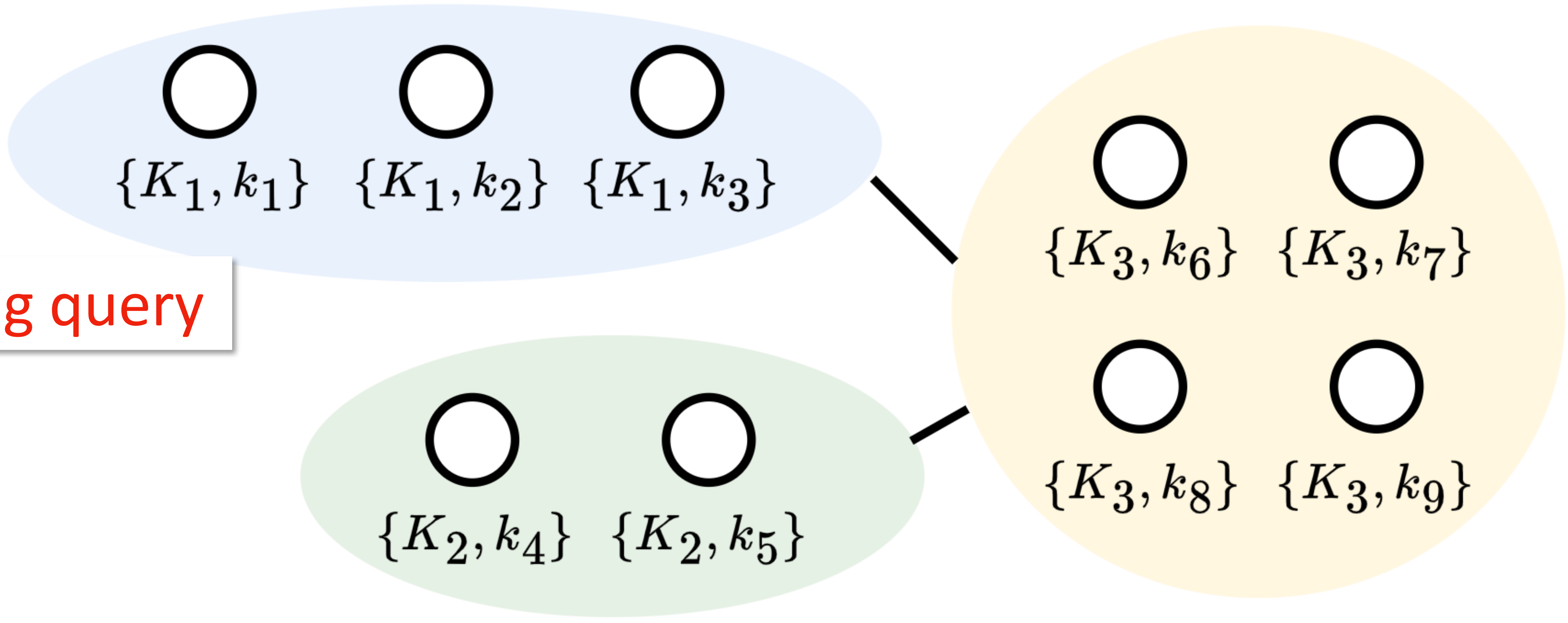
$$\{K_j, k_t\} \in \mathcal{X}_C \times \mathcal{X}_N$$

$$\{Q_i, q_i\} \in \mathcal{X}_C \times \mathcal{X}_N$$

We should construct a kernel on the tensor product space $\mathcal{X}_C \times \mathcal{X}_N$

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters	Q_1, Q_2, Q_3	q_1, q_2, q_3
Keys of nodes	K_1, K_2, K_3	$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$

corresponding query



How to design the Node-to-Cluster Attention Mechanism?

κ_{B} : a valid kernel in the cluster-level space $\mathcal{X}_C \times \mathcal{X}_N$

How to design the Node-to-Cluster Attention Mechanism?

κ_{B} : a valid kernel in the cluster-level space $\mathcal{X}_C \times \mathcal{X}_N$

We mainly consider two options for κ_{B}

How to design the Node-to-Cluster Attention Mechanism?

