Reasoning over Coupled Receptive Fields: Eliminating Subgraph Redundancy at Scale

Jing Yang
Zhengzhou University
Zhengzhou, China
yangjinghust123@gmail.com

Bo Wen
Zhengzhou University
Zhengzhou, China
bovane@gs.zzu.edu.cn

Yuan Gao*
Zhengzhou University
Zhengzhou, China
yuangaohnu@gmail.com

Xiaowen Jiang
Zhengzhou University
Zhengzhou, China
jxw@hainanu.edu.cn

Zhihao Zhang
Zhengzhou University
Zhengzhou, China
zzh20230902@stu.zzu.edu.cn

Shikun Yan
Zhengzhou University
Zhengzhou, China
ysk2963@stu.zzu.edu.cn

Abstract

Conditional message passing (CMP) currently represents a leading paradigm for knowledge graph reasoning. Since vanilla CMP performs full-graph propagation per query, existing methods adopt subgraph-wise sampling to improve computational efficiency. However, these extracted subgraphs still exhibit high structural overlap. Consequently, the same local structures are repeatedly accessed and processed, leading to substantial redundancy that remains a major bottleneck for scaling reasoning to large knowledge graphs. To address this issue, we propose reasoning over **Co**upled **R**eceptive **F**ields (CoRF) to eliminate subgraph redundancy at scale. Rather than sampling a separate subgraph for each query, CoRF constructs a set of receptive supports that allow multiple queries to share receptive fields, thereby reducing subgraph redundancy. Meanwhile, we introduce boundary padding and residual compensation mechanisms to ensure topological integrity and query coverage. We further incorporate cost-aware load balancing to improve distributed execution efficiency. Comprehensive experiments on state-of-the-art CMP models demonstrate that CoRF achieves up to a 7.3× speedup and a 10.0× reduction in memory usage, without compromising reasoning accuracy.

1 Introduction

Knowledge graph reasoning aims to predict plausible but unobserved facts based on existing facts in a knowledge graph. It serves as an important foundation for knowledge graph completion and has been widely applied to downstream scenarios such as question answering and recommendation systems [1, 2]. A variety of technical approaches have been developed for this task. Knowledge graph embedding methods, such as TransE and RotatE [3, 4], learn low-dimensional representations of entities and relations to score candidate triples. Path-based models, such as MINERVA, DRUM, and RNNLogic [5, 6, 7], learn probabilistic or logical rules to capture relational path patterns in knowledge graphs. While they offer strong interpretability, they typically incur high costs in path exploration and rule composition. Recent knowledge graph reasoning methods based on conditional message passing, such as NBFNet, RED-GNN, Ultra, and DuetGraph [8, 9, 10, 11], capture multi-hop relational evidence by performing query-conditioned message passing over the graph.

However, large-scale knowledge graphs such as Freebase and Wikidata [12, 13] pose increasing challenges to the training efficiency and scalability of knowledge graph reasoning methods. Sampling-

*Corresponding author: Yuan Gao (yuangaohnu@gmail.com).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

based methods reduce computation by localizing message propagation and can be broadly categorized into three types as follows. Representative examples of node-wise sampling methods include GraphSAGE and PinSAGE [14, 15], which sample a fixed number of neighbors for each node at each message passing layer. To alleviate the neighborhood explosion problem in node-wise sampling, FastGCN and GRAPES [16, 17] adopt a layer-wise sampling strategy by sampling a fixed-size subset of nodes at each layer; DPMPN, AdaProp, and A*Net [18, 19, 20] dynamically select more informative local structures during propagation. Subgraph-wise sampling methods, in contrast, construct query-specific subgraphs for each query. GraIL and Meta-iKG [21, 22] extract enclosing subgraphs between the query entity u and each candidate answer v, while GLAR and One-shot [23, 24] construct open subgraphs centered around the query. Although these methods restrict the computation scope of individual queries, they still construct and process local structures independently for each query. As local subgraphs from different queries often heavily overlap, the same structures are repeatedly accessed, loaded, and computed, leading to substantial redundancy.

To illustrate the structural redundancy in subgraph-wise sampling methods, we present aggregated visualizations under different subgraph extraction strategies in Figure 1 and Figure 2. As shown in the figures, the local structures associated with different queries exhibit substantial overlap. As the number of queries increases, high-frequency regions become increasingly concentrated, indicating that many queries repeatedly access similar local evidence. These results suggest that subgraph-wise sampling methods tend to introduce substantial structural redundancy when constructing subgraphs independently for different queries. This, in turn, increases the overhead of subgraph construction, data loading, and repeated computation, thereby limiting the scalability of large-scale knowledge graph reasoning. Additional visualizations are provided in Appendix B.

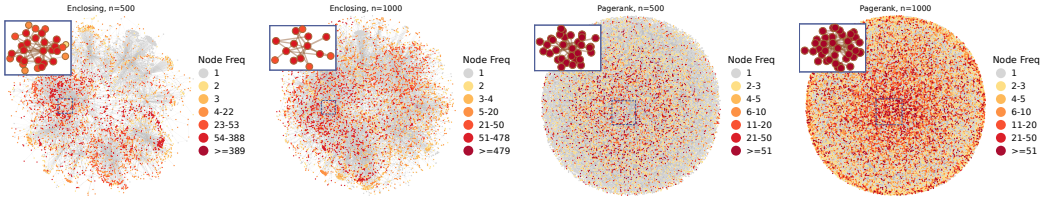


Figure 1: Query subgraph overlap visualization on WN18RR.

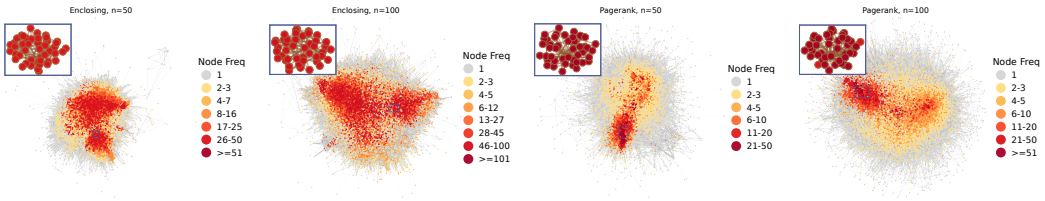
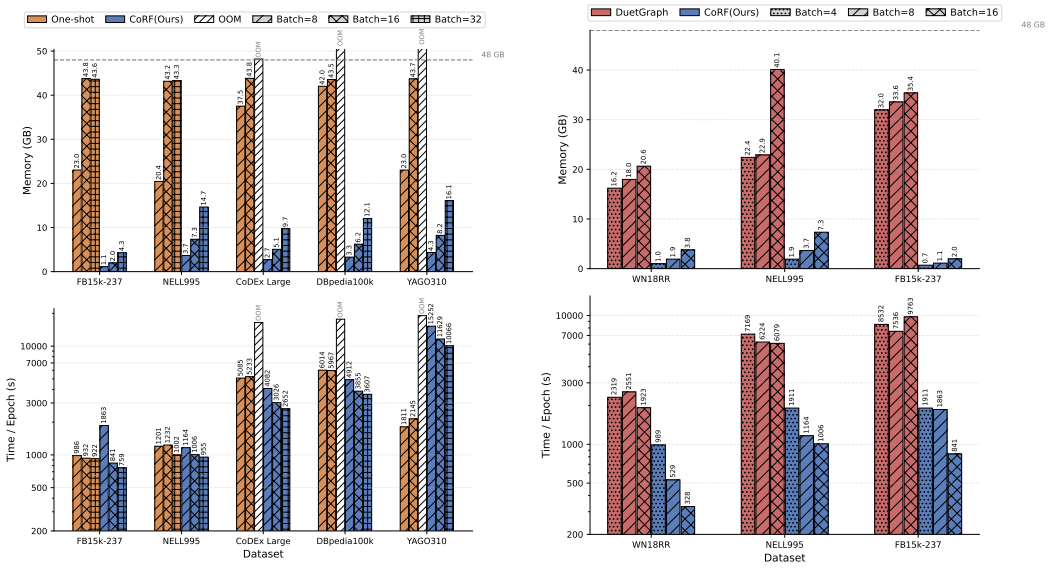


Figure 2: Query subgraph overlap visualization on FB15k237.

Finally, structural redundancy directly translates into system overhead. Figure 3a reports the memory usage and training time of One-shot [24] under different batch sizes. As the batch size and graph scale increase, its memory usage grows rapidly on several datasets. Meanwhile, its per-epoch training time remains high, indicating that per-query subgraph construction and repeated propagation incur substantial overhead. Similarly, Figure 3b shows the memory usage and training time of the recent SOTA method DuetGraph [11] under different batch sizes. Even on medium-scale datasets, DuetGraph still exhibits high memory demand and training cost, indicating that existing conditional message passing models remain limited in efficiency and scalability.

Building on this observation, we propose reasoning over coupled receptive fields (CoRF) to eliminate subgraph redundancy at scale. In contrast to layer-wise and subgraph-wise sampling, CoRF introduces receptive supports as the basic structural units. Each receptive support encapsulates coupled receptive fields that cover the local evidence required by multiple queries. Meanwhile, to ensure topological integrity and query coverage, we introduce boundary padding and residual compensation mechanisms. By integrating these designs, CoRF effectively reduces structural redundancy across



(a) One-shot vs CoRF.

(b) DuetGraph vs CoRF.

Figure 3: Comparison of training time and GPU memory usage under different batch sizes.

subgraphs while preserving the integrity of reasoning evidence, thereby enabling scalable training. We further design a cost-aware load balancing strategy for distributed training and incorporate a hyperparameter optimization module to improve training efficiency and identify better hyperparameter configurations.

Extensive experiments show that CoRF achieves the best reasoning performance on most of the 13 transductive datasets. Its efficiency advantages are particularly pronounced on large-scale datasets: on CoDEx-Large, DBpedia100k, and YAGO310, CoRF achieves on average 5.6× training speedup and 5.0× memory reduction. Moreover, several baselines encounter out-of-memory errors on YAGO310 and Hetionet, whereas CoRF remains trainable and attains strong performance, demonstrating its superior scalability on large knowledge graphs.

2 Preliminaries

In this section, we introduce the preliminaries of this work. For clarity, a summary of the main notations is provided in Appendix A.

Knowledge Graph Reasoning. A knowledge graph is defined as $G = (V, E, R)$, where V is the set of entities, R is the set of relation types, and $E \subseteq V \times R \times V$ is the set of factual triples. Knowledge graph reasoning aims to answer incomplete queries such as $(h, r, ?)$. Head prediction can be converted into tail prediction using inverse relations, so we focus on tail entity prediction.

Conditional Message Passing Model. For knowledge graph reasoning, recent studies utilize graph neural networks based on query-conditioned message passing to model multi-hop relational evidence [8, 9, 11, 19, 20, 10]. Given a query $q = (h, r, ?)$, these methods start from the source entity h and propagate information over the graph conditioned on the query relation r, producing query-specific representations for candidate entities v.

