ECML PKDD 2024 VILNIUS 9-13 SEPTEMBER

KAIST

Soongsil University

Random Walk with Restart on Hypergraphs: Fast Computation and an Application to Anomaly Detection

Jaewan Chun

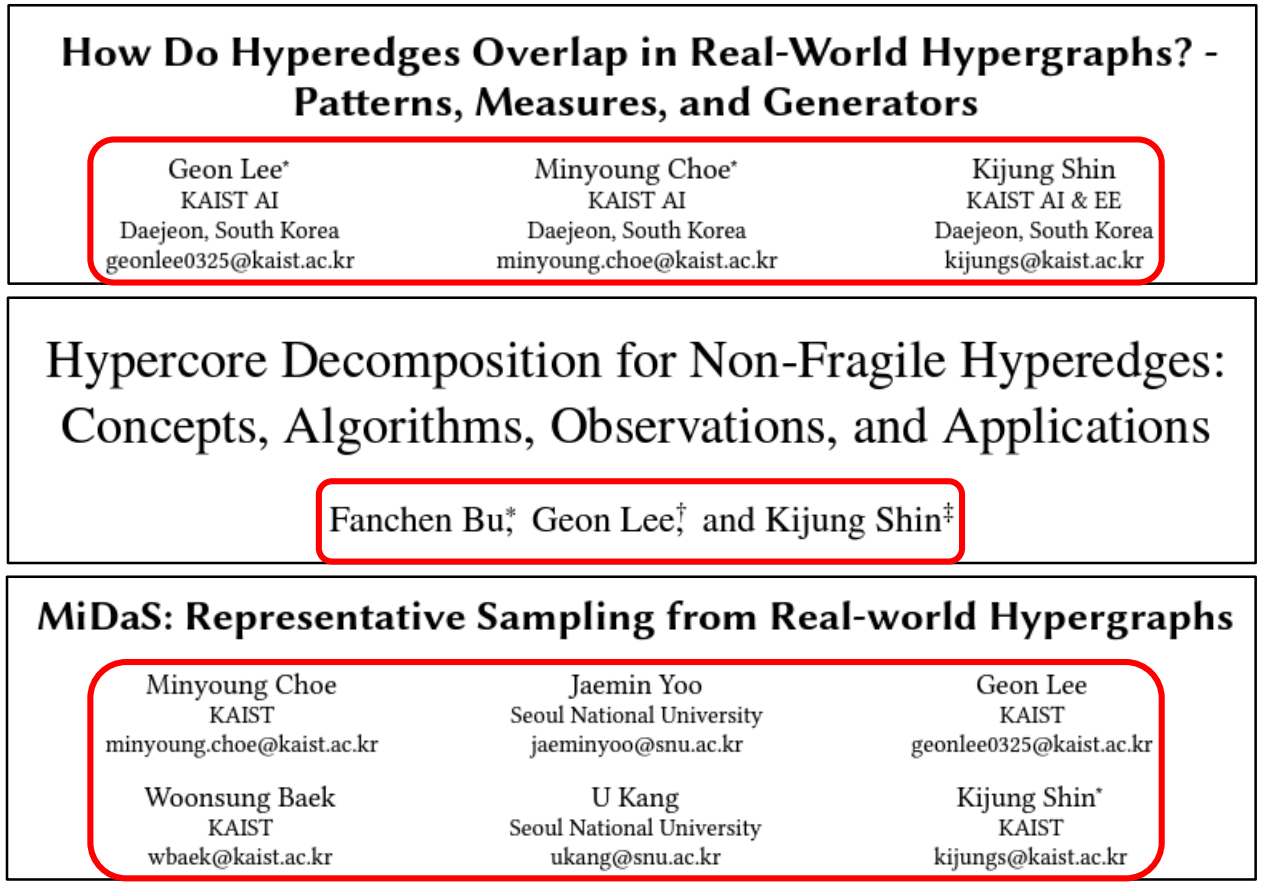
Geon Lee

Kijung Shin

Jinhong Jung

Group Interactions are Everywhere

- Example 1: Co-authorship of researchers



How Do Hyperedges Overlap in Real-World Hypergraphs? - Patterns, Measures, and Generators

Geon Lee* KAIST AI Daejeon, South Korea geonlee0325@kaist.ac.kr

Minyoung Choe* KAIST AI Daejeon, South Korea minyoung.choe@kaist.ac.kr

Kijung Shin KAIST AI & EE Daejeon, South Korea kijungs@kaist.ac.kr

Hypercore Decomposition for Non-Fragile Hyperedges: Concepts, Algorithms, Observations, and Applications

