HyperSearch: Prediction of New Hyperedges through Unconstrained yet Efficient Search - Supplementary Document

APPENDIX

A. Additional Details on Dataset Preprocessing (Supplementing Sect. VI-A)

The Cora, High, Primary, Ubuntu, and Math datasets contain only hyperedges of size up to 5. For the other five datasets, we retained only hyperedges with sizes up to 10 (before splitting them into E_1 and E_2) for computational efficiency. This preprocessing step was applied to all methods.

Moreover, we identified hyperedge sizes i where the proportion of hyperedges was below 1% and removed them. As shown in Table I, after such preprocessing, in each dataset, at least 90% of the original hyperedges remained.

B. Hyperparameter Search Spaces for HNN Models (Supplementing Sect. VI-A)

Table II summarizes the hyperparameter search spaces used for the HNN models employed in our experiments. The hyperparameters are selected based on the original settings proposed in their respective papers and further optimized based on validation performance. Common hyperparameters (e.g., epochs, hidden dimensions, optimizer, learning rate, batch size) are consistently aligned across methods, while method-specific parameters are indicated separately.

C. Additional Evaluation Metric: Average F1 Score (Supplementing Sect. VI-A and Sect.VI-B)

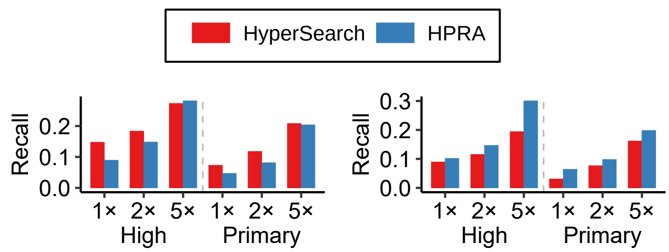
To supplement the Recall@$\mathcal{K}$ evaluation in the main article, we also report the results using the *Average F1 Score*, which quantifies the closeness between predicted and ground-truth hyperedges. Specifically, for each predicted hyperedge, we compute the F1 score against its best-matching ground-truth hyperedge, and vice versa, then average the two values across the dataset. This provides a complementary perspective that captures partial correctness, especially in cases where predicted and true hyperedges have significant but not complete overlap. Formally, the Average F1 Score is defined as:

$$\text{Average F1 Score} := \frac{1}{2}\left(\begin{array}{c}\frac{1}{|E_2|}\sum\limits_{e_i \in E_2} F1(e_i, e'_{g(i)}) \\ +\frac{1}{|E'|}\sum\limits_{e'_i \in E'} F1(e_{g'(i)}, e'_i)\end{array}\right) \quad (1)$$

where:

- E_2 is the set of missing hyperedges (test set),
- E' is the set of predicted hyperedges,
- $g(i)$ returns the index of the best-matching predicted hyperedge e'_j for each e_i, i.e., $g(i) = \arg\max_j \mathrm{F1}(e_i, e'_j)$,
- $g'(i)$ returns the index of the best-matching ground-truth hyperedge e_j for each e'_i, i.e., $g'(i) = \arg\max_j \mathrm{F1}(e_j, e'_i)$.

Tables III and IV summarize the results. Compared to Recall@$\mathcal{K}$, which focuses on exact matching, Average F1 Score highlights subtler differences in hyperedge structure and matching behavior.



(a) All Candidates (b) Size-3 Candidates

Fig. 1. (a) The superiority of HyperSearch in predictive accuracy diminishes as the number of predicted hyperedges increases. (b) HyperSearch is outperformed by HPRA on size-3 hyperedges, likely due to the lack of promising candidates based on our scoring function. This highlights a limitation of HyperSearch in identifying *structurally novel* hyperedges composed of node combinations absent in observed hyperedges.

D. Additional Results On “Q1. Accuracy” (Supplementing Sect.VI-B)

As shown in Fig. 1(a), the superiority of HyperSearch in predictive accuracy diminishes as the number of predicted hyperedges increases. As depicted in Fig. 1(b), HyperSearch is outperformed by HPRA w.r.t. the performance on size-3 hyperedges, this is likely because there are not many size-3 promising candidates based on our scoring function, which reveals a potential limitation of HyperSearch in identifying “structurally novel” hyperedges consisting of node combinations that do not exist in observed hyperedges.