At the ℓ-th propagation layer, each entity v aggregates relation-aware messages from its neighbors:

$$\mathbf{m}_q^{(\ell)}(v) = \mathrm{AGG}\left(\left\{\psi_\theta\left(\mathbf{h}_q^{(\ell-1)}(u), r_{uv}, q\right) \mid (u, r_{uv}, v) \in E\right\}\right), \tag{1}$$

followed by a state update:

$$\mathbf{h}_q^{(\ell)}(v) = \mathrm{UPD}\left(\mathbf{h}_q^{(\ell-1)}(v), \mathbf{m}_q^{(\ell)}(v)\right). \tag{2}$$

Here, ψ_θ denotes the relation-aware message function, AGG the neighborhood aggregation operator, and UPD the state update function.

Definition 1 (Receptive Supports). We refer to the local evidence region associated with a training query as its *receptive field*. We adopt the k-hop receptive field as a general abstraction. To determine whether a support covers a labeled training query, we write the query as $q = (h, r, t)$, where t is the observed target and the corresponding prediction query is $(h, r, ?)$. Its local evidence node set is defined as

$$\mathrm{Ev}_k(q) = \{v \in V \mid \mathrm{dist}(v, \{h, t\}) \leq k\}, \tag{3}$$

where $\mathrm{dist}(\cdot, \cdot)$ denotes the shortest-path distance on the underlying undirected graph that ignores edge directions and relation types. This set characterizes the local evidence region associated with the training query q.

We further introduce *receptive supports* as reusable local structures that cover such receptive fields. Formally, a receptive support is defined as a node subset $S \subseteq V$ together with its induced subgraph $G[S]$. If $\mathrm{Ev}_k(q) \subseteq S$, we say that S covers query q, that is, all local evidence required by q is contained in the support. Since such a support couples the receptive fields of multiple queries through a shared local structure, we refer to it as a *coupled receptive field*.

In the following support construction and training procedure, queries refer to labeled training queries $q = (h, r, t)$, whose set is denoted by $\mathcal{Q}$. For this training query set, both full-graph conditional message passing and subgraph-wise sampling require $|\mathcal{Q}|$ local structures. In contrast, CoRF organizes computation around a reusable set of receptive supports $\mathbb{S}$, where typically $|\mathbb{S}| \ll |\mathcal{Q}|$. This substantially reduces redundant structures, thereby providing advantages in both speed and memory usage and leading to better scalability.

3 Method

Our goal is to construct a set of size-controlled receptive supports that cover the local evidence required by a given knowledge graph and query set, while minimizing the redundant structural cost introduced by subgraph-wise sampling. This objective can be formulated as a combinatorial coverage problem, which is NP-hard in general. We therefore adopt an approximation strategy and impose the following constraints during construction to balance computational efficiency and reasoning performance.

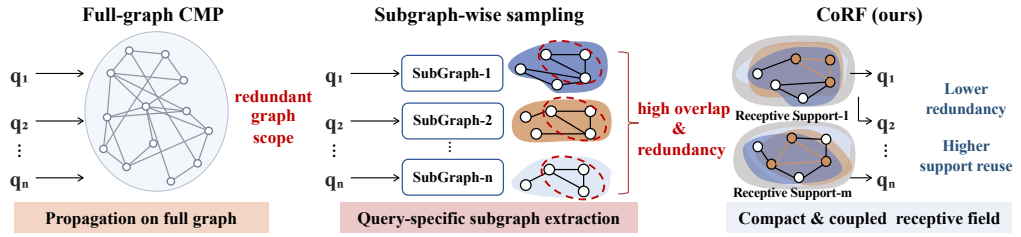
(1) Topological Integrity Constraint. *For a receptive support S, a node $u \in S$ is defined as a boundary node if there exist $v \in V \setminus S$ and $\rho \in R$ such that $(u, \rho, v) \in E$ or $(v, \rho, u) \in E$. Removing such boundary connections may disrupt multi-hop reasoning paths across the support boundary.* Hence, this constraint requires selectively retaining external nodes adjacent to boundary nodes, together with the corresponding edges, in order to preserve the topological connectivity necessary for local reasoning.

(2) Query Coverage Constraint. *Each query is constrained to be covered by at least one receptive support: $\forall q \in \mathcal{Q}, \quad \exists S \in \mathbb{S}, \quad \mathrm{Ev}_k(q) \subseteq S$.* This constraint ensures that every query obtains its required local reasoning evidence from a specific receptive support, while queries with overlapping evidence demands should be jointly covered by the same support whenever possible.

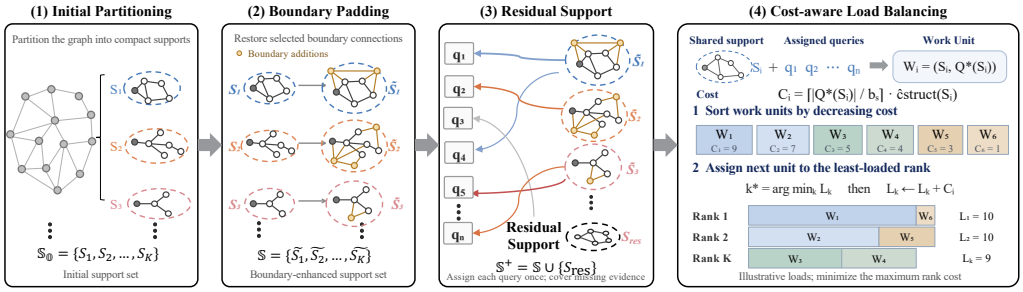
3.1 Coupled Receptive Fields Construction

Based on the optimization objective and constraints introduced above, we describe the construction of coupled receptive fields.

Initial Partitioning. We begin by generating an initial set of receptive supports through topology-preserving partitioning. Since queries with overlapping local evidence demands typically rely on proximal neighborhood structures, we employ general graph partitioning in an algorithm-agnostic manner to preserve local topological connectivity while keeping the support scale under control. Through this process, we obtain the initial set of receptive supports, denoted as $\mathbb{S}_0 = \{S_1, S_2, \ldots, S_K\}$. More details are provided in Appendix C.1.



(a): Comparison with Full-graph CMP and Subgraph-wise sampling



(b): Overall framework of CoRF

Figure 4: Overview of CoRF: (a) comparison with full-graph CMP and subgraph-wise sampling; (b) the overall CoRF framework.

Boundary Padding for Topological Integrity. To satisfy the topological integrity constraint, we introduce a boundary padding mechanism to enhance the initial receptive supports. Specifically, the mechanism selectively incorporates external nodes and restores their connections to S, improving boundary connectivity. We do not include all external boundary nodes to avoid excessive expansion. Instead, we prioritize nodes that are more strongly connected to the current receptive support. Formally, the set of candidate external nodes with respect to S is defined as

$$\mathcal{B}(S) = \{v \in V \setminus S \mid \exists u \in S,\ \exists \rho \in R : (u, \rho, v) \in E \vee (v, \rho, u) \in E\}\,.$$

For each candidate node $v \in \mathcal{B}(S)$, we define its relevance to S as the number of nodes in S connected to v, regardless of edge direction or relation type:

$$\text{score}(v, S) = |\{u \in S \mid \exists \rho \in R : (u, \rho, v) \in E \vee (v, \rho, u) \in E\}|\,. \tag{4}$$

We then select the top-r nodes with the highest $\text{score}(v, S)$, denoted by V_{ext}, and construct the enhanced support as $\tilde{S} = S \cup V_{\text{ext}}$, together with its induced subgraph $G[\tilde{S}]$.

This strategy prioritizes nodes that are tightly connected to the boundary of S, restoring boundary connectivity with minimal overhead. After initial construction and boundary padding, we obtain the enhanced receptive supports $\{\tilde{S}_1, \tilde{S}_2, \ldots, \tilde{S}_K\}$.

Residual Compensation for Coverage. The enhanced set of receptive supports provides a shared structural basis for covering the local evidence required by queries. To formalize query assignment, we first define the coverage set of a query q as the subset of supports that fully contain its required local evidence:

$$\mathcal{H}(q) = \{S \in \mathbb{S} \mid \text{Ev}_k(q) \subseteq S\}.$$

Based on this definition, we address two boundary cases to ensure rigorous and high-performance execution:

Queries Covered by Multiple Supports ($|\mathcal{H}(q)| > 1$): A query may be covered by multiple supports. Executing the query across all of them would introduce redundant computation. To circumvent this, we restrict each redundantly covered query to exactly one effective covering support. To mitigate load imbalance, we preferentially select the support with the fewest currently assigned queries.

Uncovered Queries ($\mathcal{H}(q) = \emptyset$): A query is considered uncovered if it lacks full coverage from any existing support. For these queries, we aggregate their local evidence and construct a residual

support:

$$S_{\text{res}} = \bigcup_{q:\mathcal{H}(q)=\emptyset} \text{Ev}_k(q). \tag{5}$$

By construction, S_{res} contains the complete local evidence set $\text{Ev}_k(q)$ of every previously uncovered query. The final set of supports is then given by $\mathbb{S}^+ = \mathbb{S} \cup \{S_{\text{res}}\}$. With the introduction of the residual support, all queries are guaranteed to be covered by $\mathbb{S}^+$, thereby strictly satisfying the coverage constraint.

Consequently, let $\mathcal{Q}^*(S)$ denote the final set of queries exclusively assigned to support $S \in \mathbb{S}^+$. Compared to standard subgraph-wise sampling methods, which repeatedly perform subgraph extraction and loading per query, our coupled receptive field approach significantly reduces redundant structural overhead. As a result, it substantially improves both time and memory efficiency while preserving the integrity of reasoning performance.

3.2 Distributed Training and Cost-Aware Scheduling

In CoRF, receptive supports differ in scale and the number of covered queries, imposing different workloads. Simply distributing supports evenly across GPUs can therefore cause load imbalance and reduce training throughput. To address this issue, we propose a cost-aware scheduling strategy to better balance workloads in distributed training.

Workload Estimation. We define a work unit (WU) as the basic scheduling unit in distributed execution. Each work unit consists of a receptive support and the set of queries it covers $\mathcal{W} = (S, \mathcal{Q}^*(S))$, where $S = (V_S, E_S)$ denotes the receptive support, and $\mathcal{Q}^*(S)$ is the set of queries ultimately covered by S.

For a work unit $\mathcal{W} = (S, \mathcal{Q}^*(S))$, the model performs L-layer conditional message passing over $S = (V_S, E_S)$. The computational cost is therefore largely governed by the number of edges $|E_S|$ and nodes $|V_S|$.