Fanchen Bu*, Geon Lee†, and Kijung Shin‡

MiDaS: Representative Sampling from Real-world Hypergraphs

Minyoung Choe KAIST minyoung.choe@kaist.ac.kr

Jaemin Yoo Seoul National University jaeminyoo@snu.ac.kr

Geon Lee KAIST geonlee0325@kaist.ac.kr

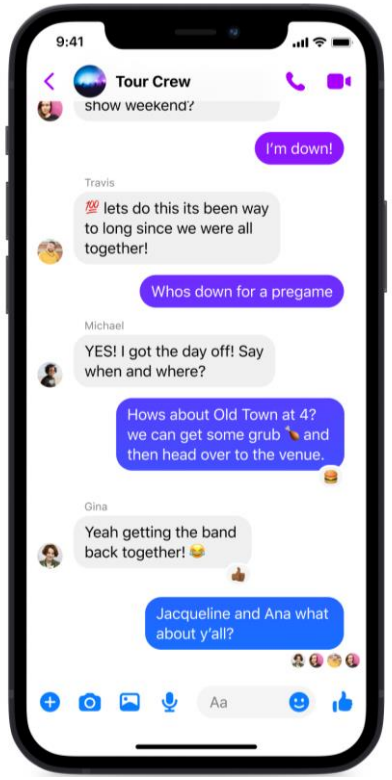
Woonsung Baek KAIST wbaek@kaist.ac.kr

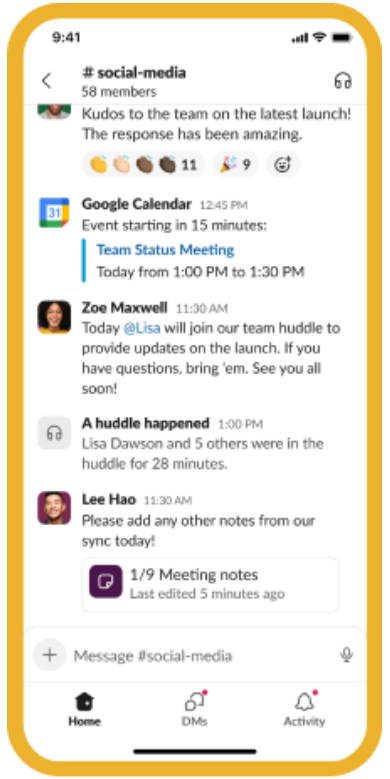
U Kang Seoul National University ukang@snu.ac.kr

Kijung Shin* KAIST kijungs@kaist.ac.kr

Group Interactions are Everywhere (cont.)