E. Refinement with HNNs (Supplementing Sects. VI-A & VI-B)

To further refine and enhance the quality of the predicted hyperedges, we incorporate a filtering step using Hypergraph Neural Networks (HNNs). This step applies an HNN model to the hyperedge candidates generated in the earlier stages, leveraging learned representations to improve prediction performance. Specifically, it involves the following steps:

TABLE I
PROPORTION OF HYPEREDGES BASED ON THEIR SIZE (≤ 10).

Size (Unit: %)	Citeseer	Cora	Cora-A	DBLP-A	Enron	Eu	High	Primary	Ubuntu	Math
Total (≤ 10)	**98.61**	**100.00**	**92.16**	**92.51**	**98.56**	**96.68**	**100.00**	**100.00**	**100.00**	**100.00**
2	48.61	37.90	41.65	41.86	55.53	52.21	70.32	60.99	19.41	14.92
3	24.10	29.80	19.69	19.88	21.76	20.29	26.75	36.21	36.05	37.75
4	11.35	20.30	10.52	10.90	9.47	9.35	2.84	2.73	27.01	30.12
5	6.37	12.00	7.84	6.71	4.32	5.57	0.09	0.07	17.53	17.21
6	3.59	-	4.64	4.33	2.95	3.63	-	-	-	-
7	2.19	-	2.89	3.15	1.85	2.29	-	-	-	-
8	1.00	-	2.16	2.41	1.51	1.44	-	-	-	-
9	0.60	-	1.13	1.73	0.41	1.12	-	-	-	-
10	0.80	-	1.65	1.55	0.75	0.78	-	-	-	-

TABLE II
HYPERPARAMETER SEARCH SPACES FOR HNN MODELS.

Hyperparameter	**MHP**	**AHP**	**Hyper-SAGNN**	**NHP**
Epochs	500	500	500	500
Hidden Dimension	{64, 128}	400	64	{64, 128, 256}
Optimizer	Adam	Adam	Adam	Adam
Learning Rate	{1e-3, 5e-4, 1e-4}	Generator & Discriminator: {1e-3, 1e-4, 1e-5, 1e-6}	{1e-2, 1e-3, 1e-4}	{1e-2, 1e-3}
Batch Size	{128, 256, 512}	128	128	{32, 64, 128}
Number of Samples	{2, 5, 8, 10, 12, 15}	-	-	-
Normalization Factor (α, β)	-	{(0,0), (1,1)}	-	-
Dropout	-	{0, 0.5}	-	-
node2vec (p,q)	-	-	{(1,1), (50,0.5), (50,1), (100,0.5)}	-

TABLE III
AVERAGE F1 SCORE. <u>HYPERSEARCH ACHIEVES STRONG PERFORMANCE UNDER AVERAGE F1 SCORE ACROSS MOST SETTINGS.</u> WE REPORT THE AVERAGE VALUES AND STANDARD DEVIATIONS OVER THE FIVE RANDOM SPLITS OF AVERAGE F1 SCORE, EVALUATED ON THE FOUR DATASETS WITHOUT EDGE TIMESTAMPS. FOR EACH SETTING, THE BEST AND SECOND-BEST METHODS ARE HIGHLIGHTED IN GREEN AND YELLOW, RESPECTIVELY.