To combine these factors, we normalize edge and node counts and estimate the per-mini-batch computation cost on S as

$$\widehat{c}_{\text{struct}}(S) = L\left(\frac{|E_S|}{\max_{S' \in \mathbb{S}^+} |E_{S'}|} + \frac{|V_S|}{\max_{S' \in \mathbb{S}^+} |V_{S'}|}\right). \tag{6}$$

Given a batch size bs, the cost of a work unit is estimated as

$$\text{cost}(\mathcal{W}) = \left\lceil \frac{|\mathcal{Q}^*(S)|}{bs} \right\rceil \cdot \widehat{c}_{\text{struct}}(S). \tag{7}$$

Greedy Cost-Aware Scheduling. Given a set of work units $\mathbb{W} = \{\mathcal{W}_1, \ldots, \mathcal{W}_M\}$ and K computational devices, the goal is to partition them into K subsets $\Pi = \{\Pi_1, \ldots, \Pi_K\}$, where Π_k denotes the set of work units scheduled on device k. We aim to balance the workload across devices by minimizing the maximum accumulated cost:

$$\min_{\Pi} \max_{k=1,\ldots,K} \sum_{\mathcal{W} \in \Pi_k} \text{cost}(\mathcal{W}). \tag{8}$$

This problem corresponds to a weighted multi-machine load balancing problem, for which finding the optimal solution is generally intractable. To achieve a good trade-off between scheduling quality and efficiency, we adopt a greedy approximation strategy. Specifically, we first sort all work units in descending order of $\text{cost}(\mathcal{W})$, and then iteratively assign each work unit to the device with the smallest current load.

Let L_k denote the current load of device k. Each work unit $\mathcal{W}$ is then assigned to the device with the minimum current load:

$$L_k = \sum_{\mathcal{W} \in \Pi_k} \text{cost}(\mathcal{W}), \qquad k^* = \arg\min_{k \in \{1,\ldots,K\}} L_k. \tag{9}$$

The overall time complexity of this procedure is $O(M \log M + MK)$, where M is the number of work units. Because this strategy performs load balancing according to the actual workload, it can be naturally extended to more general load balancing scenarios in distributed GNN training.

Hyperparameter Search. CoRF involves both structural and training hyperparameters. To efficiently explore hyperparameter space, we adopt a hierarchical search strategy that first optimizes structural parameters and then tunes training parameters conditioned on the selected structure. Detailed formulations and the full search pipeline are provided in Appendix F.

4 Experiments

In this section, we conduct extensive experiments to systematically evaluate the effectiveness and scalability of CoRF.

4.1 Experimental Setup

Datasets. To comprehensively evaluate our method across different graph scales, we conduct experiments on 13 benchmark datasets for transductive knowledge graph reasoning. As shown in Table 1, these datasets are grouped into small, medium, and large categories based on the numbers of triples and entities. Detailed dataset statistics and evaluation metrics are provided in Appendix D and Appendix E, respectively.

Table 1: Transductive datasets grouped by graph scale.

Scale	Datasets
Small (S)	CoDEx Small, WDsinger, NELL23k
Medium (M)	FB15k237, CoDEx Medium, WN18RR, AristoV4, NELL995
Large (L)	CoDEx Large, ConceptNet100k, DBpedia100k, YAGO310, Hetionet

Baselines. We consider two representative categories of knowledge graph reasoning methods as baselines. The first category is a typical conditional message passing (CMP) approach, **Ultra** [10]. The second category is an advanced subgraph-wise sampling method designed for scalable knowledge graph reasoning, **One-shot** [24]. Due to the inherent trade-off between efficiency and effectiveness in One-shot models, we report results under two configurations: $fact_{ratio} = 0.95, 0.75$, corresponding to the use of approximately 5% and 25% of the training triples, respectively. Since the original paper does not specify the exact data splitting procedure, we randomly partition the training set on all datasets and reproduce One-shot to ensure a fair comparison.

Implementation Details. To ensure a fair comparison of both effectiveness and efficiency, for all methods, we standardize the model and training settings as much as possible, including batch size, hidden dimension (64), and number of layers (6). Other hyperparameters follow the default settings provided in the official implementations. Receptive supports are used only during training. During validation and testing, we follow the backbone model's original inference procedure: given $(h, r, ?)$, the model scores candidate tail entities without support-based query assignment. Ground-truth tails are used for evaluation, not for constructing or selecting the inference graph.

4.2 Overall Performance

Table 2 summarizes the overall results on 13 transductive datasets. Overall, CoRF achieves competitive or superior MRR and Hits@10 on most medium- and large-scale datasets, while substantially reducing training time and memory consumption.

For example, on DBpedia100k (batch size = 16), CoRF achieves a $7.33\times$ training speedup and reduces memory usage by $7.04\times$ compared to One-shot (25%). Compared to Ultra, it achieves a $3.20\times$ speedup and reduces memory usage by $3.06\times$. On YAGO310 (batch size = 16), CoRF achieves a $6.58\times$ speedup and reduces memory usage by $5.30\times$ compared to One-shot (25%), and a $4.72\times$ speedup with $2.80\times$ memory reduction compared to Ultra.

Figure 5 further illustrates the trade-offs between reasoning accuracy and efficiency across different methods. CoRF consistently lies in more favorable regions on most datasets, with particularly pronounced advantages on large-scale benchmarks. These results demonstrate that CoRF signifi-

Table 2: Overall performance on 13 representative transductive datasets.

Dataset	Batch	One-shot (train triples = 5%)				One-shot (train triples = 25%)				Ultra (train triples = 100%)				CoRF(ours)(train triples = 100%)			
		MRR	H@10	S/E(s)↓	M(GB)↓	MRR	H@10	S/E(s)↓	M(GB)↓	MRR	H@10	S/E(s)↓	M(GB)↓	MRR	H@10	S/E(s)↓	M(GB)↓
(L) CoDEx Large	8	0.297	0.411	5085.58	37.52	0.310	0.423	21095.98	22.81	0.357	0.491	14070.02	7.64	0.385	0.518	**4082.27**	**2.72**
	16			5233.92	43.82			20321.29	23.01			11306.24	14.83			**3026.74**	**5.06**
	32			OOM	OOM			OOM	OOM			11864.53	29.17			**2652.29**	**9.73**
(L) ConceptNet100k	8	0.250	0.460	**650.34**	8.42	0.279	0.490	2185.48	6.20	0.350	0.574	1289.24	7.62	0.367	0.599	823.32	**4.17**
	16			**410.42**	16.83			1835.37	12.14			1181.42	14.51			717.75	**8.16**
	32			**411.85**	33.56			2567.32	21.97			1361.54	30.115			670.79	**16.22**
(L) DBpedia100k	8	0.463	0.597	6014.49	42.02	0.471	0.607	23948.23	22.92	0.485	0.630	14500.34	9.83	0.518	0.658	**4912.46**	**3.29**
	16			5967.22	43.54			28248.07	43.86			12357.32	19.07			**3855.35**	**6.23**
	32			OOM	OOM			OOM	OOM			13180.03	39.08			**3607.66**	**12.05**
(L) YAGO310	8	0.595	0.712	**1811.54**	23.03	0.599	0.714	69993.28	23.01	0.593	0.721	74438.18	12.11	0.612	0.736	15252.96	**4.33**
	16			**2145.71**	43.70			76573.03	43.54			54873.79	23.03			11629.73	**8.22**
	32			OOM	OOM			OOM	OOM			OOM	OOM			**10066**	**16.05**
(L) Hetionet	8	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	0.399	0.538	80721.66	8.29	0.482	0.597	**32355.85**	**5.12**
	16			OOM	OOM			OOM	OOM			66653.91	9.71			**25685.60**	**6.81**
	32			OOM	OOM			OOM	OOM			–	–			**25121.37**	**10.25**
(M) FB15k237	8	0.315	0.478	**986**	23.03	0.325	0.490	4226.25	19.74	0.372	0.568	1853.71	1.69	0.427	0.603	1863.22	**1.12**
	16			932.59	43.77			3474.67	22.98			1467.26	3.06			**841.26**	**1.97**
	32			922.78	43.61			3606.00	23.03			1337.00	5.81			**759.00**	**4.28**
(M) CoDEx Medium	8	0.304	0.430	**401.65**	7.45	0.316	0.454	2407.53	5.52	0.379	0.536	1392.50	1.75	0.407	0.554	1272.91	**1.05**
	16			**470.42**	14.58			1832.46	12.75			1140.74	3.34			648.95	**1.89**
	32			**456.29**	33.00			1605.93	2.89			1034.48	6.50			373.53	**3.24**
(M) WN18RR	8	0.539	0.638	**507.82**	1.96	0.546	0.645	2210.52	**1.83**	0.506	0.633	689.86	4.00	0.540	0.641	529.54	1.92
	16			416.84	3.77			807.77	**3.65**			567.24	7.61			**328.39**	3.76
	32			**173.05**	7.77			633.00	**7.26**			577.00	15.76			287.00	7.45
(M) AristoV4	8	0.320	0.436	**4022.52**	43.84	0.330	0.448	27571.18	22.97	0.412	0.547	13687.61	6.26	0.426	0.554	13373.37	**3.72**
	16			**2575.30**	43.92			10166.86	43.86			12188.72	10.67			11007.09	**5.55**
	32			OOM	OOM			OOM	OOM			11828.04	19.77			**11029.68**	**9.27**
(M) NELL995	8	0.537	0.639	1201.90	20.43	0.544	0.647	5062.67	11.6465	0.516	0.660	1879.86	7.04	0.550	0.685	**1164.64**	**3.66**
	16			1232.38	43.16			3329.00	23.03			1661.98	14.51			**1006.97**	**7.34**
	32			1002.24	43.26			4233.06	43.51			1950.72	28.90			**955.00**	**14.65**
(S) CoDEx Small	8	0.393	0.521	**27.49**	3.23	0.412	0.532	133.66	1.39	0.486	0.679	198.24	0.25	0.511	0.692	109.56	**0.24**
	16			**17.02**	2.31			65.79	4.11			123.68	0.49			52.66	0.33
	32			**14.40**	9.90			52.67	6.07			49.22	0.85			52.66	**0.74**
(S) WDsinger	8	0.305	0.393	19.40	0.637	0.315	0.428	169.40	**0.56**	0.417	0.522	94.42	1.12	0.424	0.523	88.89	0.57
	16			**9.24**	1.13			44.80	**1.06**			56.51	2.15			33.45	1.09
	32			**8.75**	2.25			141.88	2.05			36.85	4.17			28.57	**1.95**
(S) NELL23k	8	0.415	0.538	**61.82**	1.16	0.423	0.541	105.92	**1.01**	0.262	0.445	156.74	2.32	0.272	0.452	160.04	1.51
	16			**37.13**	2.04			133.85	**1.86**			175.13	4.57			86.60	2.94
	32			**20.80**	3.70			108.40	**3.27**			88.19	9.06			66.31	5.80

cantly reduces training overhead on large graphs while maintaining competitive reasoning accuracy. Additional results are provided in Appendix G.

4.3 Ablation Study

Effect of Boundary Padding and Residual Support. We conduct ablation studies on seven representative transductive datasets. We consider two variants: removing the boundary padding mechanism (w/o boundary) and removing the Residual Support (w/o residual). As shown in Table 3, the full model consistently achieves the best performance across all datasets, indicating that both the boundary padding and Residual Support mechanisms contribute positively to the final results. Removing boundary padding leads to a clear performance drop, highlighting the importance of preserving boundary topological integrity for multi-hop reasoning. Removing Residual Support also degrades performance, suggesting that even a small number of uncovered queries can still affect training quality and that additional compensation is necessary for complete local evidence coverage.

