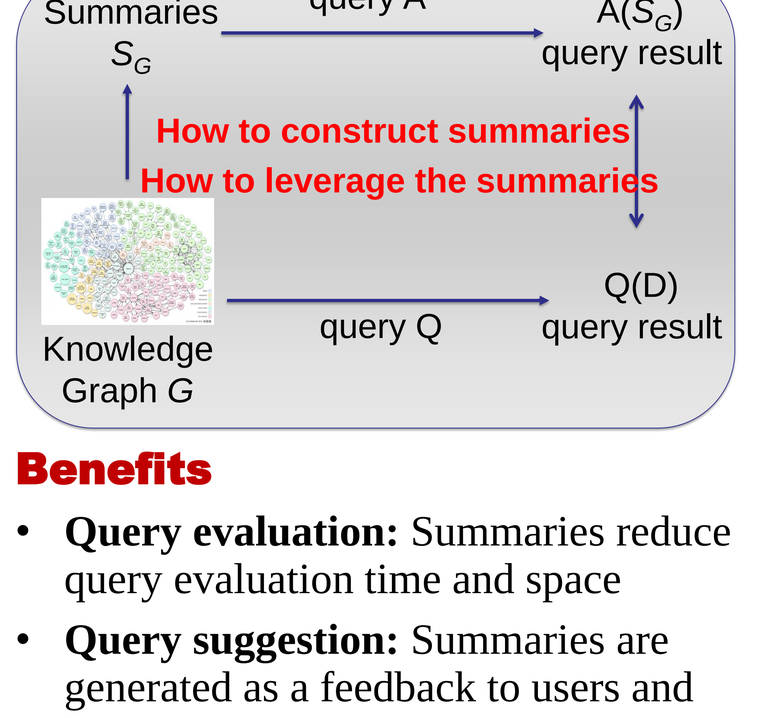
# Parallel Graph Summarization for Knowledge Search

Qi Song, Mohammad Hossein Namaki, Peng Lin and Yinghui Wu
{ qsong, mnamaki, plin1, yinghui} @eecs.wsu.edu
School of Electronic Engineering and Computer Science, Washington State University

## Introduction

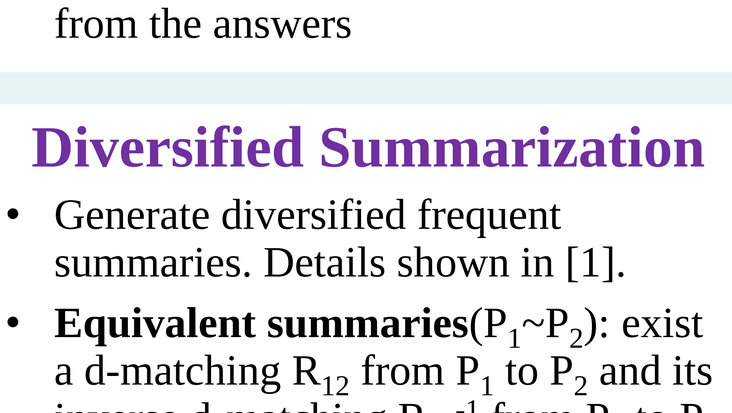
### Motivation

- Querying heterogeneous and large-scale knowledge graphs are **expensive**
- Graph patterns can benefit knowledge search by **suggesting "views"**




### Contribution

- **Use summarization to facilitate query evaluation**

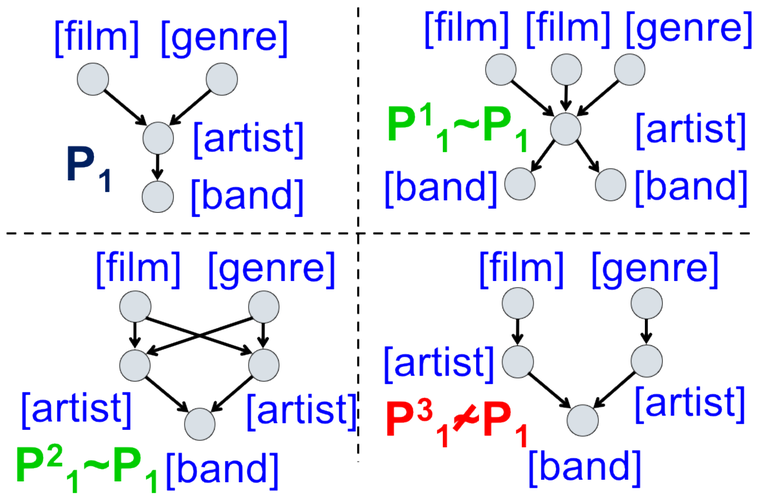



### Benefits

- **Query evaluation:** Summaries reduce query evaluation time and space
- **Query suggestion:** Summaries are generated as a feedback to users and help them write more accurate queries
- **Result understanding:** Help end-users to better capture the information from the answers

