Less is More:
One-shot Subgraph Reasoning on Large-scale Knowledge Graphs

Zhanke Zhou

Hong Kong Baptist University

with Yongqi Zhang, Jiangchao Yao, Quanming Yao, and Bo Han

ICLR 2024 Paper: https://arxiv.org/pdf/2403.10231.pdf Code: https://github.com/tmlr-group/one-shot-subgraph

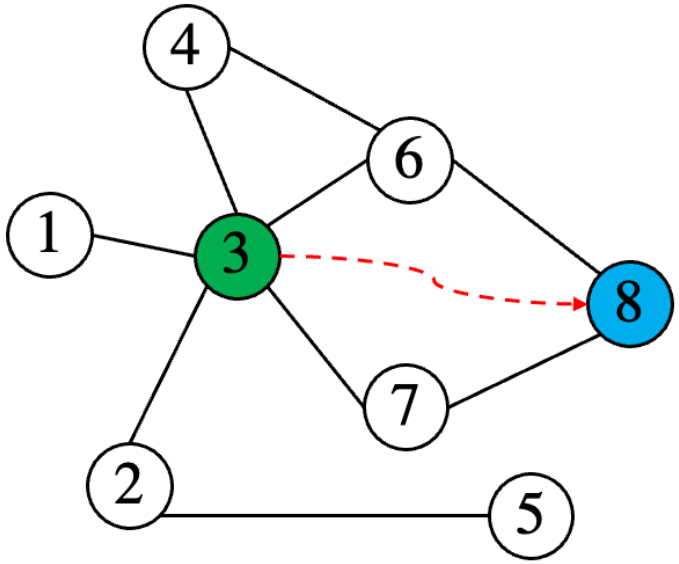
Outline

- Background
- Method
- Experiments
- Summary

Background

Link prediction task in knowledge graph (KG)

- Given a query $(u, q, ?)$, to find the answer v, making (u, q, v) valid
  - u: query entity, q: query relation, v: answer entity
- Namely, to predict the latent (unknown) edges, based on the observed (known) edges



a KG $\mathcal{G}$

$$\text{query} = (u, q, ?)$$

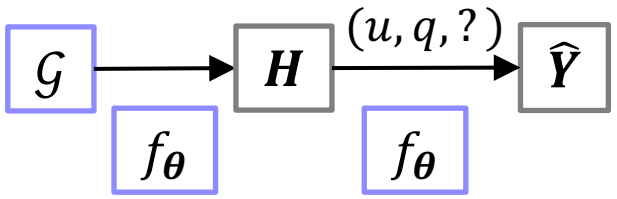
$$u \leftarrow ③, q \leftarrow \text{relation}, v \leftarrow ⑧$$

Background

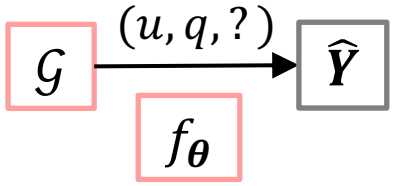
Two classes of existing works

- **semantic models** (computation-efficient but parameter-expensive)
  - $p(u, q, v)$ is measured by a scoring function, utilizing their representations $\boldsymbol{h}_u, \boldsymbol{h}_q, \boldsymbol{h}_v$
- **structural models** (parameter-efficient but computation-expensive)
  - learn the sequential order of structures by leveraging the relational paths between u and v
  - or, directly use the graph structure for reasoning, capturing more complex semantics

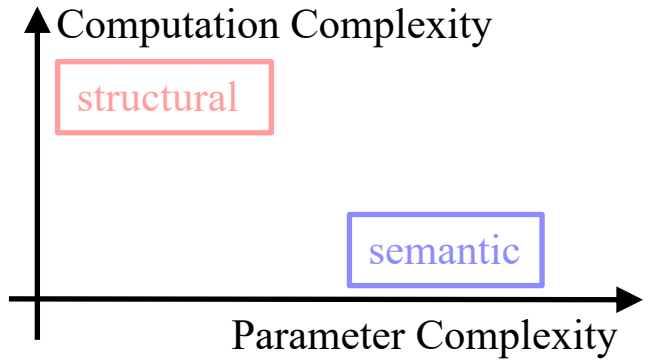
Abstraction:



semantic models



structural models



complexity comparison

➔ **The whole graph ($\mathcal{G}$), model (f_θ), and prediction ($\widehat{Y}$) are coupled**

➔ **f_θ acts on $\mathcal{G}$ to obtain $\widehat{Y}$ of all entities**

Research Problem

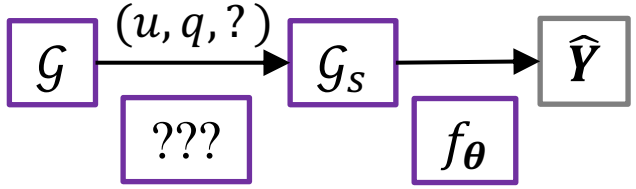
Graph sampling is an intuitive solution, however, existing sampling methods are not good enough