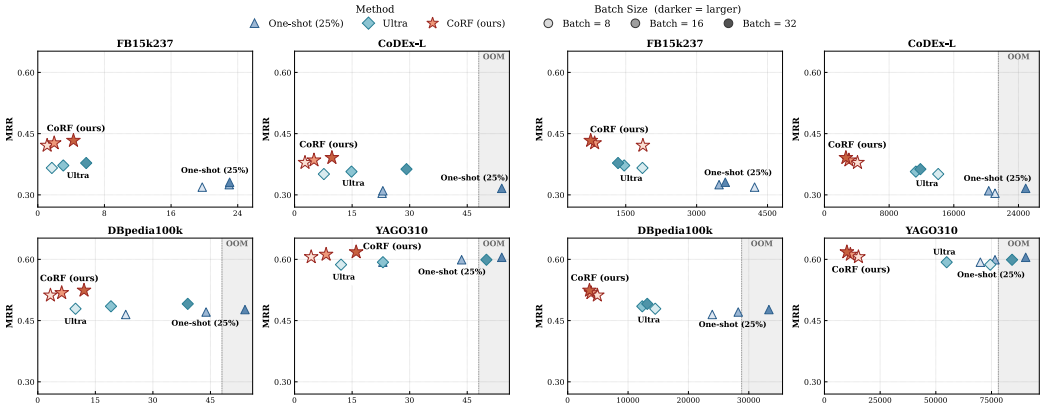
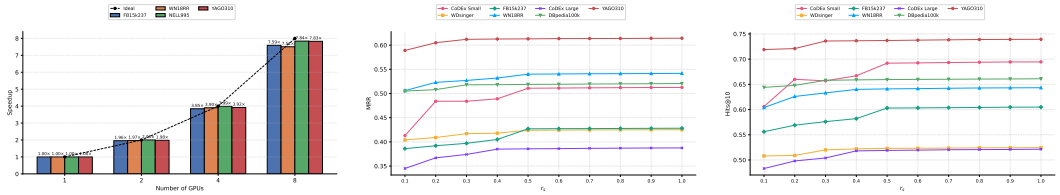


Figure 5: Accuracy–efficiency trade-offs of different methods on transductive datasets. Upper-left is better.



(a) Distributed training speedup of CoRF.

(b) Effect of initial receptive support size on MRR.

(c) Effect of initial receptive support size on Hits@10.

Figure 6: Scalability and sensitivity analysis of CoRF.

Table 3: Ablation and scheduling results.

Dataset	CoRF		w/o bnd.		w/o res.		RR			Random			Cost-aware		
	MRR	H@10	MRR	H@10	MRR	H@10	ΔT	Imb.	ΔM	ΔT	Imb.	ΔM	ΔT	Imb.	ΔM
FB15k237	**0.427**	**0.603**	0.372	0.545	0.389	0.542	248.42	1.82	0.74	358.28	1.80	0.77	20.49	**1.04**	0.34
WN18RR	**0.540**	**0.641**	0.511	0.615	0.519	0.621	29.26	1.15	0.56	54.62	1.40	1.43	11.33	**1.07**	0.21
NELL995	**0.550**	**0.685**	0.512	0.626	0.521	0.642	118.32	1.22	1.48	163.66	1.42	0.83	29.85	**1.05**	0.18
CoDEx-L	**0.385**	**0.518**	0.368	0.482	0.374	0.495	432.82	1.26	2.01	458.39	1.29	1.92	82.62	**1.04**	0.43
DBpedia100k	**0.518**	**0.658**	0.491	0.623	0.504	0.631	502.41	1.25	1.03	336.68	1.16	2.52	55.51	**1.05**	0.35
YAGO310	**0.612**	**0.736**	0.582	0.667	0.601	0.682	–	–	–	–	–	–	96.65	**1.07**	1.76
Hetionet	**0.482**	**0.597**	0.452	0.563	0.461	0.576	–	–	–	–	–	–	220.80	**1.10**	0.44

Cost-aware Scheduling and Distributed Scaling. To further evaluate the effectiveness of the proposed cost-aware scheduling strategy, we compare it against two widely used load balancing baselines, namely random scheduling (Random) and round-robin scheduling (Round-robin), which are commonly adopted in distributed graph learning and high-performance computing [25, 26]. The results are summarized in Table 3 and Figure 6a.

As shown in Table 3, the cost-aware strategy consistently achieves the lowest runtime imbalance across all datasets, together with substantially smaller ΔTime and ΔMem, where ΔTime and ΔMem denote the gap between the maximum and minimum per-rank epoch time and peak memory, respectively, indicating more balanced workload distribution across devices. On larger datasets such as YAGO310 and Hetionet, both random and round-robin scheduling fail to finish, whereas the proposed strategy successfully completes training, demonstrating better robustness under heavy workloads.

Figure 6a further shows that CoRF achieves stable and near-linear speedup as the number of GPUs increases. Across datasets of different scales, the measured speedup closely follows the ideal linear trend, suggesting that the proposed scheduling strategy effectively mitigates straggler-induced

waiting and improves the efficiency of distributed training without introducing significant additional memory overhead.

Initial Receptive Support Size. We further study the sensitivity of CoRF to the initial receptive support size. As shown in Fig. 6b and Fig. 6c, smaller datasets such as CoDEx Small, WDsinger, and FB15k237 benefit more from larger supports, whereas on large-scale datasets such as CoDEx Large, DBpedia100k, and YAGO310, performance saturates quickly with relatively small supports.

This suggests that CoRF does not require proportionally larger receptive supports as graph size increases, which helps control structural overhead while maintaining stable reasoning performance. Furthermore, on extremely large knowledge graphs such as Wikidata5M [27], experiments show that using only about 2% of the original graph scale in receptive supports is sufficient for effective training.

5 Conclusion

In this paper, we address the problem of subgraph redundancy in large-scale knowledge graph reasoning and propose CoRF, which constructs a set of reusable receptive supports to explicitly reduce subgraph redundancy. Extensive experiments show that CoRF achieves competitive reasoning performance while significantly reducing time and memory costs, and also demonstrates strong scalability in distributed training. Overall, although existing approaches have improved the scalability of knowledge graph reasoning, repeated loading of highly overlapping local structures still incurs substantial time and GPU memory overhead. CoRF provides a simple and effective solution by enabling the reuse of shared receptive fields across queries.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China under Grant 62302131, the Postdoctoral Fellowship Program (Grade B) of the China Postdoctoral Science Foundation under Grant GZB20250405, the Fellowship Program (Grade C) of the China Postdoctoral Science Foundation under Grant GZC20251050, and the China Postdoctoral Science Foundation under Grant 2025M781526. The authors declare no competing interests.

References

[1] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2022.

[2] Guanglin Niu, Bo Li, and Yangguang Lin. A survey of task-oriented knowledge graph reasoning: Status, applications, and prospects. arXiv preprint arXiv:2506.11012, 2025.

[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

[4] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, pages 1649–1666, 2019.

[5] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*, pages 4708–4725, 2018.

[6] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. DRUM: End-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 15321–15331, 2019.

[7] Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, and Jian Tang. RNNLogic: Learning logic rules for reasoning on knowledge graphs. In *International Conference on Learning Representations*, pages 6203–6222, 2021.

[8] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural Bellman-Ford networks: A general graph neural network framework for link prediction. In *Advances in Neural Information Processing Systems*, pages 29476–29490, 2021.

[9] Yongqi Zhang and Quanming Yao. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM Web Conference 2022*, pages 912–924, 2022.

[10] Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *International Conference on Learning Representations*, pages 31598–31619, 2024.

[11] Jin Li, Zezhong Ding, and Xike Xie. DuetGraph: Coarse-to-fine knowledge graph reasoning with dual-pathway global-local fusion. In *Advances in Neural Information Processing Systems*, pages 46077–46108, 2025.

[12] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, 2008.

[13] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[15] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[16] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*, pages 3104–3118, 2018.

[17] Taraneh Younesian, Daniel Daza, Emile van Krieken, Thiviyan Thanapalasingam, and Peter Bloem. GRAPES: Learning to sample graphs for scalable graph neural networks. *Transactions on Machine Learning Research*, 2025.

[18] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. In *International Conference on Learning Representations*, pages 9259–9288, 2020.

[19] Yongqi Zhang, Zhanke Zhou, Quanming Yao, Xiaowen Chu, and Bo Han. AdaProp: Learning adaptive propagation for graph neural network based knowledge graph reasoning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3446–3457, 2023.

[20] Zhaocheng Zhu, Xinyu Yuan, Michael Galkin, Louis-Pascal Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. A*Net: A scalable path-based reasoning approach for knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 59323–59336, 2023.

[21] Komal K. Teru, Etienne G. Denis, and William L. Hamilton. Inductive relation prediction by subgraph reasoning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9448–9457, 2020.

[22] Shuangjia Zheng, Sijie Mai, Ya Sun, Haifeng Hu, and Yuedong Yang. Subgraph-aware few-shot inductive link prediction via meta-learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):6512–6517, 2023.

[23] Zhiwen Xie, Yi Zhang, Guangyou Zhou, Jin Liu, Xinhui Tu, and Jimmy Xiangji Huang. One subgraph for all: Efficient reasoning on opening subgraphs for inductive knowledge graph completion. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):8914–8927, 2024.

[24] (Andrew) Zhanke Zhou, Yongqi Zhang, Jiangchao Yao, Quanming Yao, and Bo Han. Less is more: One-shot subgraph reasoning on large-scale knowledge graphs. In *International Conference on Learning Representations*, pages 44980–45011, 2024.

[25] Yingxia Shao, Hongzheng Li, Xizhi Gu, Hongbo Yin, Yawen Li, Xupeng Miao, Wentao Zhang, Bin Cui, and Lei Chen. Distributed graph neural network training: A survey. *ACM Computing Surveys*, 56(8):1–39, 2024.

[26] Claire Songhyun Lee, V. Hewes, Giuseppe Cerati, Kewei Wang, Adam Aurisano, Ankit Agrawal, Alok Choudhary, and Wei-Keng Liao. Addressing GPU memory limitations for graph neural networks in high-energy physics applications. *Frontiers in High Performance Computing*, 2:1458674, 2024.

[27] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

[28] Tara Safavi and Danai Koutra. CoDEx: A comprehensive knowledge graph completion benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 8328–8350, 2020.

[29] Yixin Cao, Xiang Ji, Xin Lv, Juanzi Li, Yonggang Wen, and Hanwang Zhang. Are missing links predictable? An inferential benchmark for knowledge graph completion. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 6855–6865, 2021.

[30] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, 2015.

[31] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1811–1818, 2018.

[32] Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[33] Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L. Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E. Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *eLife*, 6:e26726, 2017.

[34] Wenhan Xiong, Thien Hoang, and William Yang Wang. DeepPath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 564–573, 2017.

[35] Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2925–2933, 2020.

[36] Boyang Ding, Quan Wang, Bin Wang, and Li Guo. Improving knowledge graph embedding using simple constraints. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 110–121, 2018.

[37] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In *Proceedings of the 7th Biennial Conference on Innovative Data Systems Research*, pages 1–11, 2015.

[38] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, pages 483–485, 1967.