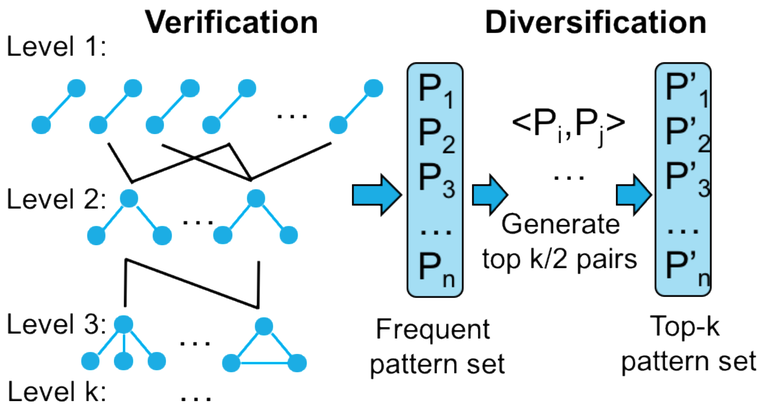
## Diversified Summarization

- Generate diversified frequent summaries. Details shown in [1].
- **Equivalent summaries**($P_1$~$P_2$): exist a d-matching $R_{12}$ from $P_1$ to $P_2$ and its inverse d-matching $R_{12}^{-1}$ from $P_2$ to $P_1$
- **Reduced summaries**: there exists no smaller summaries P' such that P~P'




## Sequential Algorithms

- ***approxDis*:** generate top-*k* diversified patterns (2-approximation guarantee)



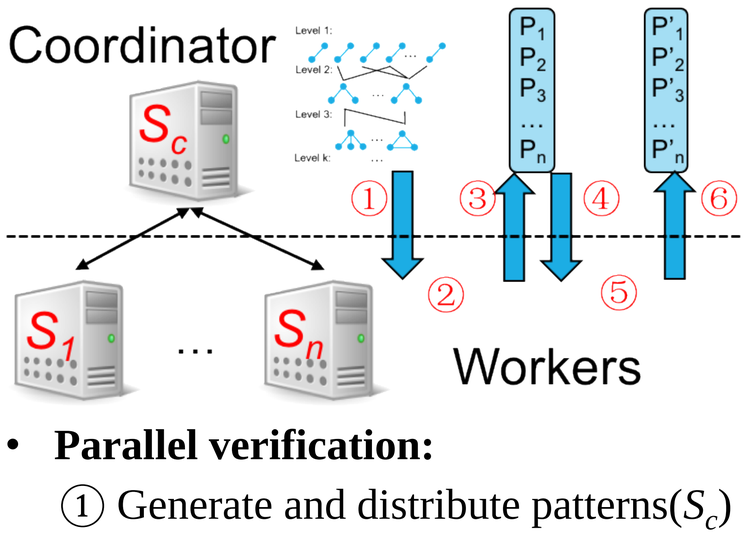

- **Pattern verification:** Pattern generation via edge expansion and validation via d-simulation
- **Pattern diversification:** Diversify frequent patterns via pair-wise score and output top-k ones
- ***streamDis*:** maintain a pattern cache and perform. Anytime algorithm and still preserve 2-approximation

## Parallel Algorithm (*paraDis*)

- **Parallel scalability:** denote the time cost of *approxDis* as $t(|G|,b_p,k)$. *paraDis* is parallel scalable as its running time by *n* processors is:

$$T(|G|, b_p, k, n) = O(\frac{t(|G|, b_p, k)}{n})$$

- ***paraDis:*** following BSP model, in each super step perform verification and diversification in parallel
- **Architecture:**




- **Parallel verification:**
  - ① Generate and distribute patterns($S_c$)
  - ② Pattern validation in parallel ($S_j$)
  - ③ Gather validated frequent patterns and their matches from workers ($S_c$)
- **Parallel diversification:**
  - ④ Distribute summary pairs ($S_c$)
  - ⑤ Pairwise distance calculation and local top-*k* pairs generation ($S_j$)
  - ⑥ Collect local top-k pairs and update global top-k summaries ($S_c$)
- **Load balancing:** iteratively assign work units with the smallest cost to the workers with the least load
- ***streamDis* parallelization:** $S_c$ caches a set of summaries and their top-k most different summaries. Top-k summaries can be returned whenever requested.