κ_{B} : a valid kernel in the cluster-level space $\mathcal{X}_C \times \mathcal{X}_N$

We mainly consider two options for κ_{B}

- tensor product of κ_C and κ_N

How to design the Node-to-Cluster Attention Mechanism?

κ_{B} : a valid kernel in the cluster-level space $\mathcal{X}_C \times \mathcal{X}_N$

We mainly consider two options for κ_{B}

- tensor product of κ_C and κ_N
- Convex linear combination of κ_C and κ_N

How to design the Node-to-Cluster Attention Mechanism?

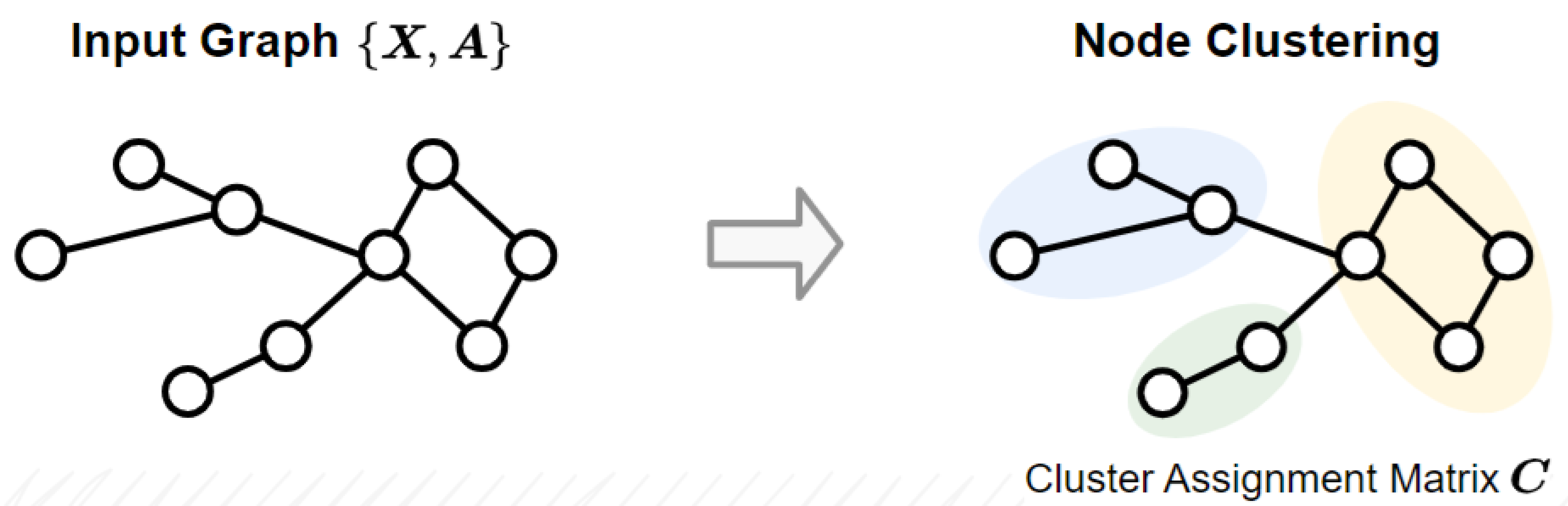
κ_{B} : a valid kernel in the cluster-level space $\mathcal{X}_C \times \mathcal{X}_N$