Appendix

A Main Notations

To facilitate a rigorous and consistent presentation of our framework, the primary notations and their mathematical definitions are summarized in Table 4.

Table 4: Main notations used in this paper.

Notation	Definition
$G = (V, E, R)$	Knowledge graph with entity set V, triple set E, and relation set R.
$q = (h, r, t)$	Labeled training query with head entity h, relation r, and observed target t.
$\mathcal{Q}$	Set of labeled training queries.
$\mathrm{Ev}_k(q)$	k-hop local evidence set of query q.
S	A receptive support, i.e., a reusable local structure.
$\mathbb{S}_0$	Initial set of receptive supports before boundary enhancement.
$\mathbb{S}^+$	Final set of receptive supports.
$\mathcal{Q}^*(S)$	Set of queries covered by receptive support S.
$\mathcal{H}(q)$	Set of receptive supports that cover query q.
S_{res}	Residual support for queries not covered by existing supports.
$\mathcal{W}$	Work unit consisting of a receptive support and its covered queries.
Π_k	Set of work units assigned to device k.
θ_c	Hyperparameters related to receptive support construction.
θ_t	Hyperparameters related to training and system execution.

B Additional Visualization of Subgraph Overlap

To further investigate the structural redundancy introduced by subgraph-wise sampling methods, this appendix provides additional analyses from four complementary perspectives: global redundancy statistics, node-level overlap visualization, pairwise overlap analysis, and redundancy growth analysis.

First, to quantify the structural redundancy among subgraphs, we measure node redundancy and edge redundancy under different subgraph extraction strategies across multiple benchmark datasets. For a set of subgraphs $\{G_i = (V_i, E_i)\}$, the node redundancy ratio and edge redundancy ratio are defined as

$$\mathrm{NodeRedundancy} = 1 - \frac{|\bigcup_i V_i|}{\sum_i |V_i|}, \qquad \mathrm{EdgeRedundancy} = 1 - \frac{|\bigcup_i E_i|}{\sum_i |E_i|}. \tag{10}$$

Here, the denominators represent the cumulative structural size processed across all subgraphs, while the numerators denote the corresponding unique structural size after deduplication. These metrics quantify the proportion of repeatedly processed nodes and edges among all structures involved in computation.

Figure 7 presents the global redundancy statistics under different subgraph extraction strategies. The results reveal that significant node and edge redundancy persist across diverse extraction paradigms, suggesting that independent subgraph sampling inevitably incurs substantial structural overlap when processing multiple queries. This redundancy issue is further exacerbated as the graph scale and topological connectivity increase.

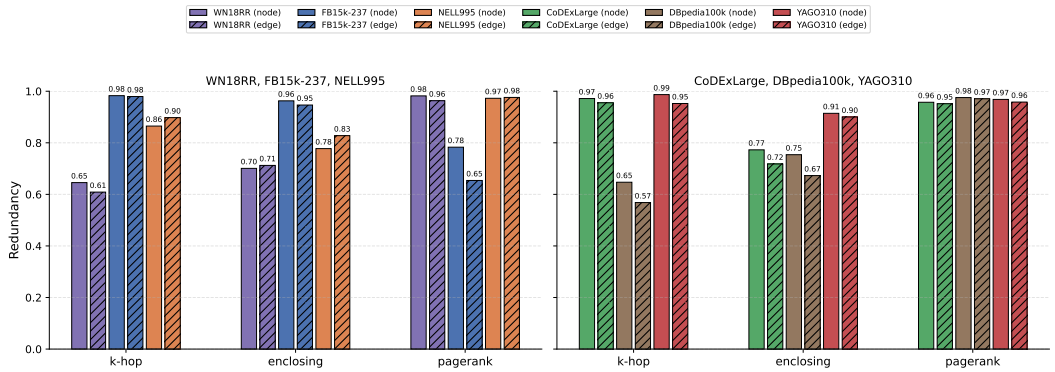


Figure 7: Node and edge redundancy under different subgraph extraction strategies.

B.1 Pairwise Overlap Analysis

We further examine the pairwise overlap between query-specific subgraphs. Figure 8 shows the distribution of Jaccard similarity on FB15k237.

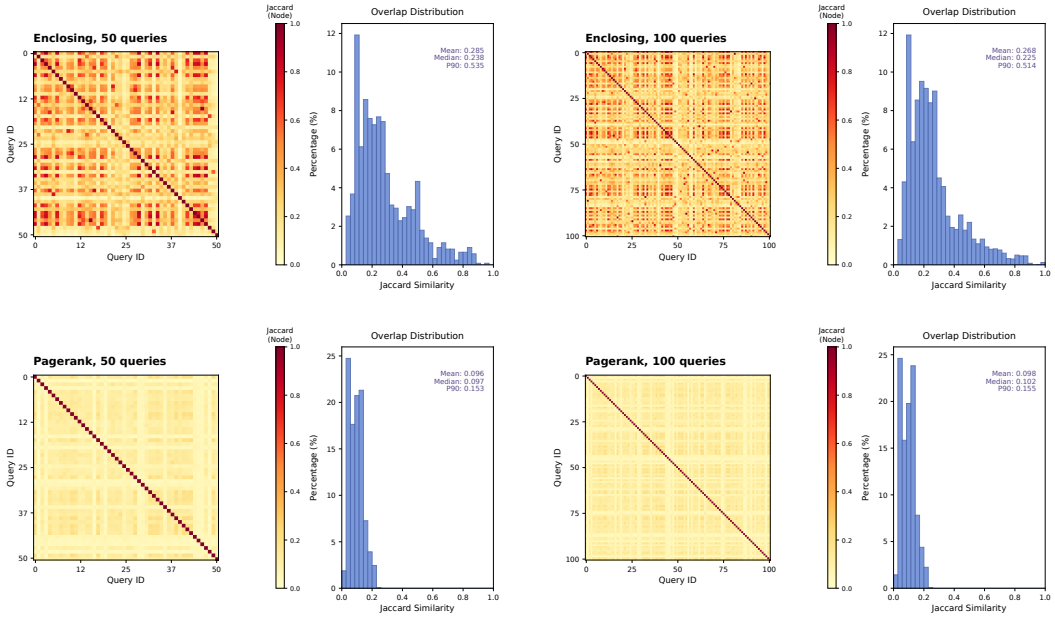


Figure 8: Pairwise overlap between sampled subgraphs on FB15k237.

The results demonstrate that non-trivial structural overlap is pervasive across disparate queries. Notably, even among a limited query set, a substantial proportion of query pairs exhibits high similarity, suggesting that multiple queries frequently anchor their reasoning on congruent local topological structures, and underscoring the commonality of local receptive fields among different reasoning tasks. A consistent pattern is observed on the WN18RR dataset, as illustrated in Figure 9.

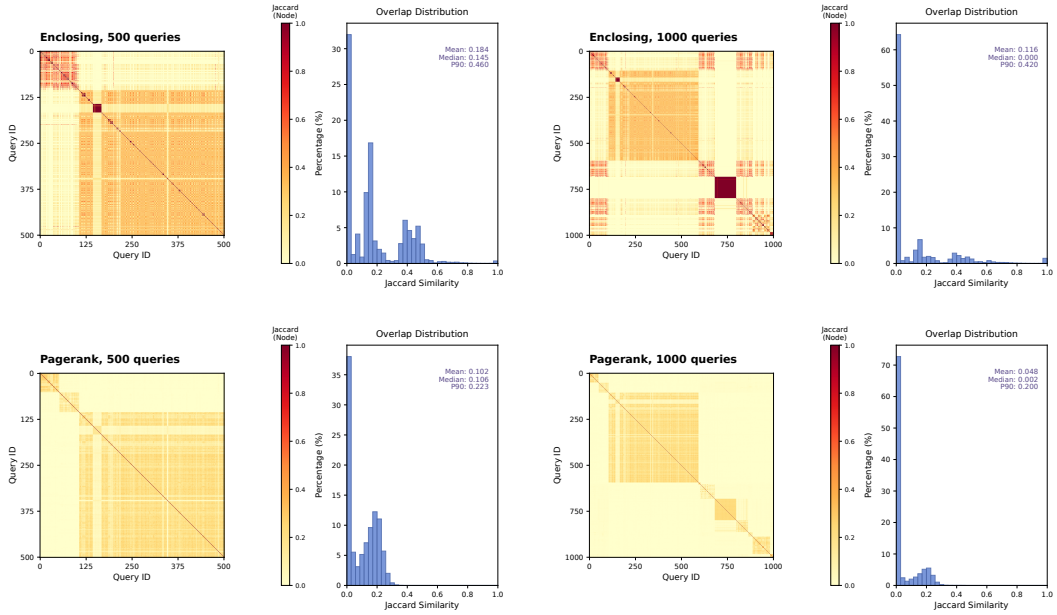


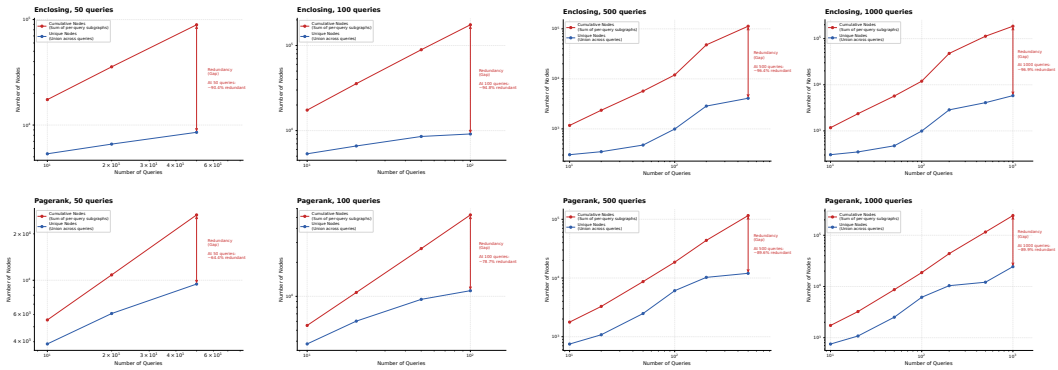
Figure 9: Pairwise overlap between sampled subgraphs on WN18RR with 500 and 1000 queries.

As the number of queries increases, the similarity distribution becomes more dispersed, yet regions of high overlap remain prominent. This indicates that structural sharing across queries is a persistent and widespread phenomenon, rather than an artifact of a small subset of queries.

B.2 Redundancy Growth Analysis

Finally, we analyze how cumulative and unique structural sizes evolve as the number of queries increases. Figure 10a shows the results on FB15k237. As observed, the cumulative number of nodes (i.e., the sum of per-query subgraph sizes) grows rapidly with the number of queries, while the number of unique nodes (i.e., the union of all subgraphs) increases much more slowly. The widening gap between these two curves indicates that an increasing fraction of computation is spent on repeatedly accessing the same structures, leading to substantial redundancy.

A similar trend is observed on WN18RR. Figure 10b presents results under different query scales. Under larger query scales, the gap further enlarges, suggesting that structural redundancy becomes more pronounced in large-scale settings.