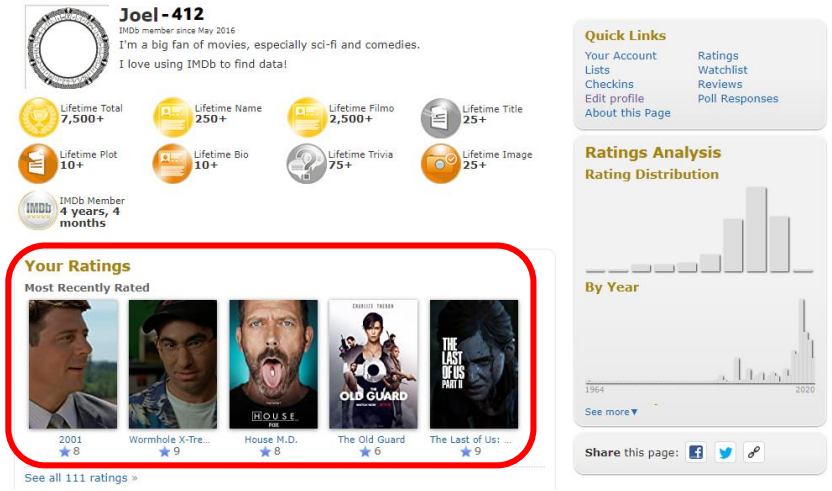
- Example 2: Online group chats

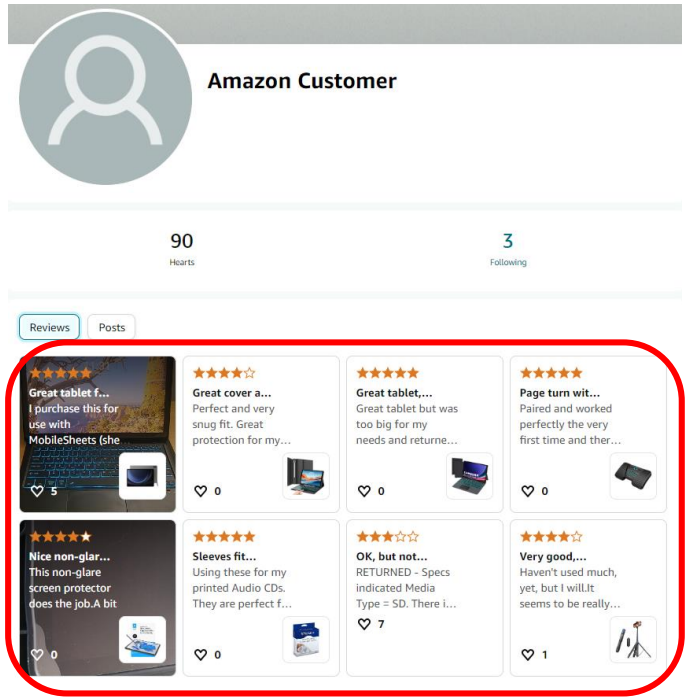




Group Interactions are Everywhere (cont.)

- Example 3: Reviewed item set





Hypergraphs: Model for Group Interactions

- **Hypergraphs** model group interactions among individuals or objects
- Each **hyperedge** is a subset of any number of nodes

Authors (Nodes)

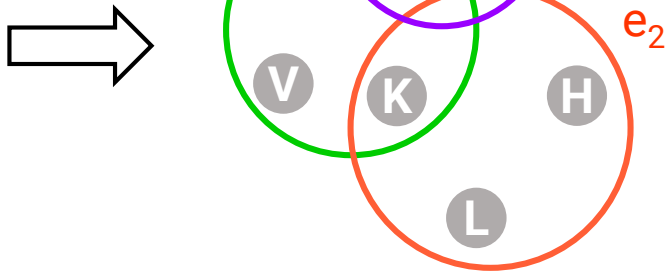
Austin Benson (**B**)	Daniel Huttenlocher (**H**)
Junteng Jia (**J**)	Jon Kleinberg (**K**)
Jure Leskovec (**L**)	Nate Veldt (**V**)

Publications (Hyperedges)

e_1: (V, B, K) NeruIPS'21

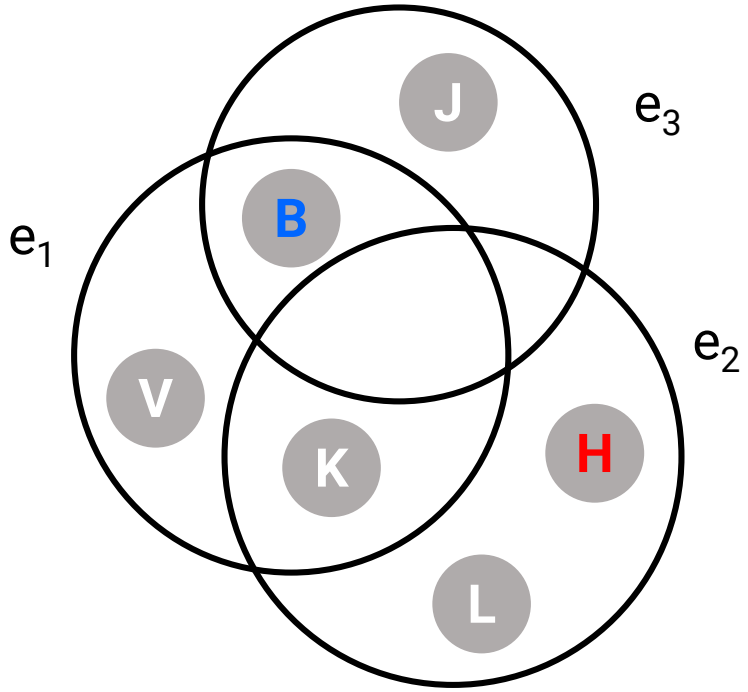
e_2: (L, H, K) WWW'10

e_3: (J, B) KDD'20



Measuring Proximity between Nodes

- Measuring **proximity** between nodes in a hypergraph:
  - How **close are two nodes** in a hypergraph to each other?



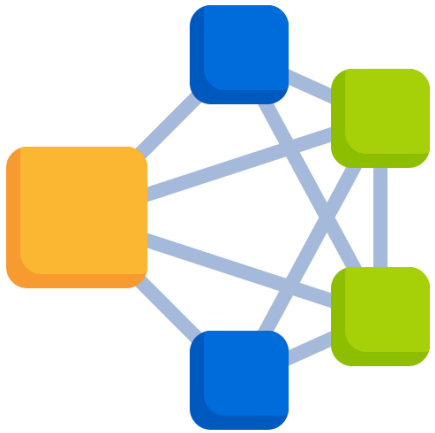
How close is node H to node B?
That is, what is the proximity of H w.r.t. B?