Dataset	**Citeseer**			**Cora**			**Cora-A**			**DBLP-A**		
Method ($\downarrow$) **/** $\mathcal{K}$ ($\rightarrow$)	$1\times$	$2\times$	$5\times$	$1\times$	$2\times$	$5\times$	$1\times$	$2\times$	$5\times$	$1\times$	$2\times$	$5\times$
HyperSearch (Proposed)	40.0 (3.6)	43.8 (2.8)	49.5 (2.0)	48.7 (1.6)	51.3 (1.3)	54.2 (0.8)	18.8 (3.5)	23.6 (1.8)	29.7 (1.3)	17.7 (0.8)	20.5 (0.7)	27.0 (0.5)
CNS	37.5 (1.2)	41.9 (0.7)	48.1 (0.5)	45.3 (1.8)	49.0 (1.0)	53.5 (1.1)	17.1 (1.0)	21.0 (0.7)	26.9 (1.1)	16.7 (0.2)	20.7 (0.2)	26.7 (0.2)
HPRA	25.9 (1.2)	31.4 (2.1)	36.6 (0.8)	33.2 (1.3)	37.1 (0.8)	42.8 (0.9)	12.8 (1.7)	18.1 (1.2)	26.5 (0.8)	12.2 (0.2)	17.5 (0.2)	28.5 (0.1)
MHP	33.9 (1.9)	39.1 (2.0)	44.5 (1.3)	39.9 (0.5)	42.8 (0.4)	46.8 (0.8)	19.3 (2.2)	24.2 (1.9)	31.2 (1.7)	-	-	-
MHP-C	31.7 (2.2)	37.0 (1.9)	-	46.2 (1.1)	48.2 (0.9)	-	19.4 (1.9)	21.2 (1.5)	26.0 (1.3)	-	-	-
AHP-C	34.8 (3.0)	39.1 (2.8)	-	41.8 (1.7)	44.7 (1.4)	-	16.3 (2.0)	19.7 (2.2)	24.0 (2.2)	-	-	-
SAGNN-C	30.2 (1.3)	34.4 (2.0)	-	42.0 (1.9)	44.9 (1.7)	-	15.0 (1.5)	17.7 (1.3)	20.7 (1.4)	13.8 (0.3)	16.6 (0.3)	20.1 (0.4)
NHP-C	36.9 (1.3)	42.2 (1.2)	-	51.8 (0.7)	55.4 (0.2)	-	19.6 (1.7)	23.7 (1.2)	28.9 (2.0)	16.0 (0.4)	20.1 (0.4)	25.8 (0.2)
MHP-H	31.6 (2.2)	36.9 (2.0)	-	46.2 (1.1)	48.3 (0.9)	53.0 (0.6)	19.2 (1.8)	21.1 (1.5)	26.0 (1.2)	-	-	-
AHP-H	24.5 (6.7)	27.6 (7.0)	-	31.4 (3.5)	35.4 (3.4)	40.8 (2.7)	12.4 (3.0)	15.8 (3.6)	22.2 (4.6)	-	-	-
SAGNN-H	17.6 (2.2)	20.1 (1.4)	-	28.4 (1.5)	32.2 (1.2)	36.0 (1.4)	10.0 (1.5)	13.4 (1.3)	19.5 (1.4)	11.3 (0.3)	15.5 (0.3)	-
NHP-H	21.9 (2.2)	24.9 (1.3)	-	35.8 (1.6)	39.1 (1.2)	43.1 (0.9)	7.8 (1.0)	10.2 (1.1)	14.7 (0.9)	12.7 (0.3)	17.5 (0.3)	-

-: out-of-time (> 2 days).

1) **Training**: The existing HNN method is trained for the hyperedge prediction task, on the same dataset split used in the earlier stages.
2) **Scoring**: The trained HNN is applied to the hyperedge candidates generated in the first step to estimate their likelihood of being ground-truth hyperedges.
3) **Filtering**: The top-ranked candidates, based on the HNN scoring function, are retained for the final prediction.

For instance, if the target number of hyperedge candidates is $\mathcal{K}$, we first predict $2\mathcal{K}$ candidates in the first step of the algorithm. These candidates are then refined using the trained HNN, with the final output consisting of the most promising $\mathcal{K}$ hyperedge candidates. This two-step approach ensures that the initial candidate generation process is efficient yet leverages the expressive power of HNNs for more accurate predictions.

<u>**Results.**</u> We assessed the impact of incorporating refinement with Hypergraph Neural Networks (HNNs). Specifically, we compared the performance of HyperSearch with variants that applied two HNN models, MHP and AHP. As shown in Fig. 2, the results indicate that refinement with HNNs did not consistently improve prediction accuracy. This suggests that the scoring mechanism based solely on our empirical observations is already robust and effective, reducing the necessity for additional refinement in many cases.