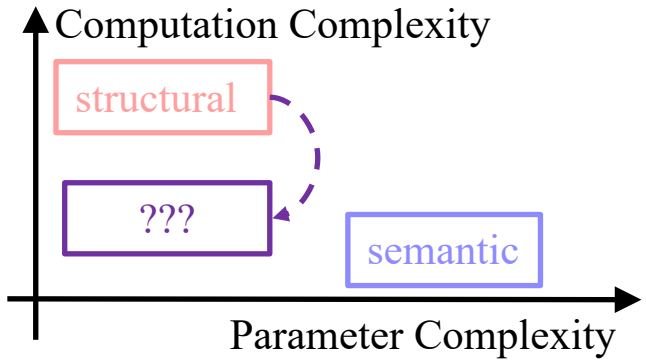
- ***non-learnable sampling methods*** **are designed to solve scalability issues of** ***node-level tasks***
  - e.g., GraphSAGE, FastGCN, and Cluster-GCN
  - cannot guarantee the coverage of answer entities
  - do not perform good on KGs

fast but not good

- ***learnable sampling methods*** **are bundled with** ***specific GNN models***
  - e.g., DPMPN, AdaProp, and AStarNet
  - the sampling and reasoning in each layer are highly coupled
  - the computation cost can be still high on large-scale KGs

good but not fast





➔ *how to efficiently and effectively conduct subgraph reasoning on KG?* 🤔

Outline

- Background
- Method
- Experiments
- Summary

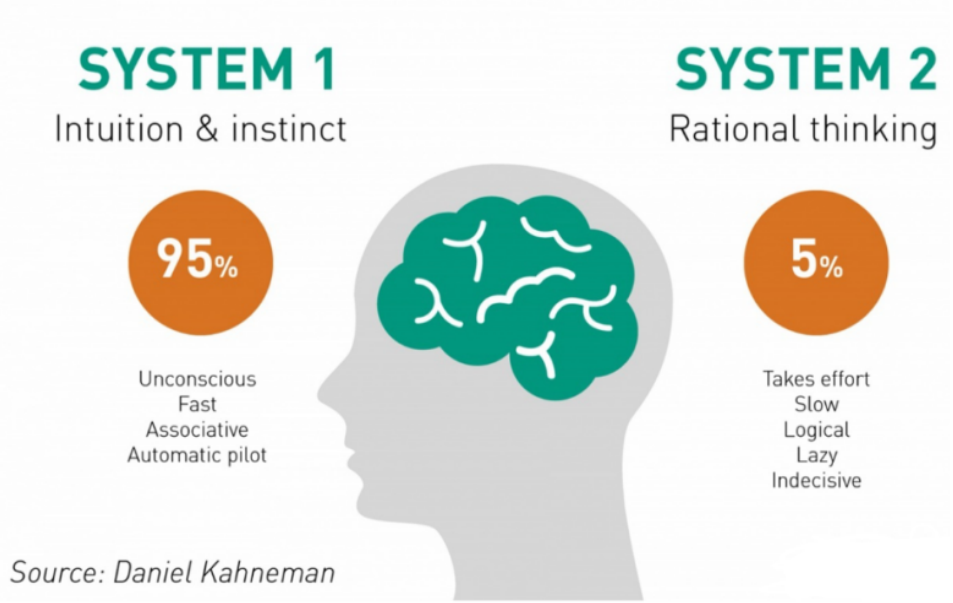
Motivation

Only partial knowledge stored in human brain is relevant to a question

- extracted by recalling
- and then utilized in the careful thinking procedure

Generating candidates and then ranking the promising ones are common

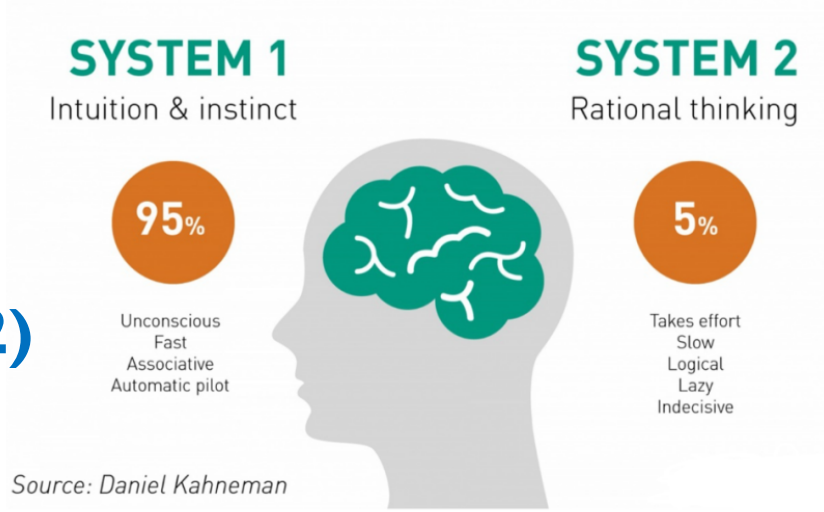
- in large-scale recommendation system
- for handling millions even billions of users and items



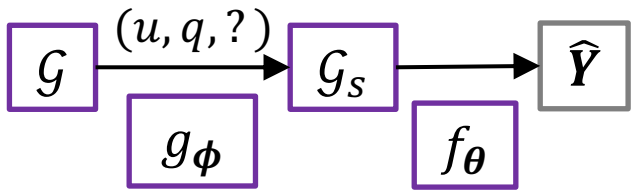
One-shot-subgraph link prediction on KGs

Design principle

- first to efficiently **identify a subgraph (system1)**
  - relevant to the given query
- then effectively **reason on the subgraph (system2)**
  - to obtain the precise ranking results



Abstraction:



two key components

- sampler g_{ϕ} efficiently samples a subgraph
- predictor f_{θ} effectively reasons on the subgraph