Measuring Proximity between Nodes (cont.)

- Measuring **proximity** between nodes in a hypergraph
  - How **close are two nodes** in a hypergraph to each other?

- Practical applications of measuring node proximity:

Anomaly Detection



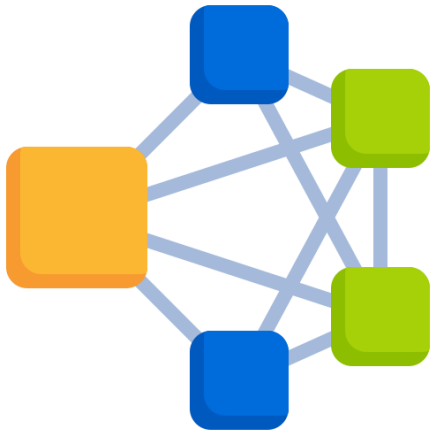
Clustering

Personalized Ranking

Measuring Proximity between Nodes: RWR

- Widely used method: **Random Walk with Restart (RWR)**
  - Proposed for graphs → extended to hypergraphs
  - Consider global network structure and multi-faceted relationship
- Successful applications of RWR:

Anomaly Detection
(ICDM'05)



Clustering
(CIKM'20)

Personalized Ranking
(ICDM'16)

RWR on Hypergraphs?

- RWR on hypergraphs has been **underexplored**

- Potential reason: lack of **fast** computation methods!
  - Various computation methods exist only for **graphs**
    - Iterative methods
      - Power iteration (Page'99)
    - Preprocessing methods
      - BEAR (SIGMOD'15)
      - BePI (SIGMOD'17)
    - Approximation-based methods
      - Spectral sparsification (ICESS'19)
      - Eulerian Laplacian approximation (FOCS'16)

RWR on Hypergraphs? (cont.)

- RWR on hypergraphs has been **underexplored**
- Potential reason: lack of **fast** computation methods!

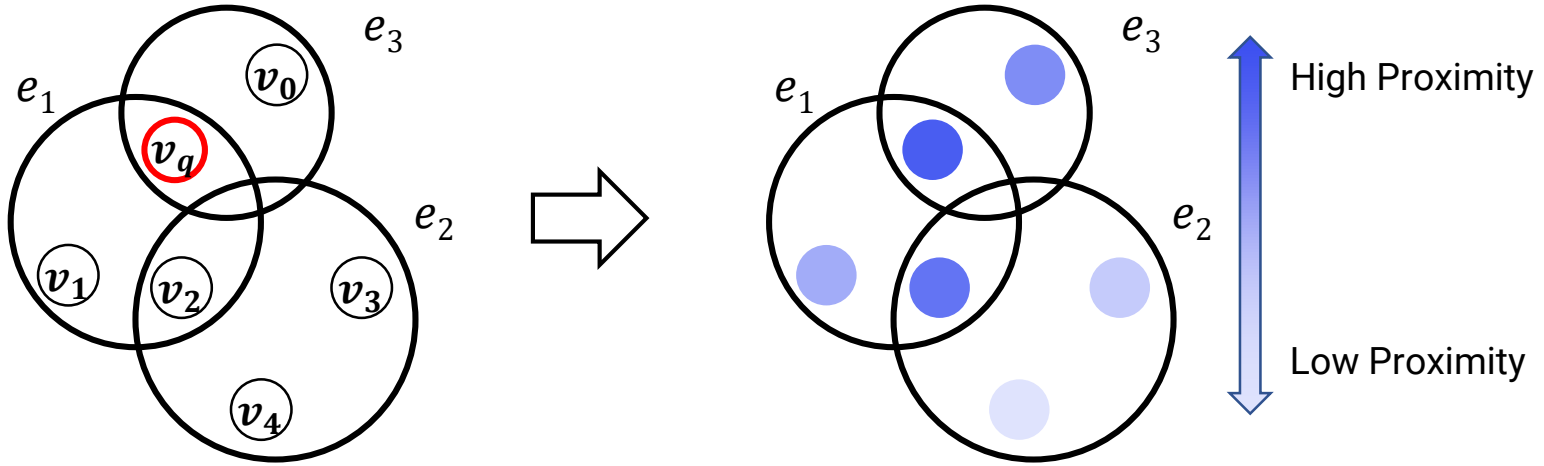
Our goal: **fast** RWR computation method for **hypergraphs**

Roadmap

- Overview
- Random Walk with Restart (RWR) on Hypergraphs <<
- Proposed Algorithm: ARCHER
- Empirical Evaluation of ARCHER
- Application: Anomaly Detection
- Conclusion