F. Additional Results on "Q2. Speed and Scalability" (Supplementing Sect. VI-C)

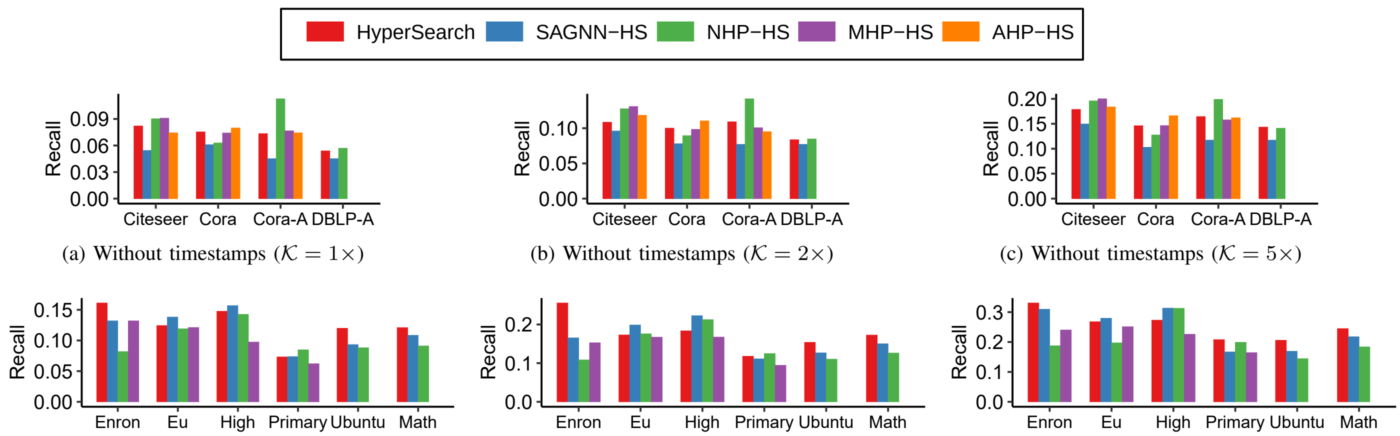
We evaluated the running times of the considered methods and analyzed the scalability of HyperSearch by examining its runtime as a function of the number of hyperedges in the input

TABLE IV

AVERAGE F1 SCORE. <u>HYPERSEARCH ACHIEVES STRONG PERFORMANCE UNDER AVERAGE F1 SCORE IN MOST SETTINGS.</u> WE REPORT THE AVERAGE F1 SCORE ON THE SIX DATASETS WITH EDGE TIMESTAMPS (NO STANDARD DEVIATION SINCE THERE IS ONLY ONE CHRONOLOGICAL SPLIT). FOR EACH SETTING, THE BEST AND SECOND-BEST METHODS ARE HIGHLIGHTED IN GREEN AND YELLOW, RESPECTIVELY.

Dataset	**Enron**			**Eu**			**High**			**Primary**			**Ubuntu**			**Math-sx**		
Method (↓) / $\mathcal{K}$ (→)	1×	2×	5×	1×	2×	5×	1×	2×	5×	1×	2×	5×	1×	2×	5×	1×	2×	5×
HyperSearch (Proposed)	60.6	62.1	63.4	58.5	59.9	62.4	58.8	61.9	64.4	57.8	60.8	63.2	73.3	73.7	74.2	74.6	75.5	76.4
CNS	54.6	57.9	63.0	56.0	59.0	63.3	58.5	59.1	59.6	57.7	58.5	59.5	62.4	64.1	66.7	66.5	68.7	71.7
HPRA	51.7	53.9	55.7	57.5	59.0	61.7	60.2	62.6	65.6	58.3	59.8	64.1	61.7	63.2	65.1	68.4	69.8	71.8
MHP	45.5	47.8	51.2	46.7	47.7	49.6	49.6	50.2	53.0	53.2	56.5	62.0	-	-	-	-	-	-
MHP-C	55.6	56.2	61.1	57.2	58.3	61.7	57.4	59.5	60.5	57.0	58.6	61.1	-	-	-	-	-	-
SAGNN-C	45.6	48.5	51.3	59.5	62.5	66.3	52.3	54.3	54.7	57.2	57.9	59.0	64.3	66.2	68.7	69.5	71.7	74.5
NHP-C	52.9	55.9	59.7	54.9	57.7	61.9	56.2	56.4	56.7	55.9	56.6	57.2	61.1	62.9	65.1	64.7	67.0	69.9
MHP-H	57.7	58.2	62.1	56.5	57.5	61.5	58.3	60.0	61.2	57.2	58.9	60.9	-	-	-	-	-	-
SAGNN-H	49.4	51.5	53.5	59.7	61.6	64.6	60.8	62.3	66.5	56.6	58.8	61.8	64.5	66.0	-	70.6	72.1	-
NHP-H	51.7	53.2	55.4	56.5	58.3	60.8	59.6	61.1	64.6	57.7	59.6	62.4	60.2	61.5	-	67.3	68.7	-

-: out-of-time (> 2 days).