➔ decoupling predictor f_{θ} and original graph $\mathcal{G}$

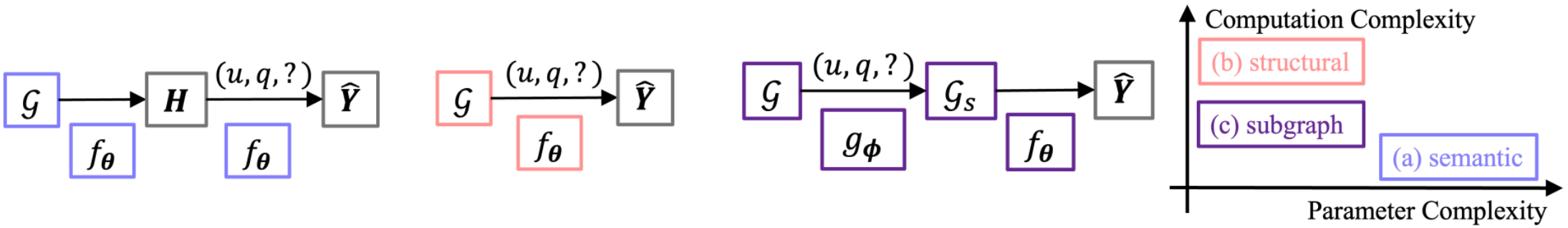
➔ only require subgraph $\mathcal{G}_s$ for reasoning

Formal Definition

Definition 1 (One-shot-subgraph Link Prediction on Knowledge Graphs). *Instead of directly predicting on the original graph $\mathcal{G}$, the prediction procedure is decoupled to two-fold: (1) one-shot sampling of a query-dependent subgraph and (2) prediction on this subgraph. The prediction pipeline becomes*

$$\mathcal{G} \xmapsto{g_{\phi},(u,q)} \mathcal{G}_s \xmapsto{f_{\theta}} \hat{\mathbf{Y}}, \tag{1}$$

where the sampler g_{ϕ} generates only one subgraph $\mathcal{G}_s$ (satisfies $|\mathcal{V}_s| \ll |\mathcal{V}|, |\mathcal{E}_s| \ll |\mathcal{E}|$) conditioned on the given query $(u, q, ?)$. Based on subgraph $\mathcal{G}_s$, the predictor f_{θ} outputs the final predictions $\hat{\mathbf{Y}}$.



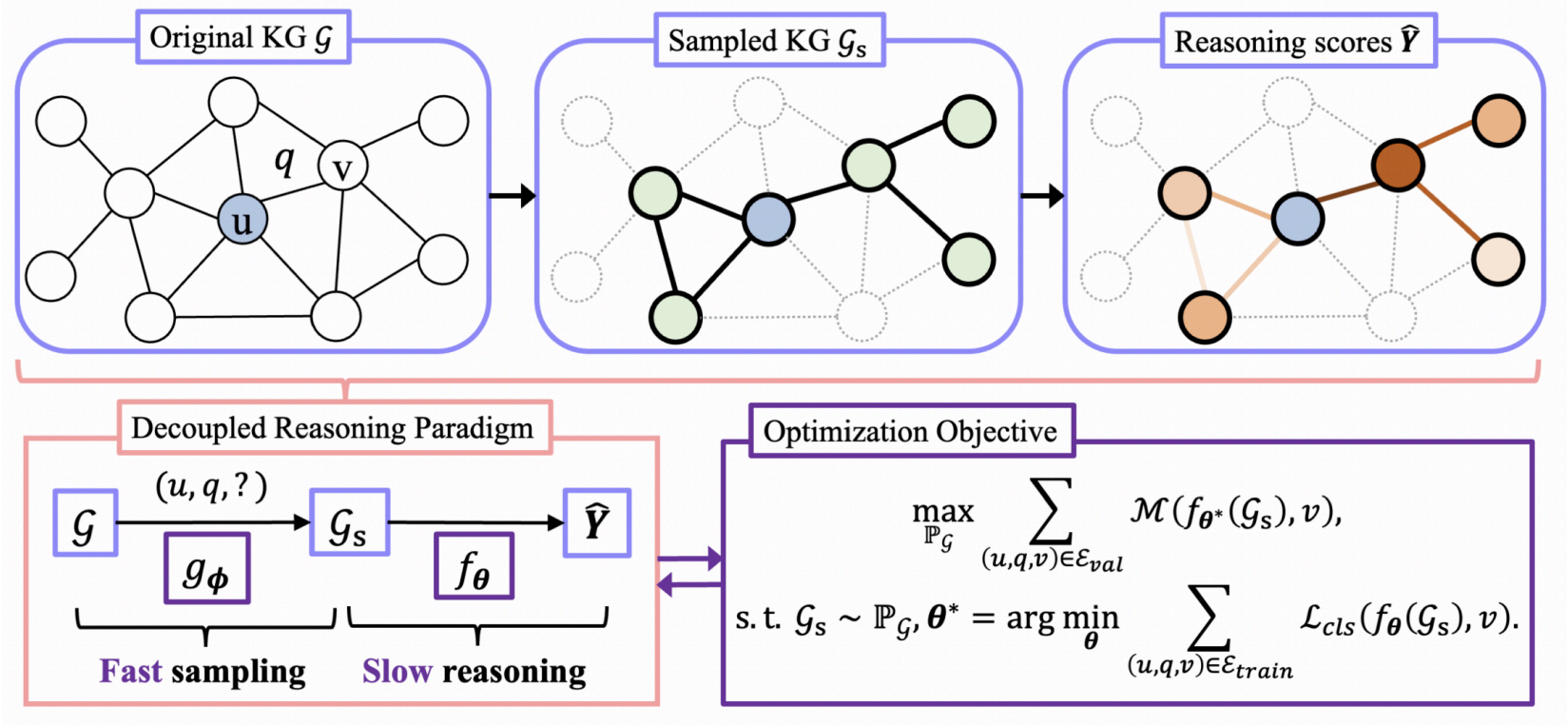
(a) semantic model (b) structural model (c) one-shot-subgraph model (d) complexity comparison