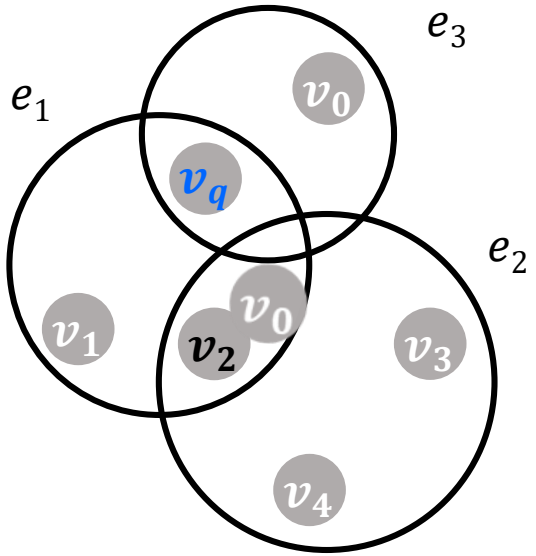
RWR on Hypergraphs: Inputs and Outputs

- **Given**
  - Hypergraph $G_H = (V, E)$
    - Node set V and hyperedge set E
  - Query node $\boldsymbol{v_q}$
  - Restart probability c
- **Output:** proximity between each node and $\boldsymbol{v_q}$



RWR on Hypergraphs: Definition

- Assume a random surfer who either..