(a) FB15k-237: 50 and 100 queries.

(b) WN18RR: 500 and 1000 queries.

Figure 10: Growth of cumulative and unique structures on FB15k-237 and WN18RR.

Algorithm 1 Initial Receptive Support Construction

Require: Knowledge graph $\mathcal{G} = (V, E, R)$, partitioning function $f(\cdot)$, core ratio r_c, support size ratio r_s.
Ensure: Initial receptive supports $\mathcal{S}_0$.
1: Set the maximum partition size $B \leftarrow r_s|V|$.
2: Set the number of core entities $N_c \leftarrow r_c|V|$.
3: Select the N_c entities with the highest topology importance scores $\rho(v)$ as the shared core set $\mathcal{C}$.
4: Partition the graph under the size constraint B: $\{P_1, \ldots, P_K\} \leftarrow f(\mathcal{G}, B)$.
5: **for** $i = 1, \ldots, K$ **do**
6: Construct $S_i \leftarrow P_i \cup \mathcal{C}$.
7: **end for**
8: $\mathcal{S}_0 \leftarrow \{S_1, \ldots, S_K\}$.
9: **return** $\mathcal{S}_0$.

Overall, these visualizations consistently demonstrate that substantial and increasingly pronounced structural overlap exists across queries in subgraph-wise sampling methods. Consequently, mitigating such redundancy is a key factor for improving the scalability of large-scale knowledge graph reasoning.

C Structural Properties and Preprocessing Efficiency of CoRF

In this appendix, we provide further details on receptive support construction, analyze the structural properties of the resulting supports, and report the preprocessing efficiency of CoRF.

C.1 Receptive Support Construction and Properties

Given a knowledge graph $\mathcal{G} = (V, E, R)$, a graph partitioning function $f(\cdot)$, a core ratio r_c, and a support size ratio r_s, CoRF first selects a small set of topologically important entities as the shared core. The graph is then partitioned under the support size constraint, and each partition is augmented with the shared core to construct the initial receptive supports $\mathcal{S}_0 = \{S_1, \ldots, S_K\}$. The detailed construction procedure is summarized in Algorithm 1.

Furthermore, after Initial Partitioning, Boundary Padding, and Residual Compensation, the final receptive support set $\mathbb{S}^+$ satisfies the following structural properties.

Coverage and bounded size. Each query is ultimately covered by a receptive support (or a residual support), ensuring that no training query is omitted. Meanwhile, the size of each support is explicitly bounded, preventing it from growing to an almost full-graph scale.

Locality and controlled overlap. Each receptive support preserves local connectivity, while limited node overlap is allowed across different supports in a controlled manner, enabling important nodes to appear in multiple supports.

Monotonic boundary augmentation. Through the boundary augmentation mechanism, a receptive support expands from S to $\tilde{S}$, satisfying

$$S \subseteq \tilde{S}, \quad E(S) \subseteq E(\tilde{S}).$$

This process only augments the structure without altering the original local topology, thereby improving the completeness of multi-hop paths near the support boundary.

Non-degeneration to subgraph-wise sampling. Residual Compensation does not construct an individual support for each uncovered query. Instead, as defined in Eq. (5), it constructs a unified residual support:

$$S_{\text{res}} = \bigcup_{q:\mathcal{H}(q)=\emptyset} \text{Ev}_k(q),$$

Table 5: Statistics of uncovered training queries on large-scale knowledge graphs.

Dataset	Support Size	Queries	Uncovered	Ratio
YAGO310	10K	1.08M	10.75K	1.0%
Hetionet	5K	2.03M	27.15K	1.3%
Wikidata5M	50K	20M	480K	2.4%

which covers all remaining uncovered queries. Therefore, CoRF does not degenerate into subgraph-wise sampling, where individual subgraphs are constructed for different queries. The number of uncovered queries is controlled by the receptive support size and Boundary Padding hyperparameters. Since receptive supports are constructed under topological completeness and coverage constraints, most queries can be covered by the initial supports, while the number of uncovered queries is minimized.

In our experiments, most datasets do not generate a large number of uncovered queries after Initial Partitioning and Boundary Padding. Only a few large-scale datasets, such as YAGO310 and Hetionet, produce a small number of uncovered queries. We further validate CoRF on the larger-scale Wikidata5M dataset, which contains around 5M entities and 20M triples. As shown in Table 5, although the absolute number of uncovered queries increases with graph scale, the ratio remains controlled under practical settings.

Overall, these properties ensure sufficient structural information for knowledge graph reasoning while maintaining controllable support size. CoRF alleviates memory bottlenecks on large-scale datasets.

C.2 Comparison with Scalable Training Strategies

For GNN-based knowledge graph reasoning, existing graph sampling-based scalable training strategies can be broadly categorized into three groups: node-wise sampling, layer-wise sampling, and subgraph-wise sampling.

Node-wise sampling methods. Node-wise sampling methods sample a fixed-size set of neighbors for each node at each message passing layer. This helps mitigate neighborhood explosion and improves scalability on large-scale graphs. However, because the number of sampled nodes still grows with the depth of message passing, and sampling may omit part of the graph structure, their effectiveness can be limited in knowledge graph reasoning.

Layer-wise sampling methods. Layer-wise sampling methods mitigate neighborhood explosion by sampling a fixed-size subset of neighbors at each layer. This helps control memory complexity and improves the scalability of GNN training on large-scale graphs. However, independent layer-wise sampling may omit some query relevant structural information, which can affect reasoning performance.

Subgraph-wise sampling methods. Subgraph-wise sampling methods perform message passing on sampled subgraphs rather than on the full graph, which reduces the graph scale involved in computation and improves scalability on large-scale graphs. However, they usually construct a subgraph for each query. Since different queries often rely on highly overlapping local evidence, these query-specific subgraphs may contain substantial structural overlap. Repeatedly extracting, loading, and processing such subgraphs introduces significant structural redundancy, which in turn limits scalability.

In contrast, CoRF is based on a reusable set of receptive supports. Each receptive support serves as a reusable local structure that can cover the required evidence for multiple queries. By reusing supports across queries, CoRF reduces the repeated extraction and loading of overlapping local structures, thereby reducing structural redundancy and improving scalability.

C.3 Preprocessing Efficiency

Subgraph-wise sampling methods typically construct or materialize a query-specific subgraph for each query during preprocessing. As a result, both preprocessing time and intermediate storage scale rapidly with the number of queries and the size of the graph, making such approaches difficult to apply to large-scale knowledge graphs.

To quantify this overhead, we compare a representative subgraph-wise sampling method (One-shot [24]) with CoRF in terms of preprocessing time and intermediate storage across multiple datasets.

Table 6: Preprocessing time and storage comparison between One-shot and CoRF.

Dataset	One-shot		CoRF		Dataset	One-shot		CoRF	
	Time	Storage	Time	Storage		Time	Storage	Time	Storage
CoDEx Small	130 s	85 MB	0.94 s	11 MB	**WDsinger**	330 s	2.4 GB	0.53 s	6.6 MB
FB15k237	7,099 s	2.5 GB	6.22 s	67 MB	**CoDEx Medium**	26,379 s (7.33 h)	3.3 GB	4.75 s	66 MB
NELL23k	977 s	5.8 GB	0.79 s	12 MB	**WN18RR**	9,245 s	20 GB	2.08 s	26 MB
AristoV4	22,958 s (6.38 h)	23 GB	5.98 s	234 MB	**Hetionet**	205,321 s (57.03 h)	23 GB	70.38 s	1.5 GB
NELL995	32,268 s (8.96 h)	64 GB	3.96 s	101 MB	**CoDEx Large**	119,301 s (33.14 h)	70 GB	13.51 s	388 MB
ConceptNet100k	26,089 s (7.25 h)	71 GB	2.54 s	45 MB	**DBpedia100k**	153,036 s (42.51 h)	118 GB	14.41 s	188 MB
YAGO310	289,404 s (80.39 h)	184 GB	24.11 s	200 MB					

Table 6 reports the preprocessing costs of One-shot and CoRF. As shown, CoRF consistently achieves substantial reductions in both preprocessing time and storage overhead across all datasets. For example, on YAGO310, One-shot requires 289,404 seconds (approximately 80.39 hours) and 184 GB of intermediate storage, whereas CoRF completes preprocessing in only 24.11 seconds with 200 MB of storage. Similarly, on DBpedia100k, One-shot takes 153,036 seconds (approximately 42.51 hours) and 118 GB, while CoRF requires only 14.41 seconds and 188 MB.

Overall, CoRF provides a significantly more lightweight preprocessing pipeline, reducing both time and storage by orders of magnitude. This advantage becomes increasingly pronounced as the dataset size grows, suggesting that avoiding per-query subgraph construction is critical for scalable preprocessing and makes CoRF particularly suitable for large-scale knowledge graph training.

D Detailed Dataset Information

To evaluate the robustness and scalability of the proposed framework across diverse knowledge domains, we conduct extensive experiments on 13 benchmark datasets. These datasets represent a wide spectrum of graph scales (Small, Medium, and Large) and structural characteristics, ranging from linguistic networks (e.g., WN18RR) to large-scale general knowledge bases (e.g., YAGO310). The detailed topological statistics are provided below, including the counts of entities and relations, as well as the specific data splits for training, validation, and testing.

Table 7: Statistics of transductive datasets used in this work. The scale labels are consistent with the dataset grouping used in the main text.

Dataset	Entities	Rels	Train	Valid	Test	Task	Reference	Scale
CoDEx Small	2034	42	32888	1827	1828	h/t	[28]	S
WDsinger	10282	135	16142	2163	2203	h/t	[29]	S
FB15k237	14541	237	272115	17535	20466	h/t	[30]	M
CoDEx Medium	17050	51	185584	10310	10311	h/t	[28]	M
NELL23k	22925	200	25445	4961	4952	h/t	[29]	S
WN18RR	40943	11	86835	3034	3134	h/t	[31]	M
AristoV4	44949	1605	242567	20000	20000	h/t	[32]	M
Hetionet	45158	24	2025177	112510	112510	h/t	[33]	L
NELL995	74536	200	149678	543	2818	h/t	[34]	M
CoDEx Large	77951	69	551193	30622	30622	h/t	[28]	L
ConceptNet100k	78334	34	100000	1200	1200	h/t	[35]	L
DBpedia100k	99604	470	597572	50000	50000	h/t	[36]	L
YAGO310	123182	37	1079040	5000	5000	h/t	[37]	L

E Evaluation Metrics

Metrics. We evaluate all methods from two perspectives: reasoning accuracy and system efficiency. For reasoning accuracy, we adopt the widely used metrics in knowledge graph reasoning, including MRR and Hits@K [3].

For system efficiency, we report both time and resource consumption during training, including the training time per epoch (Seconds per Epoch, S/E) and the peak CUDA reserved memory (GB). S/E measures training efficiency, while peak reserved memory reflects the memory footprint and hardware demand during training. Compared to allocated memory, reserved memory provides a more reliable estimate of actual GPU usage and the risk of out-of-memory (OOM) errors. Detailed definitions are provided below.