(a) Without timestamps ($\mathcal{K} = 1\times$) (b) Without timestamps ($\mathcal{K} = 2\times$) (c) Without timestamps ($\mathcal{K} = 5\times$)

(d) With timestamps ($\mathcal{K} = 1\times$) (e) With timestamps ($\mathcal{K} = 2\times$) (f) With timestamps ($\mathcal{K} = 5\times$)

Fig. 2. Refinement with HNNs does not consistently enhance the prediction accuracy of HyperSearch.

hypergraph across different datasets. The results, presented in Fig. 3 and Fig. 4, reveal the following key observations:

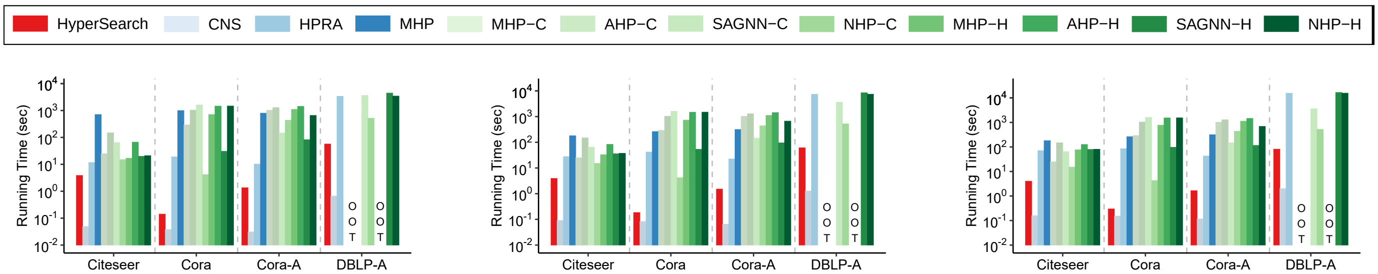
- HyperSearch runs faster than deep learning-based methods (highlighted in green in Fig. 3) in most cases, demonstrating its computational efficiency.
- The runtime of HyperSearch scales nearly linearly with the number of hyperedges in the input hypergraph, with a regression slope of 1.05 to 1.17 (Fig. 4). This confirms that HyperSearch scales efficiently as the input size increases.

<u>**Scalability on large-scale synthetic hypergraphs.**</u> To further verify the scalability of HyperSearch in controlled large-scale settings, we constructed synthetic datasets by replicating a real hypergraph m times, with $m \in \{2, 3, 5\}$. Specifically, we duplicated both the node set and hyperedge set m times, resulting in a larger hypergraph where each replicated component preserves the original structure but uses disjoint node indices. This design allows us to isolate input size as the only varying factor, while maintaining fixed structural complexity.

We then measured the runtime of HyperSearch as the number of hyperedges in the input hypergraph increases proportionally with the replication factor. As shown in Fig. 5, the runtime exhibits an approximately linear growth with respect to the number of hyperedges in the input hypergraph, with a regression slope of 0.99 to 1.20. This result confirms that HyperSearch scales efficiently under pure input size growth, even on large synthetic graphs with millions of hyperedges.