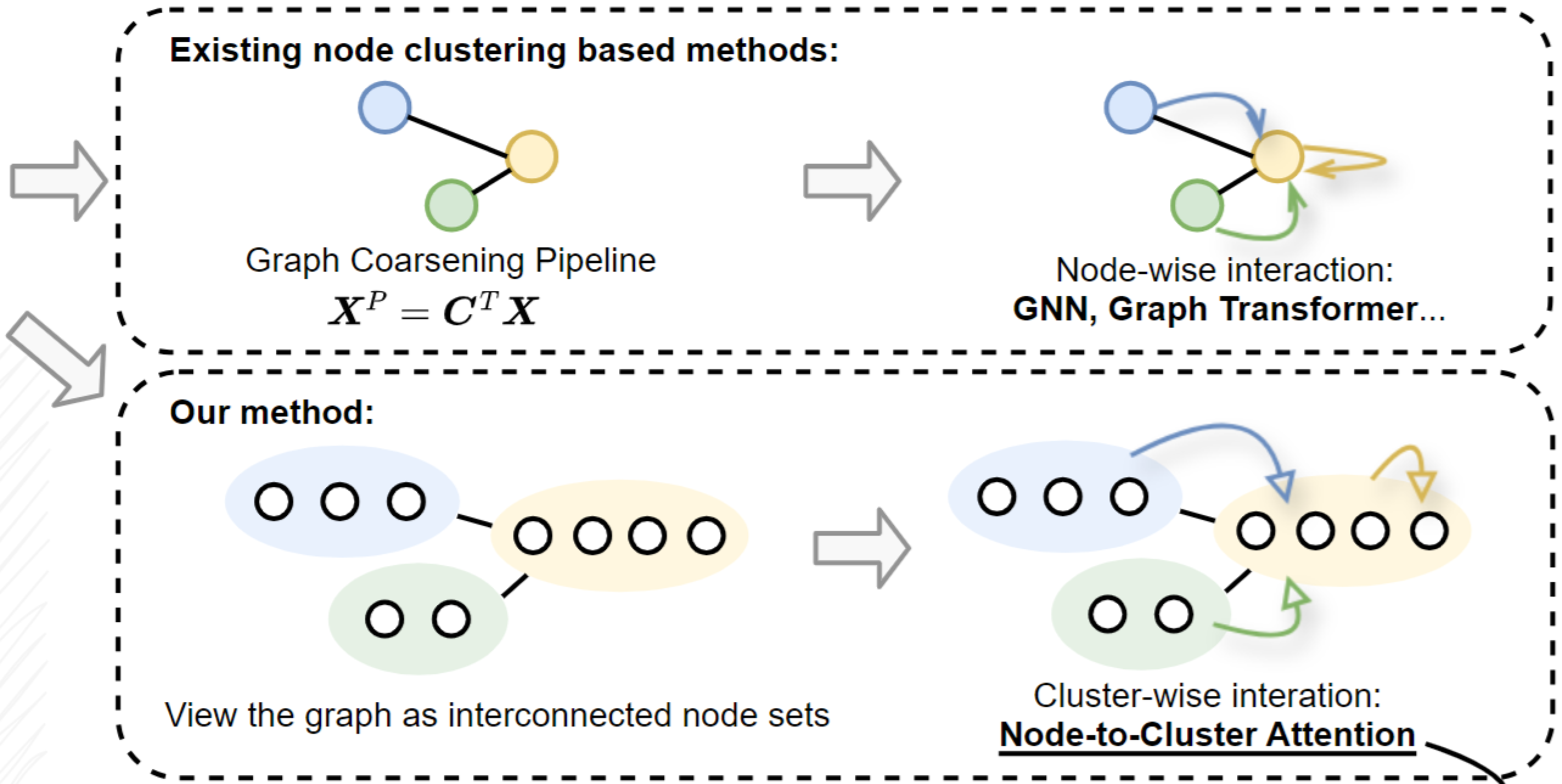
We mainly consider two options for κ_{B}

- tensor product of κ_C and κ_N
- Convex linear combination of κ_C and κ_N

Refer to the paper for more details.