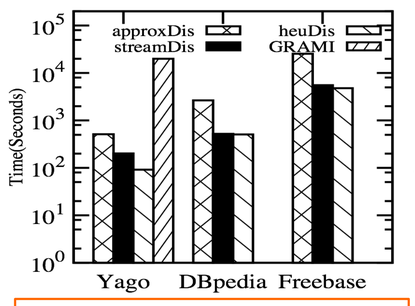
## Evaluation

### Datasets

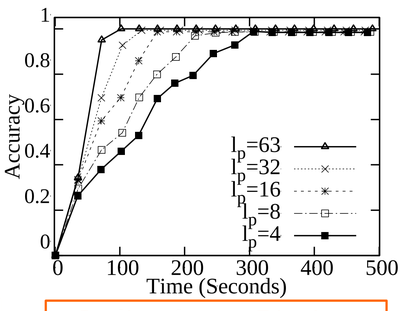
| | Yago | DBPedia | Freebase |
|---|---|---|---|
| # Nodes | 1.54M | 4.86M | 40.32M |
| # Edges | 2.37M | 15M | 63.2M |
| # Labels | 0.32M | 676 | 9630 |

### Results

- Performance of sequential summarization

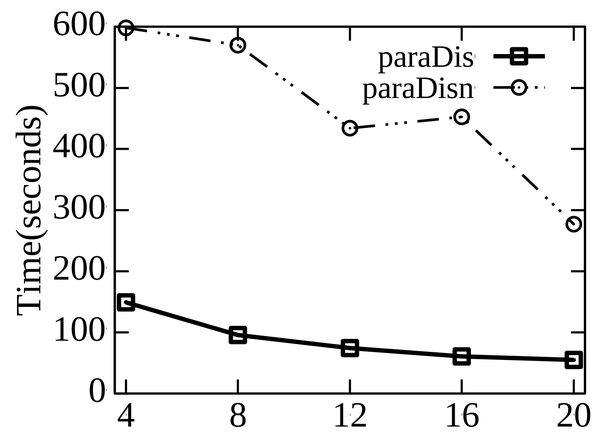



Summarization on Real life datasets
GraMi can not handle large datasets

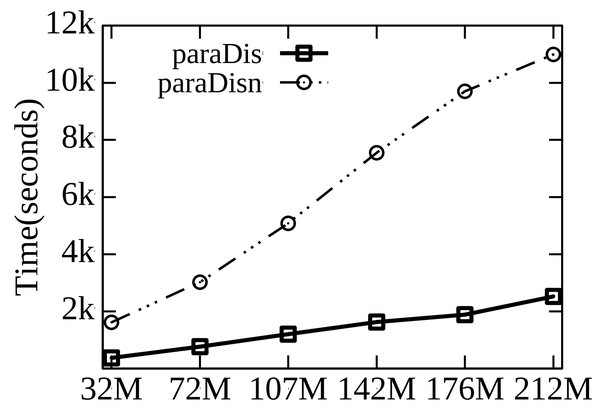



Quality of streamDis given
Different cache size (YAGO)

- Performance of parallel summarization

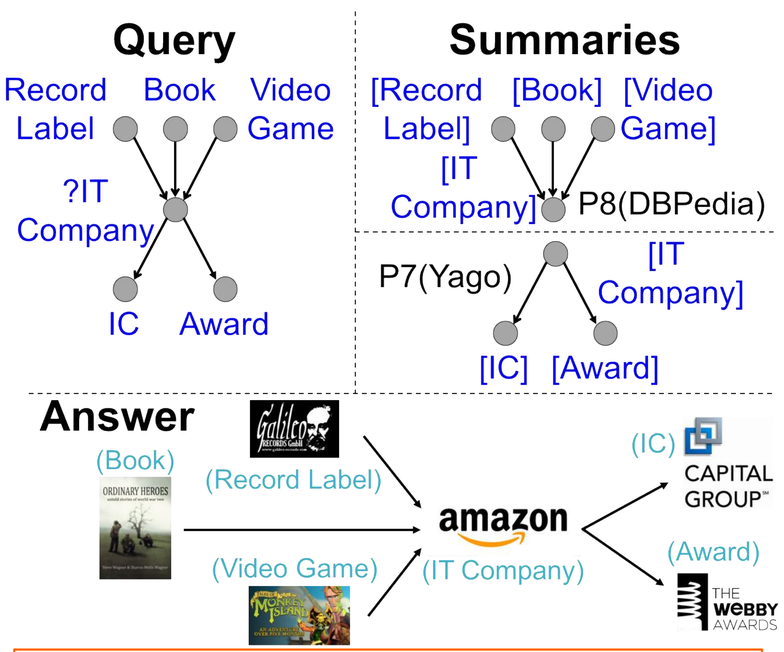



Execution time varying #of workers (YAGO)




Execution time varying
size of graphs (20 workers)

- **Case study:**




Cross-domain queries over DBPedia and Yago

## Reference and Acknowledgment

- [1]Qi Song, Yinghui Wu, and Xin Luna Dong. 2016. Mining Summaries for Knowledge Graph Search. In ICDM.
- This project is supported by NSF IIS-1633629