This section defines the evaluation metrics used in this paper, including both reasoning effectiveness and system efficiency metrics. For reasoning performance, we adopt the standard metrics **Mean Reciprocal Rank (MRR)** and **Hits@k**, which are defined as follows:

$$\text{Hits@k} = \frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{t_i \in \mathcal{T}_{\text{test}}} f(\text{rank}_i), \quad f(x) = \begin{cases} 1, & x \le k \\ 0, & x > k \end{cases} \tag{11}$$

$$\text{MRR} = \frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{t_i \in \mathcal{T}_{\text{test}}} \frac{1}{\text{rank}_i}. \tag{12}$$

To facilitate comparison across methods, we further report relative efficiency metrics. For runtime, we use the speedup ratio:

$$\text{Speedup} = \frac{T_{\text{baseline}}}{T_{\text{method}}}, \tag{13}$$

where T_{baseline} and T_{method} denote the runtime of the baseline and the compared method under the same setting, respectively. Depending on the experimental scenario, T may refer to the training time per epoch or the inference time, corresponding to training speedup and inference speedup.

For memory usage, we report the memory saving ratio:

$$\text{Memory Saving} = \frac{M_{\text{baseline}}}{M_{\text{method}}}, \tag{14}$$

where M_{baseline} and M_{method} denote the peak memory consumption of the baseline and the compared method, respectively. A larger value indicates greater memory savings. If a method cannot run on a single NVIDIA L40 (48GB) GPU, it is marked as **OOM**.

For distributed training, we adopt strong scaling speedup [38] to measure multi-GPU efficiency:

$$\text{Strong Scaling Speedup} = \frac{T_{1\text{GPU}}}{T_{n\text{GPU}}}, \tag{15}$$

where $T_{1\text{GPU}}$ and $T_{n\text{GPU}}$ denote the time required to complete the same training task on one GPU and n GPUs, respectively.

F Hierarchical Hyperparameter Search

CoRF involves two tightly coupled components: coupled receptive fields construction and model training, each characterized by a distinct set of hyperparameters. Fully manual tuning is not only time-consuming but also makes it difficult to jointly optimize reasoning performance, training efficiency, and resource usage. Therefore, we design an automated hyperparameter search pipeline to systematically explore the configuration space.

Let $\theta_c \in \Theta_c$ denote the parameters for coupled receptive fields construction, and $\theta_t \in \Theta_t$ denote the model parameters. The performance of CoRF is jointly influenced by structural parameters and training parameters. Due to the dependency between these two groups of parameters, directly searching over the joint space $\Theta_c \times \Theta_t$ is computationally expensive and often results in evaluating

many low-quality configurations. As illustrated in Fig. 11, we therefore adopt a hierarchical low-fidelity search strategy. Given a validation metric $\mathrm{Score}(\cdot)$, the overall objective of the model is

$$\max_{\theta_c \in \Theta_c,\, \theta_t \in \Theta_t} \mathrm{Score}(\theta_c, \theta_t). \tag{16}$$

To mitigate the prohibitive complexity of the joint search space $\Theta_c \times \Theta_t$, we decouple the optimization process into a sequential conditional refinement strategy. This approach exploits the inherent dependency between structural topology and training dynamics by decomposing the joint problem into two tractable sub-tasks. We approximately optimize this objective through the following two-stage search.

$$\theta_c^* = \arg\max_{\theta_c} \widehat{\mathrm{Score}}_B(\theta_c, \bar{\theta}_t), \quad \theta_t^* = \arg\max_{\theta_t} \widehat{\mathrm{Score}}_B(\theta_c^*, \theta_t), \tag{17}$$

where $\bar{\theta}_t$ represents a representative baseline training configuration, and $\widehat{\mathrm{Score}}_B(\cdot)$ denotes the performance proxy obtained after training for a maximum of B epochs.

Stage I: Structure Search. In the first stage, we fix the default training configuration $\bar{\theta}_t$ and search over structural parameters. This stage focuses on the structural quality and coverage of receptive supports. For key parameters that control the size of supports, we use grid search. For combinatorial structural parameters, we adopt random sampling to balance search quality and efficiency.

Stage II: Training Search. In the second stage, we fix the optimal structural configuration θ_c^* and search over training parameters, such as hidden dimension, number of layers, batch size, and learning rate. With the structure fixed, the search space is significantly reduced, enabling more efficient optimization of both model performance and training efficiency.

To further mitigate computational overhead, we implement a low-fidelity evaluation mechanism integrated with early-stopping heuristics. Each candidate configuration is restricted to a maximum training budget of B epochs, using validation metrics as the feedback signal for convergence monitoring. Specifically, a trial is prematurely terminated if the validation performance exhibits stagnation for P consecutive evaluation intervals. Furthermore, to ensure execution isolation and idempotency, each trial is allocated an independent cache indexed by a unique identifier $\mathrm{id}(\theta) = \mathrm{hash}(\theta)$, thereby precluding state interference across disparate configurations.

Finally, we return the configuration (θ_c^*, θ_t^*) that achieves the best validation performance. Overall, the proposed hierarchical strategy leverages the dependency structure of the parameter space to efficiently identify high-quality configurations under a limited search budget.

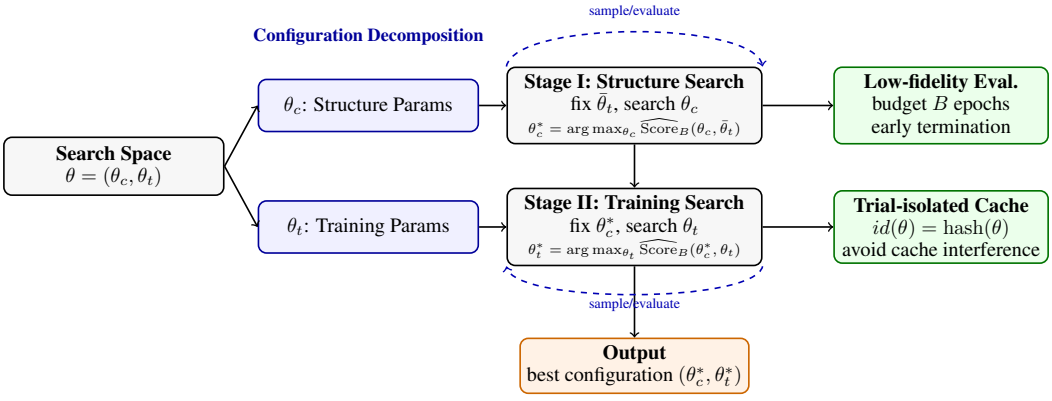


Figure 11: Hierarchical multi-fidelity hyperparameter search framework for CoRF.

Effectiveness of the hyperparameter search strategy. As shown in Figure 12, we compare the performance of Search init and Random init on 13 transductive datasets in terms of MRR and Hits@10. Specifically, we compare the configurations obtained by hierarchical search (Search init)

with those randomly sampled from the same search space (Random init), to evaluate whether the proposed strategy can consistently identify better configurations under a limited search budget. Search init consistently outperforms Random init across all datasets. It indicates that hierarchical search can identify more effective hyperparameter configurations under a limited search budget.

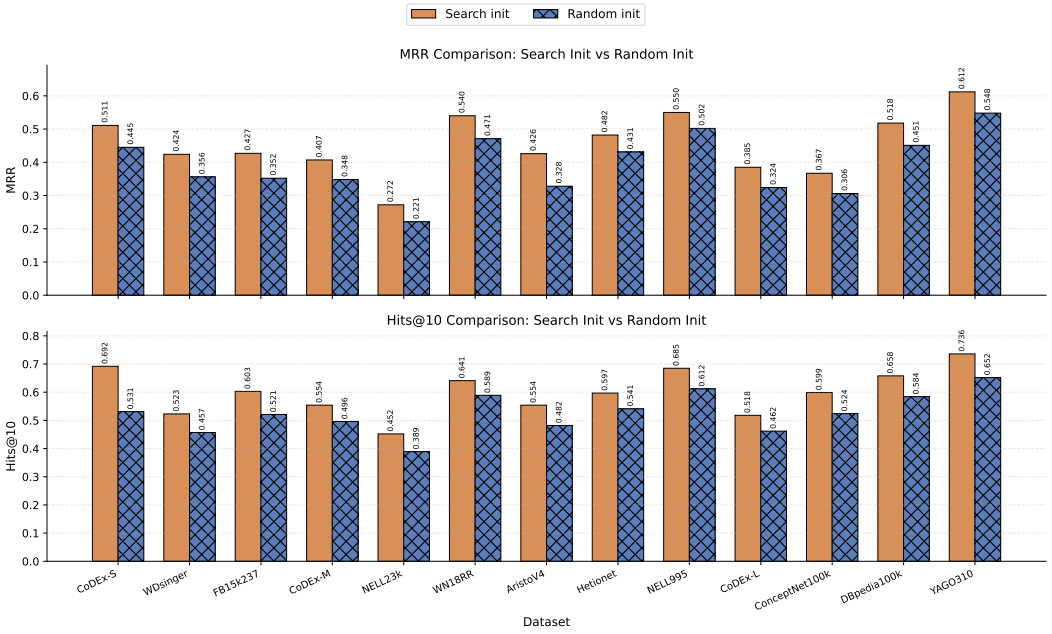


Figure 12: Performance comparison between hierarchical search initialization and random configuration sampling on 13 transductive datasets.

G Additional Results Discussion

G.1 Accuracy-Efficiency Trade-offs on More Transductive Datasets

To provide a more complete view of the accuracy–efficiency trade-offs, we present results on more transductive datasets in Figures 13 and 14. These results further show that CoRF achieves favorable trade-offs between reasoning performance and computational efficiency across datasets of different scales.

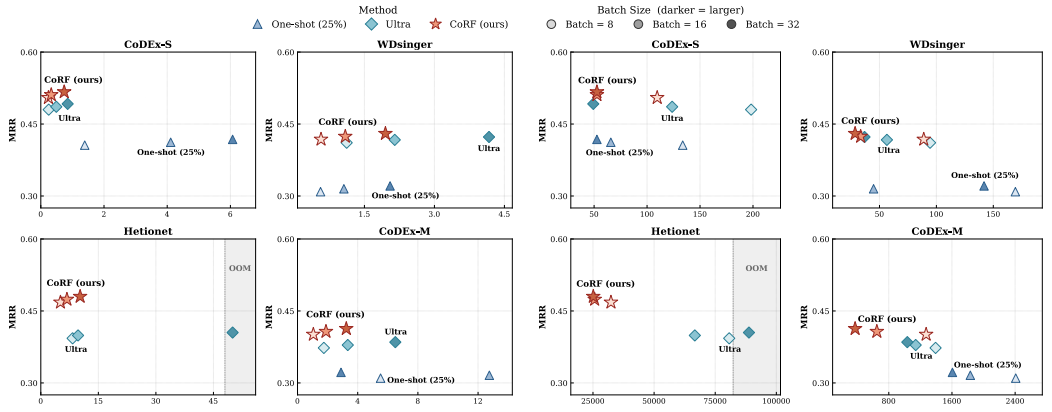


Figure 13: Accuracy–efficiency trade-offs of different methods on more transductive datasets. Upper-left is better.