G. Scalability w.r.t. the Number of Predicted Hyperedges. (Supplementing Sect. VI-C)

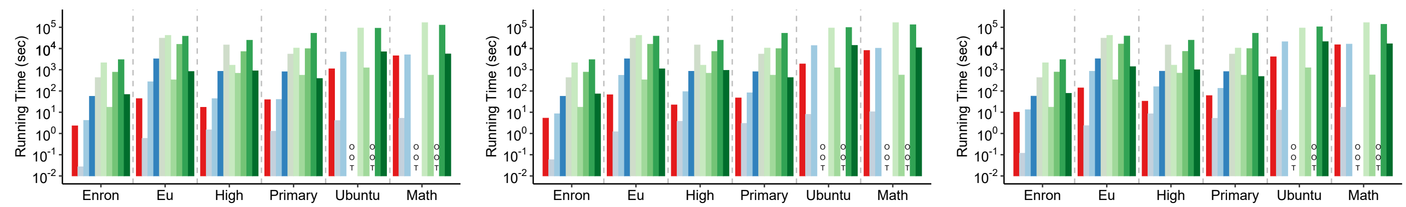
To complement our runtime analysis, we examined how the running time of HyperSearch scales with the number of predictions. Specifically, we varied the number of predicted hyperedges to be 1×, 2×, and 5× the size of the test set for each dataset, and measured the runtime accordingly. We then regressed the runtime against the number of predictions in log-log scale to assess the scalability trend, where a slope close to 1 indicates near-linear scaling and a lower slope suggests sublinear behavior. Figure 6 visualizes this relationship between the number of predicted hyperedges and runtime for each dataset. As shown in the plots, the runtime exhibits sublinear or nearly linear growth, with regression slopes ranging from 0.025 (Citeseer) to 0.909 (Enron). This demonstrates that HyperSearch maintains efficient scalability across diverse datasets as the prediction output size increases.



(a) Without timestamps ($\mathcal{K} = 1\times$)

(b) Without timestamps ($\mathcal{K} = 2\times$)

(c) Without timestamps ($\mathcal{K} = 5\times$)

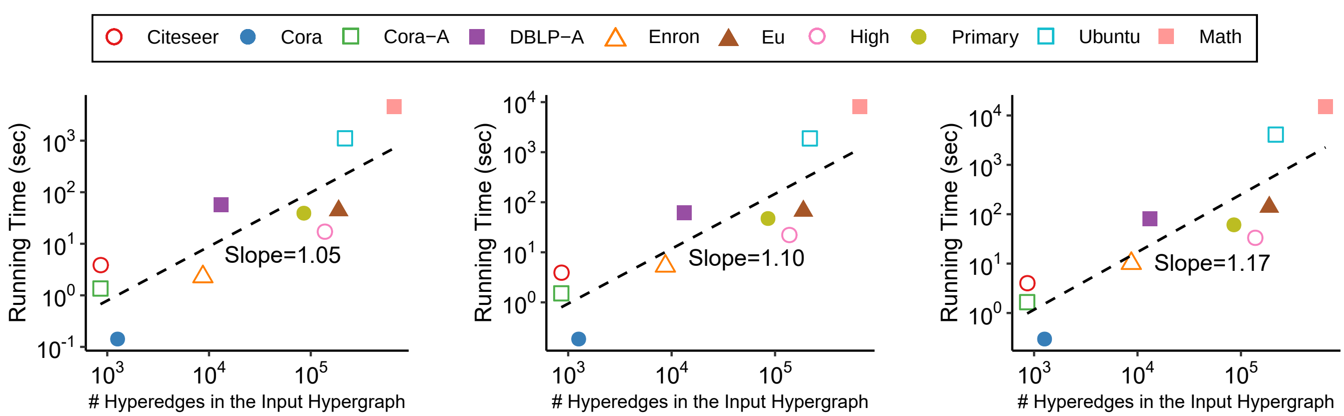


(d) With timestamps ($\mathcal{K} = 1\times$)

(e) With timestamps ($\mathcal{K} = 2\times$)

(f) With timestamps ($\mathcal{K} = 5\times$)

Fig. 3. HyperSearch runs faster than deep learning-based methods (highlighted in green) in most cases, demonstrating its computational efficiency. This suggests that HyperSearch maintains efficient scalability as the prediction task size increases.