Our proposed Node-to-Cluster Attention





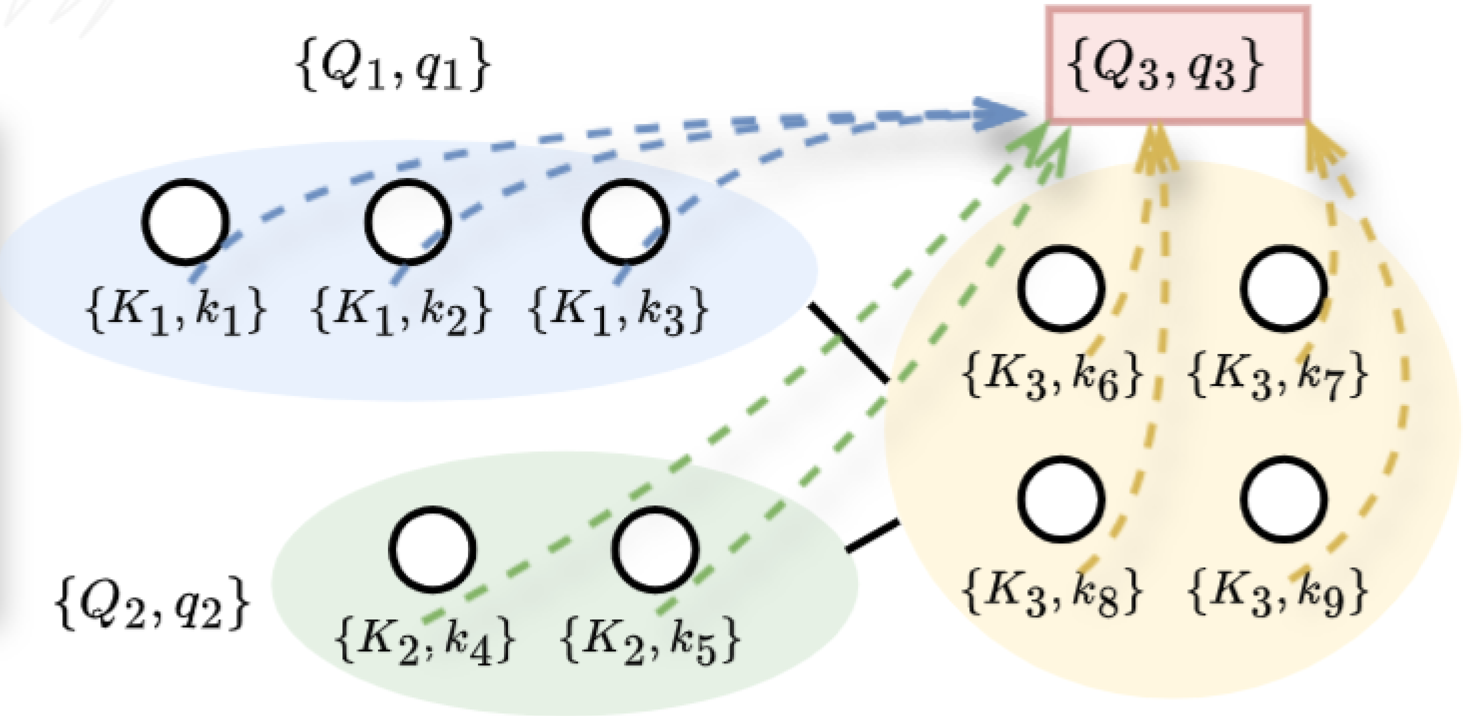
Graph Coarsening Pipeline:

1. Approximately homogenous cluster representations
2. Loss of fine-grained node-level information

Our Method:

1. Adaptive fusion of cluster-level and node-level information (subsection 3.3)
2. Maintain linear computational complexity (subsection 3.2)

Bi-level Q & K	Cluster-level $\mathcal{X}_C$	Node-level $\mathcal{X}_N$
Queries of clusters	Q_1, Q_2, Q_3	q_1, q_2, q_3
Keys of nodes	K_1, K_2, K_3	$k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9$



$$\text{N2C-Attn}(X)_i = \frac{\sum_j A^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_{\text{B}}(\{Q_i, q_i\}, \{K_j, k_t\}) v_t}{\sum_j A^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_{\text{B}}(\{Q_i, q_i\}, \{K_j, k_t\})}$$

where κ_{B} is a bi-level kernel on $\mathcal{X}_C \times \mathcal{X}_N$

Make it more Efficient

- Using the kernelized softmax trick

Make it more Efficient

- Using the kernelized softmax trick, but with a combined kernel!

Make it more Efficient

- Using the kernelized softmax trick, but with a combined kernel!
- Take Node-to-Cluster Attention with Tensor Product of Kernels as an example:

$$\text{N2C-Attn-T}(X)_i = \frac{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C(Q_i, K_j)\, \kappa_N(q_i, k_t) v_t}{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C(Q_i, K_j)\, \kappa_N(q_i, k_t)}$$

Make it more Efficient

- Using the kernelized softmax trick, but with a combined kernel!
- Take Node-to-Cluster Attention with Tensor Product of Kernels as an example:

$$\text{N2C-Attn-T}(X)_i = \frac{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C\left(Q_i, K_j\right) \kappa_N\left(q_i, k_t\right) v_t}{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C\left(Q_i, K_j\right) \kappa_N\left(q_i, k_t\right)}$$

$$= \frac{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C\left(Q_i, K_j\right) \left(\psi(q_i)^T \psi(k_t)\right) v_t}{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C\left(Q_i, K_j\right) \left(\psi(q_i)^T \psi(k_t)\right)}$$