Table 8: Comparison with A*Net and NBFNet under the transductive setting.

Dataset	CoRF			A*Net			NBFNet		
	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)
CoDEx Large	**0.385**	**2652.29**	**9.73**	–	–	OOM	–	–	OOM
DBpedia100k	**0.518**	**3607.66**	**12.05**	–	–	OOM	–	–	OOM
YAGO310	**0.612**	**10066.00**	**16.05**	–	–	OOM	–	–	OOM
Hetionet	**0.482**	**25121.37**	**10.25**	–	–	OOM	–	–	OOM
ConceptNet100k	**0.367**	**670.79**	**16.22**	0.309	692.03	21.69	–	–	OOM
FB15k237	**0.427**	**759.00**	**4.28**	0.411	2359.38	22.56	0.415	6297.77	20.45
CoDEx Medium	**0.407**	**373.53**	**3.24**	0.361	1711.87	22.86	0.375	5292.82	23.62
WN18RR	0.540	**287.00**	**7.45**	**0.552**	464.60	20.42	–	–	OOM
AristoV4	**0.426**	11029.68	**9.27**	0.274	**2036.09**	22.91	–	–	OOM
NELL995	**0.550**	955.00	14.65	0.193	**171.63**	**11.53**	0.297	1188.26	43.64
CoDEx Small	**0.511**	**52.66**	**0.74**	0.454	136.98	3.41	0.483	130.30	2.88
WDSinger	**0.424**	**28.57**	**1.95**	0.331	95.44	6.72	0.412	143.83	13.32

Table 9: Comparison with AdaProp and RED-GNN under the transductive setting.

Dataset	CoRF			AdaProp			RED-GNN		
	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)
CoDEx Large	**0.385**	**2652.29**	**9.73**	–	–	OOM	–	–	OOM
DBpedia100k	**0.518**	**3607.66**	**12.05**	0.465	19327.65	43.52	–	–	OOM
YAGO310	**0.612**	**10066.00**	**16.05**	–	–	OOM	–	–	OOM
Hetionet	**0.482**	**25121.37**	**10.25**	–	–	OOM	–	–	OOM
ConceptNet100k	**0.367**	**670.79**	**16.22**	0.349	976.47	21.87	–	–	OOM
FB15k237	**0.427**	**759.00**	**4.28**	0.401	11883.09	43.04	–	–	OOM
CoDEx Medium	**0.407**	**373.53**	**3.24**	0.384	5731.07	43.35	–	–	OOM
WN18RR	0.540	287.00	**7.45**	0.521	242.01	12.57	**0.543**	**198.17**	9.93
AristoV4	**0.426**	**11029.68**	**9.27**	–	–	OOM	–	–	OOM
NELL995	**0.550**	**955.00**	**14.65**	0.544	3005.87	41.71	–	–	OOM
CoDEx Small	**0.511**	**52.66**	**0.74**	0.484	341.26	17.13	0.493	343.17	16.61
WDSinger	**0.424**	**28.57**	**1.95**	0.403	76.03	12.43	0.415	160.76	21.92

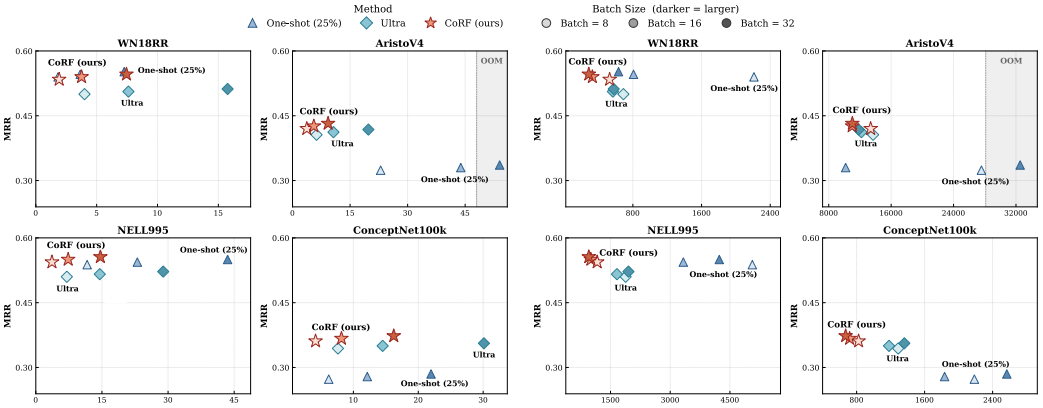


Figure 14: Accuracy–efficiency trade-offs of different methods on more transductive datasets. Upper-left is better.

G.2 CoRF vs. More CMP-based Models in the Transductive Setting

To provide a more extensive comparison with CMP-based methods, we further evaluate CoRF against NBFNet, RED-GNN, A*Net, and AdaProp under the transductive setting. All experiments use a batch size of 32, 6 message-passing layers, and a hidden dimension of 64. The results are reported in Tables 8 and 9.

The results further demonstrate the scalability advantages of CoRF over existing CMP-based methods. On medium- and large-scale datasets, NBFNet, RED-GNN, A*Net, and AdaProp encounter OOM under the same 48GB NVIDIA L40 GPU setting, while CoRF remains trainable. Specifi-

Table 10: Inductive comparison with A*Net, AdaProp, and RED-GNN.

Dataset	CoRF			A*Net			AdaProp			RED-GNN		
	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)
FB V1	**0.501**	2.15	**1.15**	0.446	7.75	1.98	0.316	**1.87**	2.72	0.335	3.92	3.07
FB V2	**0.525**	**7.67**	1.66	0.510	17.44	3.14	0.457	8.78	**0.436**	0.424	13.42	6.78
FB V3	**0.503**	**14.13**	**2.14**	0.476	35.70	4.74	0.464	51.35	3.83	0.384	52.51	14.19
FB V4	**0.499**	**23.78**	**2.65**	0.466	60.08	7.20	0.440	92.75	4.01	0.414	110.42	20.72
NELL V1	0.680	**3.75**	1.65	0.731	8.63	**0.369**	**0.741**	4.66	1.09	0.599	8.38	2.59
NELL V2	**0.481**	6.11	1.17	0.410	14.42	**0.65**	0.438	**3.69**	0.85	0.395	18.22	5.99
NELL V3	**0.515**	**14.12**	1.99	0.481	27.14	**1.20**	0.423	49.35	1.59	0.299	51.24	12.33
NELL V4	**0.495**	6.19	1.07	0.237	12.05	**0.66**	0.348	**3.48**	1.23	0.212	15.39	8.03

Table 11: Inductive comparison with NBFNet and Ultra.

Dataset	CoRF			NBFNet			Ultra		
	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)	MRR	S/E (s)	M (GB)
FB V1	0.501	**2.15**	**1.15**	0.422	7.15	4.57	**0.509**	3.76	1.60
FB V2	**0.525**	**7.67**	**1.66**	0.514	24.47	7.41	0.524	8.05	2.36
FB V3	0.503	**14.13**	**2.14**	0.476	59.93	10.41	**0.504**	16.98	3.34
FB V4	**0.499**	**23.78**	**2.65**	0.453	116.51	13.08	0.496	31.40	4.14
NELL V1	0.680	**3.75**	**1.65**	0.742	13.58	8.60	**0.757**	3.88	2.48
NELL V2	0.481	6.11	**1.17**	0.373	19.03	7.15	**0.575**	**5.96**	2.15
NELL V3	0.515	**14.12**	**1.99**	0.491	66.78	12.92	**0.563**	16.77	3.81
NELL V4	**0.495**	**6.19**	**1.07**	0.226	15.60	5.85	0.469	6.42	1.80

cally, compared with NBFNet, CoRF achieves an average $6.2\times$ speedup and reduces GPU memory consumption by approximately $5.2\times$. Compared with AdaProp, CoRF achieves an average $6.4\times$ speedup and reduces GPU memory consumption by approximately $7.8\times$.

G.3 Inductive Generalization

Although CoRF primarily focuses on large-scale transductive KG reasoning, it is also compatible with the inductive setting. To further evaluate its inductive generalization ability, we conduct experiments against NBFNet, A*Net, RED-GNN, AdaProp, and Ultra. All experiments use a batch size of 64, 6 message-passing layers, and a hidden dimension of 64. The results are reported in Tables 10 and 11.

These results demonstrate that CoRF can effectively generalize beyond the transductive setting. CoRF maintains competitive reasoning performance on the inductive benchmarks while retaining favorable computational efficiency.

H Limitations and Broader Impacts

H.1 Limitations

Although CoRF achieves substantial improvements in computational efficiency, its benefits depend on the degree of structural overlap among query receptive fields. For knowledge graphs with many disconnected components, queries may share fewer local structures, limiting opportunities for receptive field reuse. Future work could explore more adaptive receptive field sharing strategies to further improve scalability across diverse graph structures.

H.2 Broader Impacts

CoRF is a scalable framework for large-scale knowledge graph reasoning and can support reasoning under larger graph settings. It also holds strong potential for accelerating real-world knowledge discovery in domains such as biomedicine, including drug repurposing and gene-disease association prediction. As with other knowledge graph reasoning methods, incorrect predictions or biased source data may lead to misleading downstream conclusions, especially in sensitive applications such as biomedicine.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize the main contributions of the paper, namely reducing subgraph redundancy in large-scale knowledge graph reasoning through reusable receptive supports, together with the resulting gains in efficiency, memory usage, and scalability. These claims are supported by the methodological description and experimental results in the main paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the main limitations of CoRF in the appendix. In particular, CoRF's efficiency gains depend on the degree of structural overlap among query receptive fields. For knowledge graphs with many disconnected components, reduced structural sharing opportunities may limit its scalability benefits.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [No]

   Justification: The paper contains some analytical claims, such as NP-hardness results, but does not include full formal proofs, since these follow from standard complexity arguments and the main contribution of the paper is methodological and empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the datasets, evaluation settings, model configurations needed to understand and reproduce the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code is being prepared for release. The project repository is available at https://github.com/bovane/CoRF. The paper describes the datasets, settings, and implementation details used in the experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the datasets, model settings, hyperparameter search strategy, batch sizes used in the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper does not report error bars or formal statistical significance tests. The main focus is on consistent efficiency and scalability comparisons across multiple datasets and settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the compute setting used in the experiments, including GPU-based training on NVIDIA RTX 4090 (24GB) and NVIDIA L40 (48GB), as well as execution time, memory usage, and distributed training analysis.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research presented in this paper conforms to the NeurIPS Code of Ethics. The work focuses on algorithmic and systems aspects of knowledge graph reasoning and does not involve ethically sensitive experimental procedures.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses the broader impacts of the proposed method, including its potential to support large-scale knowledge graph reasoning and accelerate real-world knowledge discovery, in the appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk generative models, scraped datasets, or other assets that would require special safeguards for responsible release.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper credits the original datasets and prior methods through citations, but it does not explicitly document the licenses and terms of use of all existing assets in detail.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce new public datasets or other new assets requiring separate documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as an important, original, or non-standard component of the core methodology in this research.

    Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.