Implementation | overview

The three key steps of one-shot-subgraph LP are

1. generate the sampling distribution
2. extract a subgraph with top entities and edges
3. inference on the subgraph and get the final prediction

Q: how to build the predictor's architecture



Q: what kind of sampler is suitable here?

Q: how to optimize the sampler and predictor?

Implementation | step1/3 Generate Sampling Distribution

Notice that the answer entity are generally **near** the query entity.

Hence, we choose the single-source and non-parametric heuristic **Personalized PageRank (PPR)** as the indicator for sampling

$\boldsymbol{p}^{(k)}$: the sampling importance of each entity

Specifically, PPR starts propagation from u to evaluate the importance of each neighbor of u and generates the PageRank scores as the sampling probability that encodes the local neighborhood of the query entity u. Besides, it can also preserve the locality and connectivity of subgraphs by leveraging the information from a large neighborhood. Given a query entity u, we obtain the probability $\boldsymbol{p} \in \mathbb{R}^{|\mathcal{V}|}$

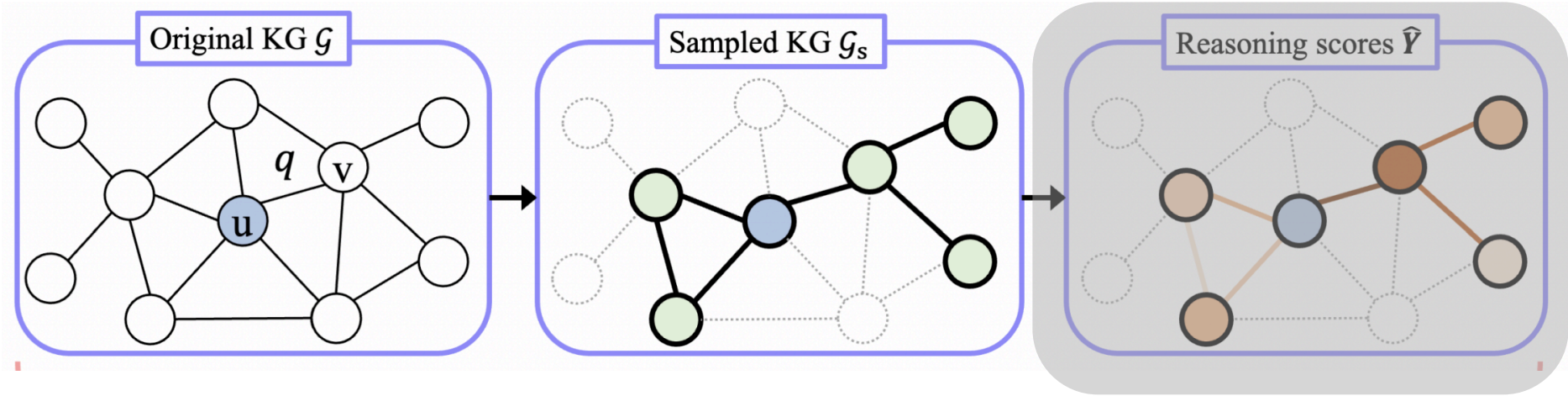
$$\texttt{Non-parametric indicator}: \boldsymbol{p}^{(k+1)} \leftarrow \alpha \cdot \boldsymbol{s} + (1-\alpha) \cdot \boldsymbol{D}^{-1}\boldsymbol{A} \cdot \boldsymbol{p}^{(k)}, \tag{2}$$

by iteratively updating the scores up to $K = 100$ steps to approximate the converged scores efficiently. Here, the initial score $\boldsymbol{p}^{(0)} = \boldsymbol{s} = \mathbb{1}(u) \in \{0,1\}^{|\mathcal{V}|}$ indicates the query entity u to be explored. The two-dimensional degree matrix $\boldsymbol{D} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ and adjacency matrix $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ together work as the transition matrix, wherein $\boldsymbol{A}_{ij} = 1$ means an edge $(i, r, j) \in \mathcal{E}$ and $\boldsymbol{D}_{ij} = \text{degree}(v_i)$ if $i = j$ else $\boldsymbol{D}_{ij} = 0$. The damping coefficient α ($= 0.85$ by default) controls the differentiation degree.