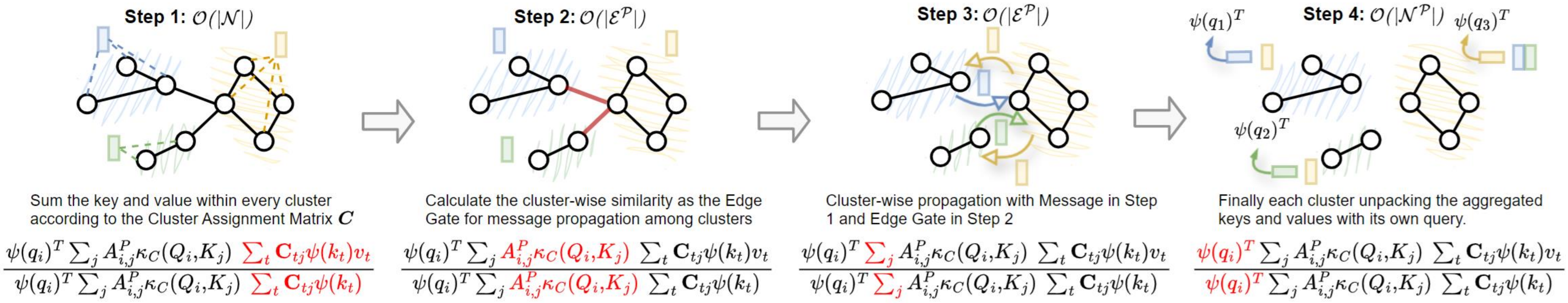
Make it more Efficient

- Using the kernelized softmax trick, but with a combined kernel!
- Take Node-to-Cluster Attention with Tensor Product of Kernels as an example:

$$
\begin{aligned}
\text{N2C-Attn-T}(X)_i &= \frac{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C\left(Q_i, K_j\right) \kappa_N\left(q_i, k_t\right) v_t}{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C\left(Q_i, K_j\right) \kappa_N\left(q_i, k_t\right)} \\
&= \frac{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C\left(Q_i, K_j\right) \left(\psi(q_i)^T \psi(k_t)\right) v_t}{\sum_j \mathbf{A}^P_{i,j} \sum_t \mathbf{C}_{tj} \kappa_C\left(Q_i, K_j\right) \left(\psi(q_i)^T \psi(k_t)\right)} \\
&= \frac{\psi(q_i)^T \sum_j \mathbf{A}^P_{i,j} \kappa_C\left(Q_i, K_j\right) \sum_t \mathbf{C}_{tj} \psi(k_t) v_t}{\psi(q_i)^T \sum_j \mathbf{A}^P_{i,j} \kappa_C\left(Q_i, K_j\right) \sum_t \mathbf{C}_{tj} \psi(k_t)}
\end{aligned}
$$

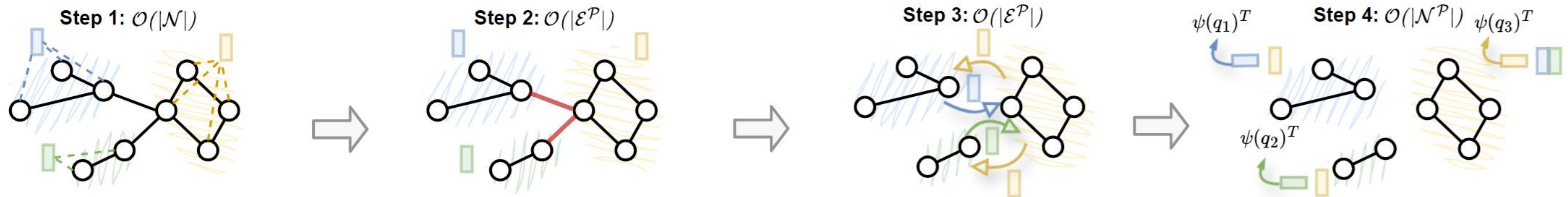
Make it more Efficient

- Using the kernelized softmax trick, but with a combined kernel!
- Take Node-to-Cluster Attention with Tensor Product of Kernels as an example:



Make it more Efficient

- Using the kernelized softmax trick, but with a combined kernel!
- Take Node-to-Cluster Attention with Tensor Product of Kernels as an example:



Sum the key and value within every cluster according to the Cluster Assignment Matrix $\boldsymbol{C}$

$$\frac{\psi(q_i)^T \sum_j A^P_{i,j}\kappa_C(Q_i,K_j) \ \sum_t \mathbf{C}_{tj}\psi(k_t)v_t}{\psi(q_i)^T \sum_j A^P_{i,j}\kappa_C(Q_i,K_j) \ \sum_t \mathbf{C}_{tj}\psi(k_t)}$$

Calculate the cluster-wise similarity as the Edge Gate for message propagation among clusters

$$\frac{\psi(q_i)^T \sum_j A^P_{i,j}\kappa_C(Q_i,K_j) \ \sum_t \mathbf{C}_{tj}\psi(k_t)v_t}{\psi(q_i)^T \sum_j A^P_{i,j}\kappa_C(Q_i,K_j) \ \sum_t \mathbf{C}_{tj}\psi(k_t)}$$

Cluster-wise propagation with Message in Step 1 and Edge Gate in Step 2

$$\frac{\psi(q_i)^T \sum_j A^P_{i,j}\kappa_C(Q_i,K_j) \ \sum_t \mathbf{C}_{tj}\psi(k_t)v_t}{\psi(q_i)^T \sum_j A^P_{i,j}\kappa_C(Q_i,K_j) \ \sum_t \mathbf{C}_{tj}\psi(k_t)}$$

Finally each cluster unpacking the aggregated keys and values with its own query.

$$\frac{\psi(q_i)^T \sum_j A^P_{i,j}\kappa_C(Q_i,K_j) \ \sum_t \mathbf{C}_{tj}\psi(k_t)v_t}{\psi(q_i)^T \sum_j A^P_{i,j}\kappa_C(Q_i,K_j) \ \sum_t \mathbf{C}_{tj}\psi(k_t)}$$

This process can be implemented as cluster-wise message propagation with PyG/DGL…

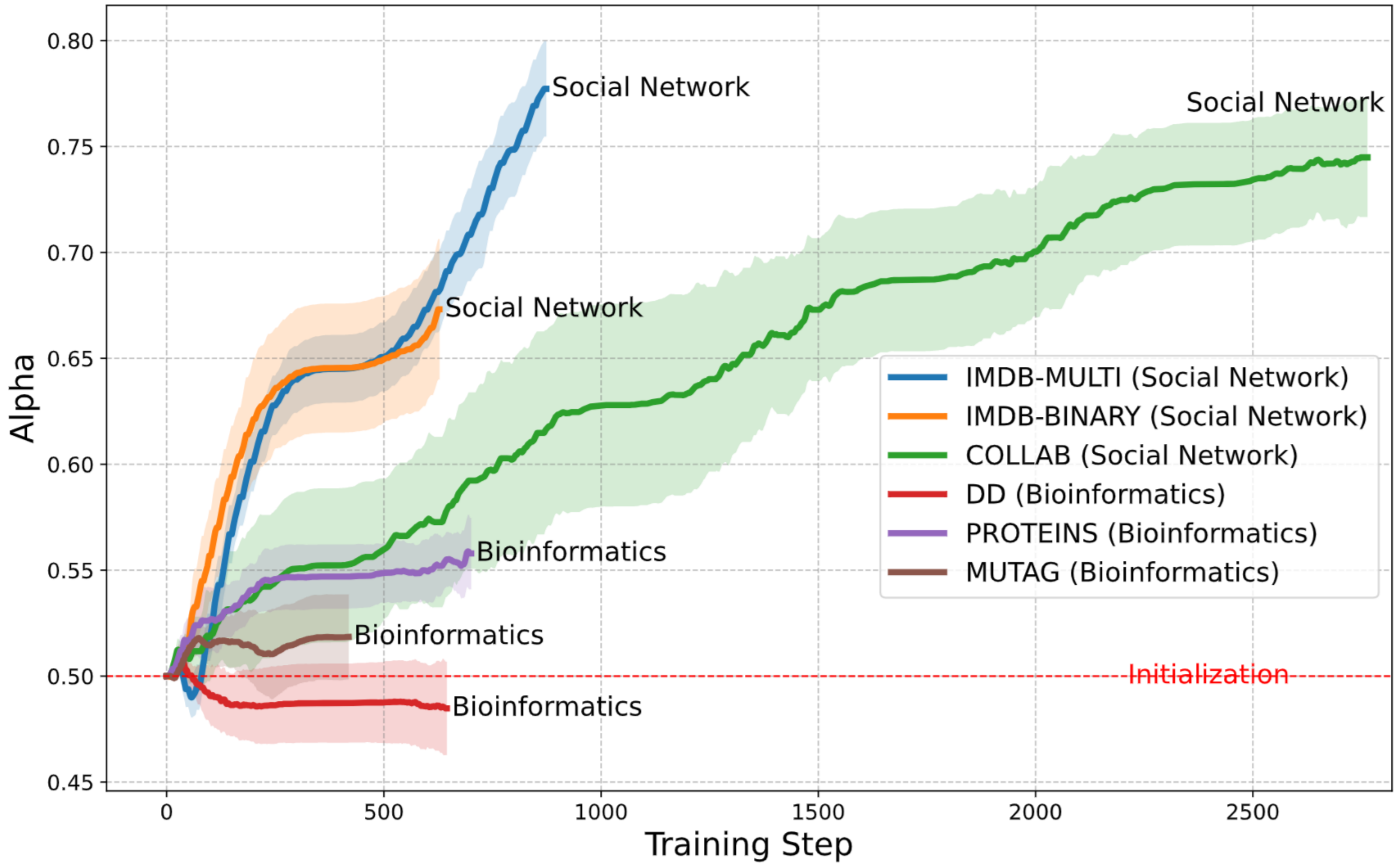
Refer to the paper for more details.