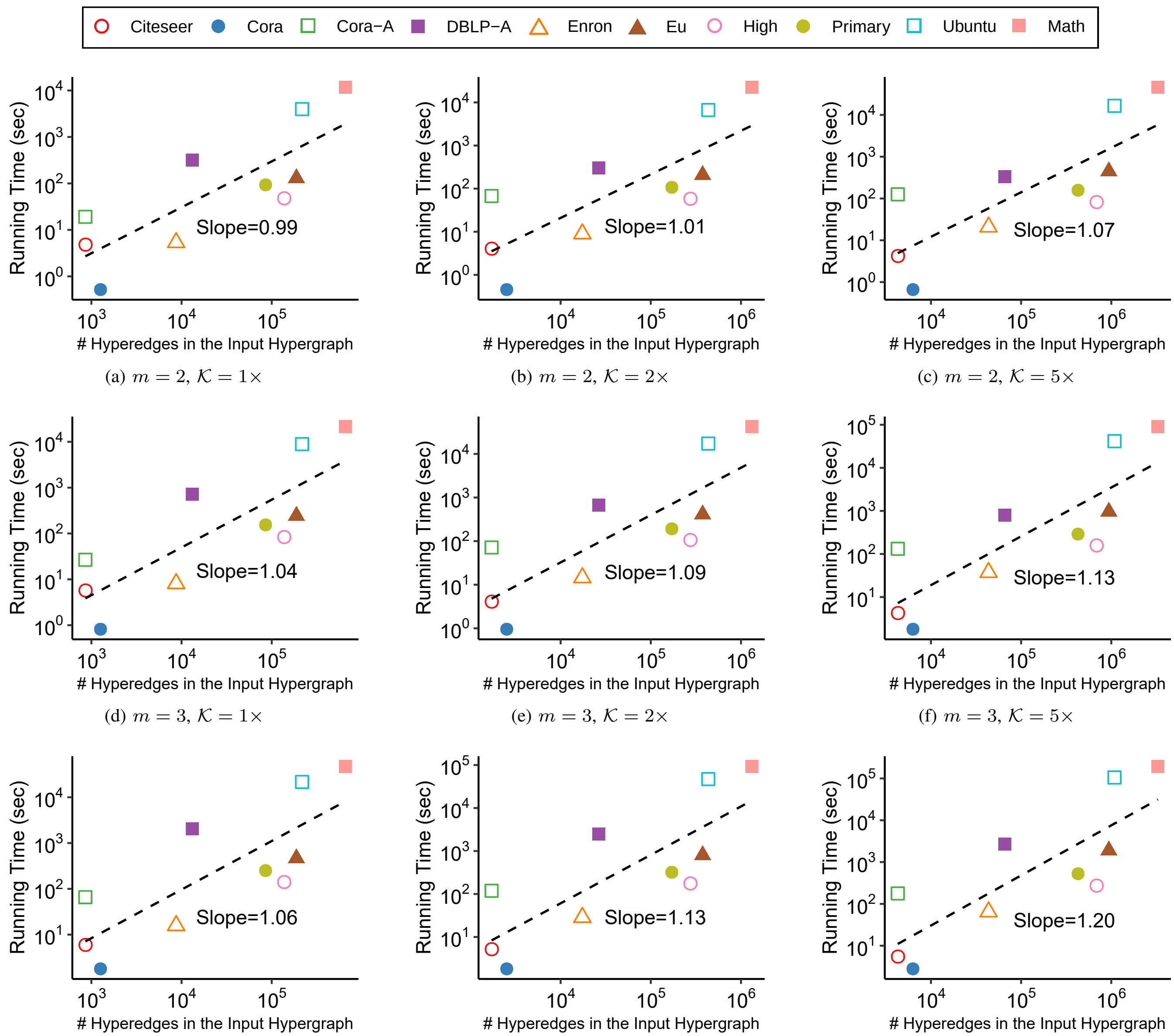


(a) $\mathcal{K} = 1\times$

(b) $\mathcal{K} = 2\times$

(c) $\mathcal{K} = 5\times$

Fig. 4. The runtime of HyperSearch scales nearly linearly with the number of hyperedges in the input hypergraph, with a regression slope of 1.05 to 1.11, indicating its efficient scalability.



(a) $m = 2$, $\mathcal{K} = 1\times$

(b) $m = 2$, $\mathcal{K} = 2\times$

(c) $m = 2$, $\mathcal{K} = 5\times$

(d) $m = 3$, $\mathcal{K} = 1\times$

(e) $m = 3$, $\mathcal{K} = 2\times$

(f) $m = 3$, $\mathcal{K} = 5\times$

(g) $m = 5$, $\mathcal{K} = 1\times$

(h) $m = 5$, $\mathcal{K} = 2\times$

(i) $m = 5$, $\mathcal{K} = 5\times$

Fig. 5. The runtime of HyperSearch scales nearly linearly with the number of hyperedges, indicating its efficient scalability under input size growth.