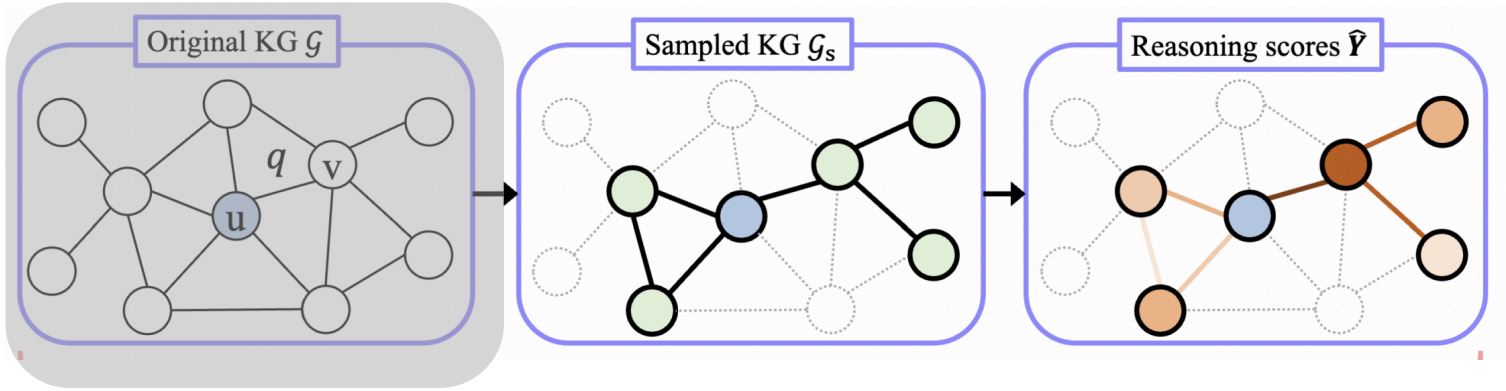
Implementation | step2/3 Extract Subgraph

Step-2. Extract a subgraph. Based on the PPR scores $\boldsymbol{p}$ (Eqn. 2), the subgraph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \mathcal{R}_s)$ (where $\mathcal{R}_s = \mathcal{R}$) is extracted with the most important entities and edges. Denoting the sampling ratios of entities and edges as $r^q_{\mathcal{V}}, r^q_{\mathcal{E}} \in (0, 1]$ that depend on the query relation q, we sample $|\mathcal{V}_s| = r^q_{\mathcal{V}} \times |\mathcal{V}|$ entities and $|\mathcal{E}_s| = r^q_{\mathcal{E}} \times |\mathcal{E}|$ edges from the full graph $\mathcal{G}$. With the $\texttt{TopK}(D, P, K)$ operation that picks up top-K elements from candidate D *w.r.t.* probability P, the entities $\mathcal{V}_s$ and edges $\mathcal{E}_s$ are given as

$$\begin{aligned} \texttt{Entity Sampling:}\ \mathcal{V}_s &\leftarrow \texttt{TopK}\left(\mathcal{V},\ \boldsymbol{p},\ K = r^q_{\mathcal{V}} \times |\mathcal{V}|\right), \\ \texttt{Edge Sampling:}\ \mathcal{E}_s &\leftarrow \texttt{TopK}\left(\mathcal{E},\ \{\boldsymbol{p}_x \cdot \boldsymbol{p}_o : x, o \in \mathcal{V}_s, (x, r, o) \in \mathcal{E}\},\ K = r^q_{\mathcal{E}} \times |\mathcal{E}|\right). \end{aligned} \tag{3}$$



Implementation | step3/3 Reason on the Subgraph



$$\texttt{Indicating:}\ \boldsymbol{h}_o^0 \leftarrow \mathbb{1}(o=u),$$

$$\texttt{Propagation:}\ \boldsymbol{h}_o^{l+1} \leftarrow \texttt{DROPOUT}\Big(\texttt{ACT}\Big(\texttt{AGG}\big\{\texttt{MESS}(\boldsymbol{h}_x^l, \boldsymbol{h}_r^l, \boldsymbol{h}_o^l) : (x,r,o) \in \mathcal{E}_s\big\}\Big)\Big)$$

intra-layer design

DROPOUT$(\cdot)$	**ACT**$(\cdot)$	**AGG**$(\cdot)$	**MESS**$(\cdot)$	**Dimension**
(0, 0.5)	Identity, Relu, Tanh	Max, Mean, Sum	$\mathrm{M}_{\mathrm{DRUM}}$, $\mathrm{M}_{\mathrm{NBFNet}}$, $\mathrm{M}_{\mathrm{REDGNN}}$	16, 32, 64, 128

inter-layer design

No. layers (L)	**Repre. initialization**	**Layer-wise shortcut**	**Repre. concatenation**	**READOUT**$(\cdot)$
{4, 6, 8, 10}	Binary, Relational	True, False	True, False	Linear, Dot product

Implementation | the full algorithm

Algorithm 1 One-shot-subgraph Link Prediction on Knowledge Graphs

Require: KG $\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{E})$, degree matrix $\boldsymbol{D} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, adjacency matrix $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$, damping coefficient α, maximun PPR iterations K , query $(u, q, ?)$, sampler $g_{\boldsymbol{\phi}}$, predictor $f_{\boldsymbol{\theta}}$.