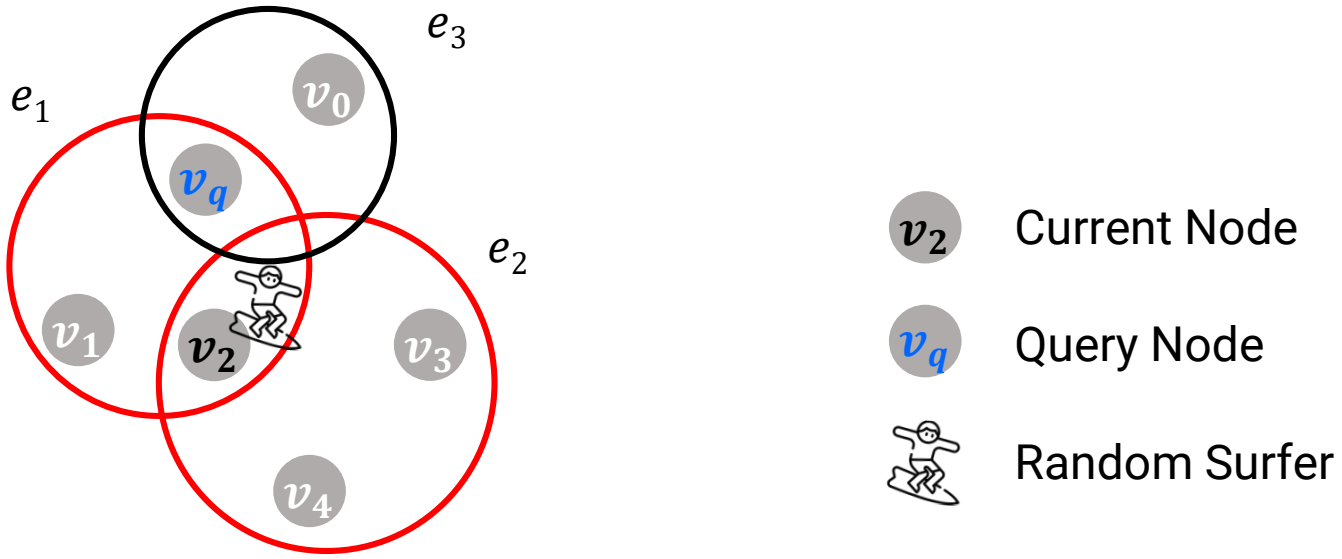
Current Node

Query Node


Random Surfer

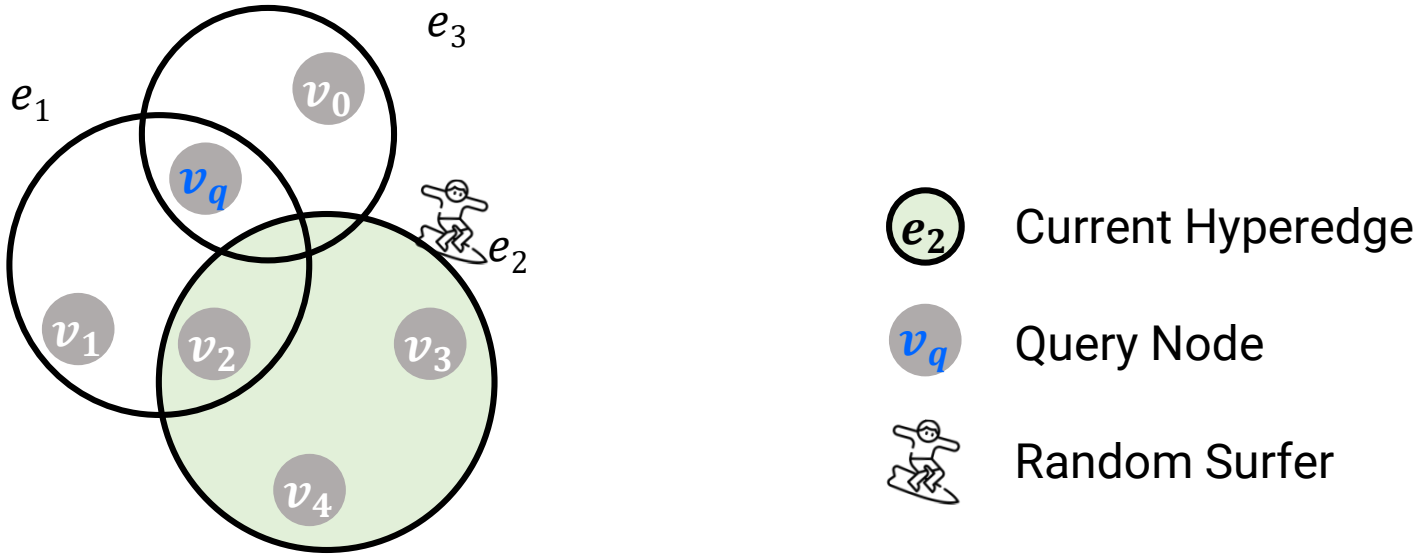
RWR on Hypergraphs: Definition (cont.)

- Assume a random surfer who either..
  - A1: Random walk (probability $1 - c$)
    - A1-1: Select a hyperedge e containing current node



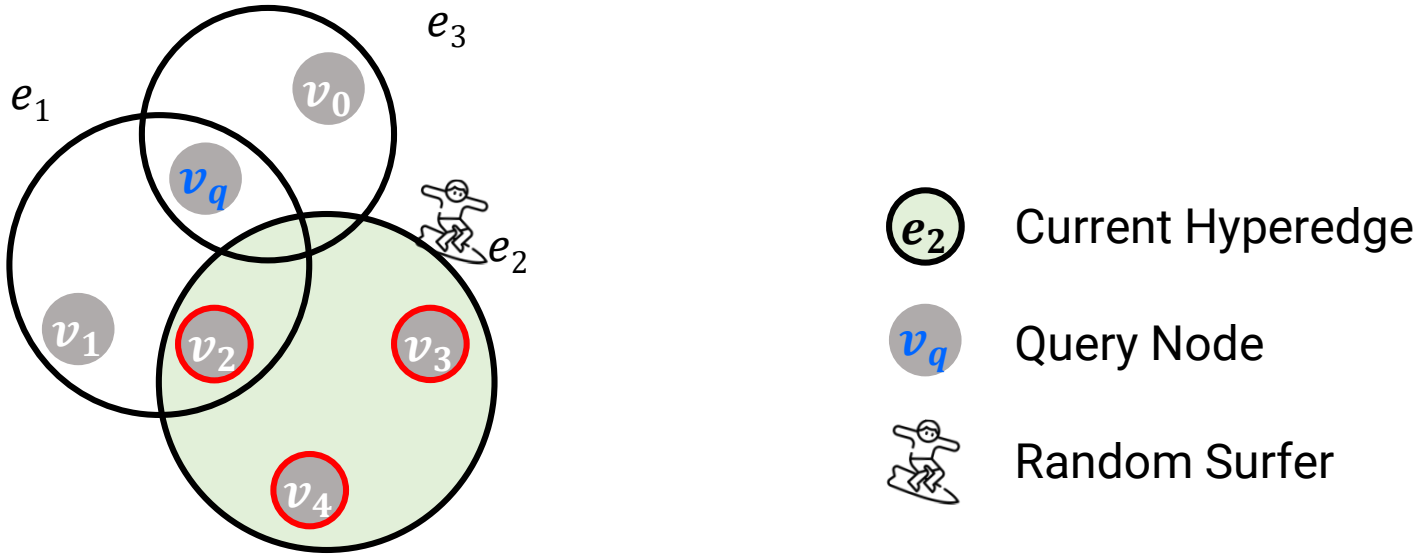
RWR on Hypergraphs: Definition (cont.)