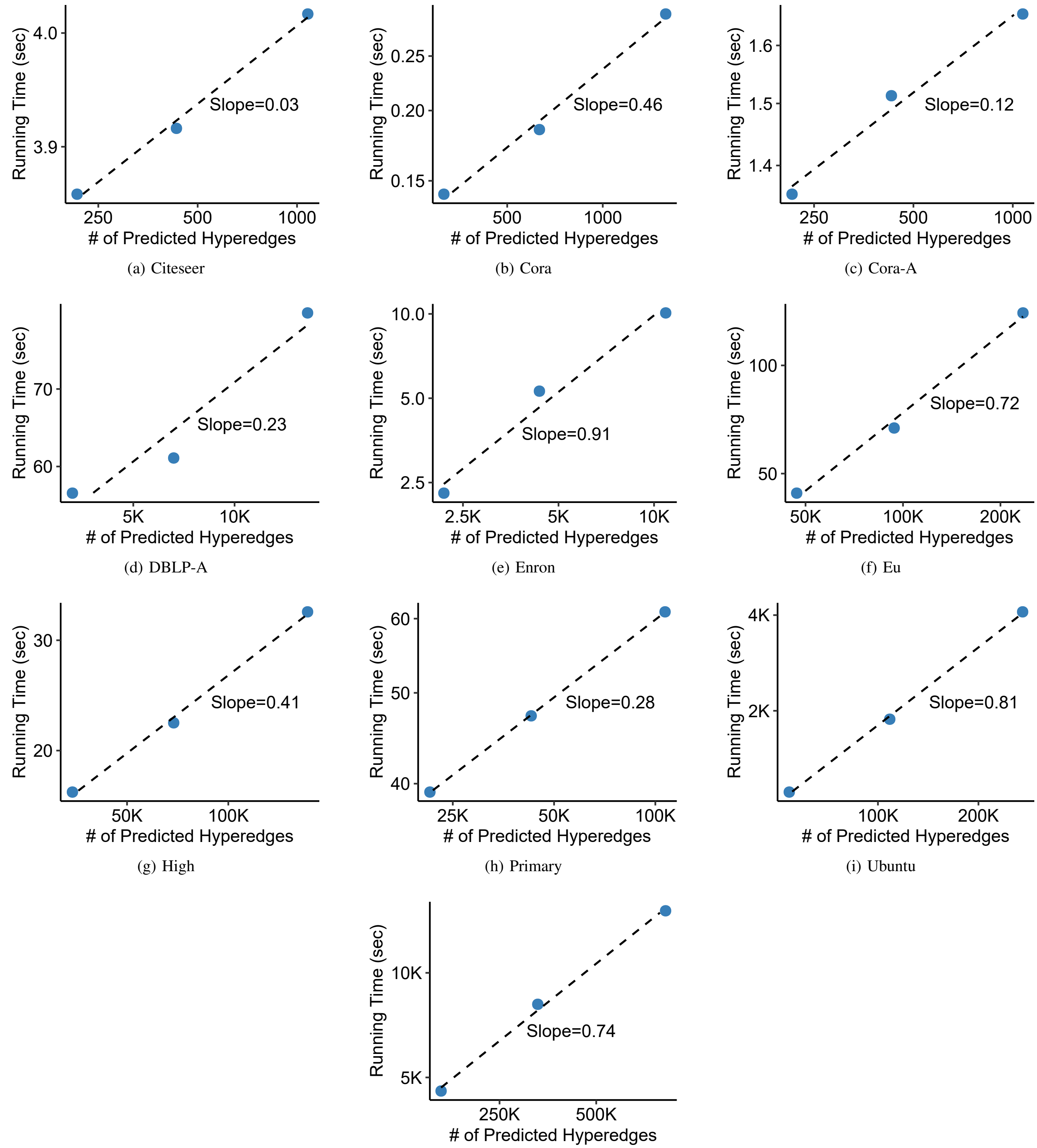


Fig. 6. The runtime of HyperSearch scales sublinearly or nearly linearly with the number of predicted hyperedges across all datasets, indicating its efficient scalability as the prediction output size increases.