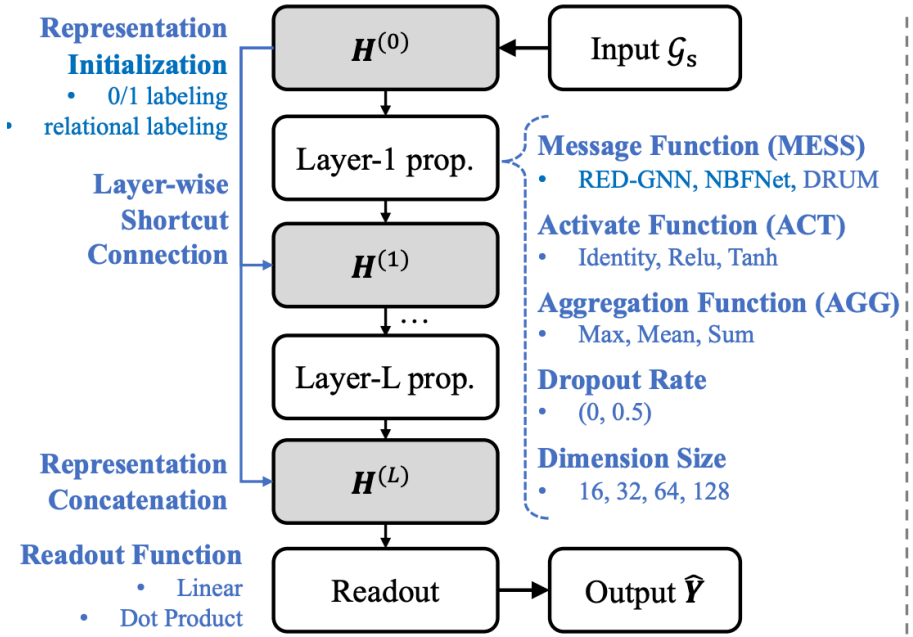
1: # Step-1. Generate sampling distribution
2: initialize $\boldsymbol{s} \leftarrow \mathbb{1}(u)$, $\boldsymbol{p}^{(0)} \leftarrow \mathbb{1}(u)$.
3: **for** $k = 1 \ldots K$ **do**
4: $\quad \boldsymbol{p}^{(k+1)} \leftarrow \alpha \cdot \boldsymbol{s} + (1-\alpha) \cdot \boldsymbol{D}^{-1}\boldsymbol{A} \cdot \boldsymbol{p}^{(k)}$.
5: **end for**
6: # Step-2. Extract a subgraph $\mathcal{G}_s$
7: $\mathcal{V}_s \leftarrow \texttt{TopK}(\mathcal{V},\ \boldsymbol{p},\ K = r_{\mathcal{V}}^{q} \times |\mathcal{V}|)$.
8: $\mathcal{E}_s \leftarrow \texttt{TopK}(\mathcal{E},\ \{\boldsymbol{p}_u \cdot \boldsymbol{p}_v : u, v \in \mathcal{V}_s, (u, r, v) \in \mathcal{E}\},\ K = r_{\mathcal{E}}^{q} \times |\mathcal{E}|)$.
9: # Step-3. Reason on the subgraph
10: initialize representations $\boldsymbol{h}_o^{(0)} \leftarrow \mathbb{1}(o = u)$.
11: **for** $\ell = 1 \ldots L$ **do**
12: $\quad \boldsymbol{h}_o^{(\ell)} \leftarrow \texttt{DROPOUT}(\texttt{ACT}(\texttt{AGG}\{\texttt{MESS}(\boldsymbol{h}_x^{(\ell-1)}, \boldsymbol{h}_r^{(\ell-1)}, \boldsymbol{h}_o^{(\ell-1)}) : (x, r, o) \in \mathcal{E}_s\}))$.
13: **end for**
14: **return** Prediction $\hat{\boldsymbol{y}}_o = \texttt{Readout}(\boldsymbol{h}_o^{(L)}, \boldsymbol{h}_u^{(L)})$ for each entity $o \in \mathcal{V}_s$.

hyperparameters $r_{\mathcal{V}}, r_{\mathcal{E}}$ and L are important
but how to find the optimal configure? 🤔

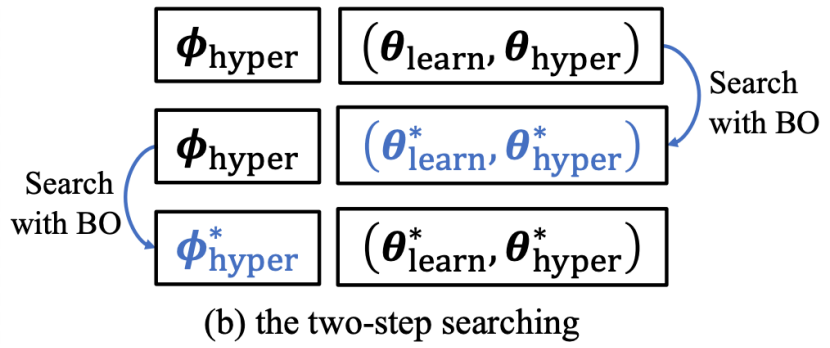
Implementation | optimization

Search Problem to find the optimal configuration ϕ^*_{hyper}

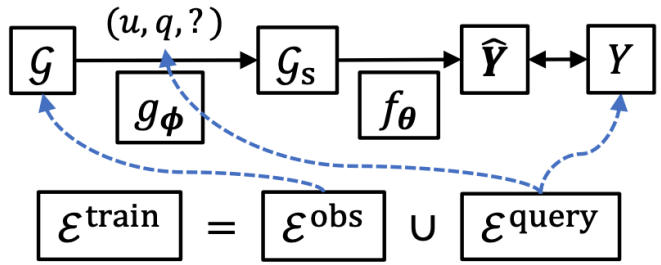
$$\phi^*_{\text{hyper}} = \arg\max_{\phi_{\text{hyper}}} \mathcal{M}(f_{(\theta^*_{\text{hyper}}, \theta^*_{\text{learn}})}, g_{\phi_{\text{hyper}}}, \mathcal{E}^{\text{val}}),$$
$$\text{s.t. } \theta^*_{\text{hyper}} = \arg\max_{\theta_{\text{hyper}}} \mathcal{M}(f_{(\theta_{\text{hyper}}, \theta^*_{\text{learn}})}, g_{\bar{\phi}_{\text{hyper}}}, \mathcal{E}^{\text{val}}), \quad (5)$$