- Assume a random surfer who either..
  - A1: Random walk (probability $1 - c$)
    - A1-1: Select a hyperedge e containing current node



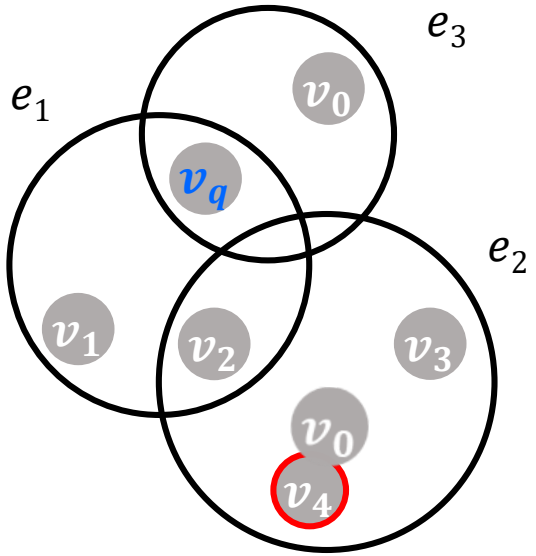
RWR on Hypergraphs: Definition (cont.)

- Assume a random surfer who either..
  - A1: Random walk (probability $1 - c$)
    - A1-1: Select a hyperedge e containing current node
    - A1-2: Move to a node v selected from the hyperedge e



RWR on Hypergraphs: Definition (cont.)

- Assume a random surfer who either..
  - A1: Random walk (probability $1 - c$)
    - A1-1: Select a hyperedge e containing current node
    - A1-2: Move to a node v selected from the hyperedge e

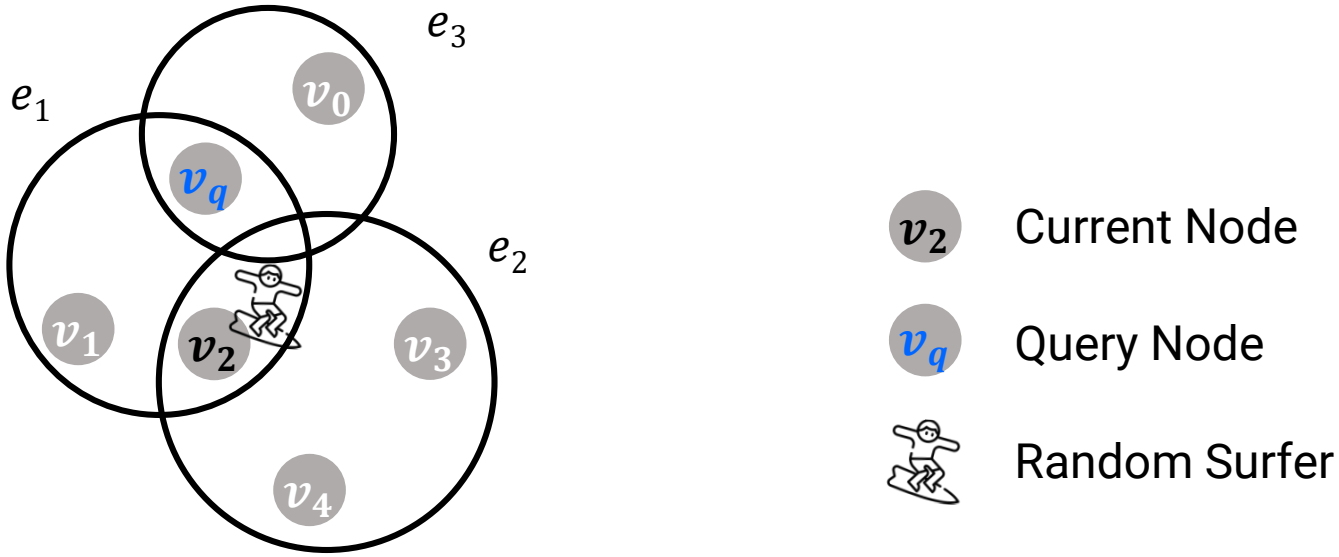


v_q Query Node

Random Surfer

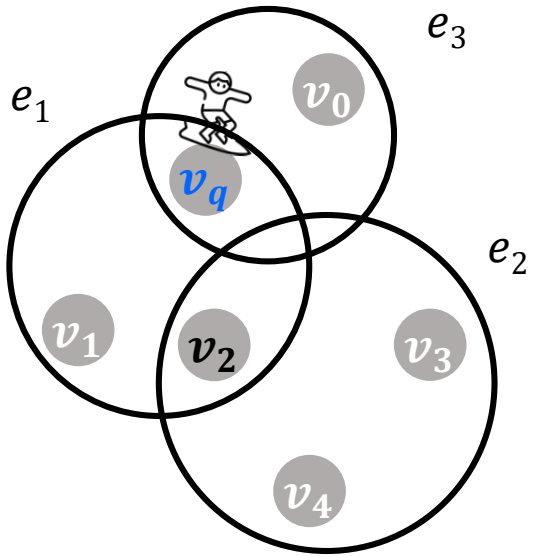
RWR on Hypergraphs: Definition (cont.)