Some Experimental Results

- Visualization of weight of the cluster-level kernel during the training process

Some Experimental Results

- Visualization of weight of the cluster-level kernel during the training process
  - For social network datasets, N2C-Attn prefers cluster-level information
  - For biology datasets, N2C-Attn balances its attention more equally between both granularities

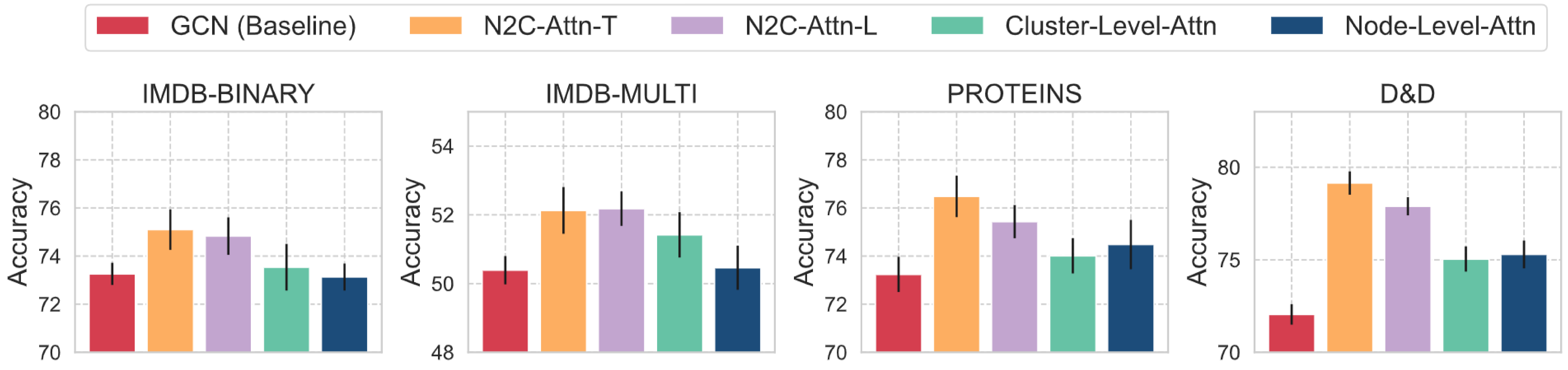


Some Experimental Results

- Comparison of attention strategies with different granularities

Some Experimental Results

- Comparison of attention strategies with different granularities
  - We find that the variants that combine attention from both levels significantly surpass those that do not.



Thanks