(a) the search space of θ_{hyper}



(b) the two-step searching



(c) the edge split trick

Outline

- Background
- Method
- Experiments
- Summary

Experiments | main results

Table 1: Empirical results of WN18RR, NELL-995, YAGO3-10 datasets. Best performance is indicated by the **bold face** numbers, and the underline means the second best. "–" means unavailable results. "H@1" and "H@10" are short for Hit@1 and Hit@10 (in percentage), respectively.

type	models	WN18RR			NELL-995			YAGO3-10		
		MRR↑	H@1↑	H@10↑	MRR↑	H@1↑	H@10↑	MRR↑	H@1↑	H@10↑
Semantic Models	ConvE	0.427	39.2	49.8	0.511	44.6	61.9	0.520	45.0	66.0
	QuatE	0.480	44.0	55.1	0.533	46.6	64.3	0.379	30.1	53.4
	RotatE	0.477	42.8	57.1	0.508	44.8	60.8	0.495	40.2	67.0
Structural Models	MINERVA	0.448	41.3	51.3	0.513	41.3	63.7	–	–	–
	DRUM	0.486	42.5	58.6	0.532	46.0	**66.2**	0.531	45.3	67.6
	RNNLogic	0.483	44.6	55.8	0.416	36.3	47.8	0.554	50.9	62.2
	CompGCN	0.479	44.3	54.6	0.463	38.3	59.6	0.489	39.5	58.2
	DPMPN	0.482	44.4	55.8	0.513	45.2	61.5	0.553	48.4	67.9
	NBFNet	0.551	49.7	**66.6**	0.525	45.1	63.9	0.550	47.9	68.3
	RED-GNN	0.533	48.5	62.4	0.543	47.6	65.1	0.559	48.3	68.9
	one-shot-subgraph	**0.567**	**51.4**	**66.6**	**0.547**	**48.5**	65.1	**0.606**	**54.0**	**72.1**

Table 2: Empirical results of two OGB datasets (Hu et al., 2020) with regard to official leaderboards.

type	models	OGBL-BIOKG			OGBL-WIKIKG2		
		Test MRR↑	Valid MRR↑	#Params↓	Test MRR↑	Valid MRR↑	#Params↓
Semantic Models	TripleRE	0.8348	0.8360	469,630,002	0.5794	0.6045	500,763,337
	AutoSF	0.8309	0.8317	93,824,000	0.5458	0.5510	500,227,800
	PairRE	0.8164	0.8172	187,750,000	0.5208	0.5423	500,334,800
	ComplEx	0.8095	0.8105	187,648,000	0.4027	0.3759	1,250,569,500
	DistMult	0.8043	0.8055	187,648,000	0.3729	0.3506	1,250,569,500
	RotatE	0.7989	0.7997	187,597,000	0.4332	0.4353	1,250,435,750
	TransE	0.7452	0.7456	187,648,000	0.4256	0.4272	1,250,569,500
Structural Models	**one-shot-subgraph**	**0.8430**	**0.8435**	**976,801**	**0.6755**	**0.7080**	**6,831,201**

Experiments | ablation study

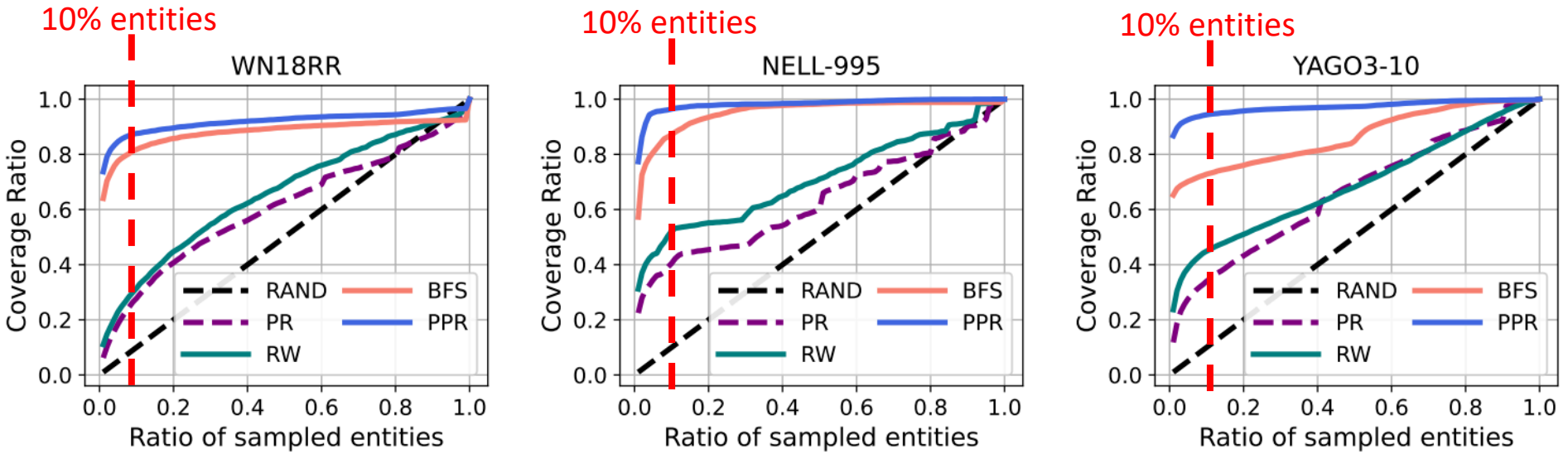


Table 3: Coverage Ratio of different heuristics. **Bold face** numbers indicate the best results in column.

heuristics	WN18RR			NELL-995			YAGO3-10		
	$r_{\mathcal{V}}^{q}=0.1$	$r_{\mathcal{V}}^{q}=0.2$	$r_{\mathcal{V}}^{q}=0.5$	$r_{\mathcal{V}}^{q}=0.1$	$r_{\mathcal{V}}^{q}=0.2$	$r_{\mathcal{V}}^{q}=0.5$	$r_{\mathcal{V}}^{q}=0.1$	$r_{\mathcal{V}}^{q}=0.2$	$r_{\mathcal{V}}^{q}=0.5$
Random Sampling (RAND)	0.100	0.200	0.500	0.100	0.200	0.500	0.100	0.200	0.500
PageRank (PR)	0.278	0.407	0.633	0.405	0.454	0.603	0.340	0.432	0.694
Random Walk (RW)	0.315	0.447	0.694	0.522	0.552	0.710	0.449	0.510	0.681
Breadth-first-searching (BFS)	0.818	0.858	0.898	0.872	0.935	0.982	0.728	0.760	0.848
Personalized PageRank (PPR)	**0.876**	**0.896**	**0.929**	**0.965**	**0.977**	**0.987**	**0.943**	**0.957**	**0.973**

Experiments | efficiency comparison

Table 7: Comparison of efficiency with an 8-layer predictor and different $r^q_{\mathcal{V}}, r^q_{\mathcal{E}}$.

phase	$r^q_{\mathcal{V}}$	$r^q_{\mathcal{E}}$	WN18RR Time	WN18RR Memory	NELL-995 Time	NELL-995 Memory	YAGO3-10 Time	YAGO3-10 Memory
Training	1.0	1.0	Out of memory		Out of memory		Out of memory	
	0.5	0.5	26.3m	20.3GB	1.6h	20.1GB	Out of memory	
	0.2	1.0	12.8m	20.2GB	1.2h	18.5GB	Out of memory	
	0.2	0.2	6.7m	6.4GB	0.6h	8.9GB	2.1h	23.1GB
	0.1	1.0	7.2m	9.8GB	0.8h	12.1GB	1.3h	13.9GB
	0.1	0.1	6.6m	5.1GB	0.3h	5.3GB	0.9h	10.2GB
Inference	1.0	1.0	7.3m	6.7GB	17.5m	12.8GB	1.6h	15.0GB
	0.5	0.5	6.0m	4.3GB	8.3m	4.5GB	1.1h	10.1GB
	0.2	1.0	3.2m	5.8GB	4.2m	12.1GB	0.7h	14.7GB
	0.2	0.2	2.8m	1.9GB	3.6m	2.5GB	0.6h	3.7GB
	0.1	1.0	2.7m	2.7GB	3.1m	9.4GB	0.4h	9.7GB
	0.1	0.1	2.3m	1.7GB	2.9m	1.9GB	0.4h	3.1GB

Outline

- Background
- Method
- Experiments
- Summary

Summary

Main contributions

- We propose **a new manner of one-shot-subgraph reasoning** on KGs to alleviate the scalability problem of existing methods and achieve efficient as well as adaptable learning on KGs
- We further introduce **the automated searching for adaptive configurations** in both data space and model space that benefits from the high efficiency of subgraph reasoning
- **Extensive experiments** on three common datasets and two large-scale benchmarks show that our method achieves leading performances with significantly improved effectiveness and efficiency

Extension

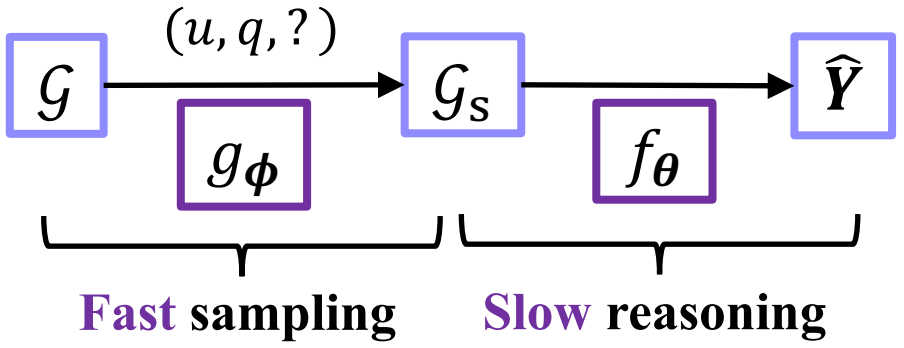
- adapt the decoupled reasoning framework to **other graph learning tasks**
  - e.g., sample a local subgraph for node classification or a global subgraph for graph classification
- enhancing the one-shot-subgraph reasoning with **instance-wise adaptation**
  - e.g., sampling a subgraph of suitable scale for each given query

Take home message

*how to **efficiently** and **effectively** conduct **subgraph reasoning** on KG?* 🤔

[FAST Sampling]
To identify a query-dependent subgraph without learning

[SLOW Reasoning]
To build an expressive GNN that is adaptive to the extracted subgraph



Thanks for your listening!

Zhanke Zhou cszkzhou@comp.hkbu.edu.hk