- Assume a random surfer who either..
  - A1: Random walk (probability $1 - c$)
  - A2: Restart (probability c)
    Restart at the query node



RWR on Hypergraphs: Definition (cont.)

- Assume a random surfer who either..
  - A1: Random walk (probability $1 - c$)
  - A2: Restart (**probability c**)
    Restart at the query node

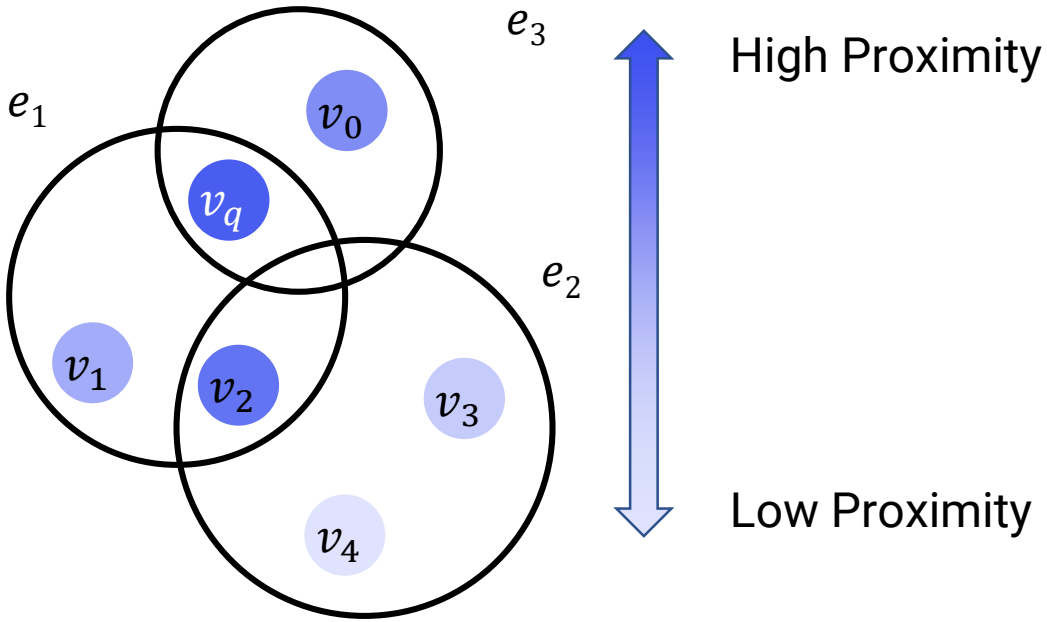


v_q Query Node

Random Surfer

RWR on Hypergraphs: Definition (cont.)

- RWR score is defined as the **stationary probability** over the nodes
- RWR score is interpreted as the proximity between nodes



Node	Probability
v_q	0.34
v_2	0.22
v_0	0.19
v_1	0.14
v_3	0.07
v_4	0.04

RWR on Hypergraphs: Challenges

- In many real-world scenarios, RWR scores need to be calculated for **multiple queries**

co-authorship example

Authors (Nodes)

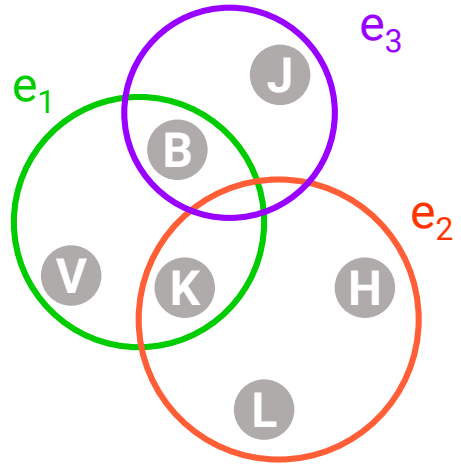
Austin Benson (**B**)	Daniel Huttenlocher (**H**)
Junteng Jia (**J**)	Jon Kleinberg (**K**)
Jure Leskovec (**L**)	Nate Veldt (**V**)

Publications (Hyperedges)

e_1: (V, B, K) NeruIPS'21

e_2: (L, H, K) WWW'10

e_3: (J, B) KDD'20



RWR on Hypergraphs: Challenges (cont.)

- In many real-world scenarios, RWR scores need to be calculated for **multiple queries**

*Which author is closest to **B**?*

Authors (Nodes)

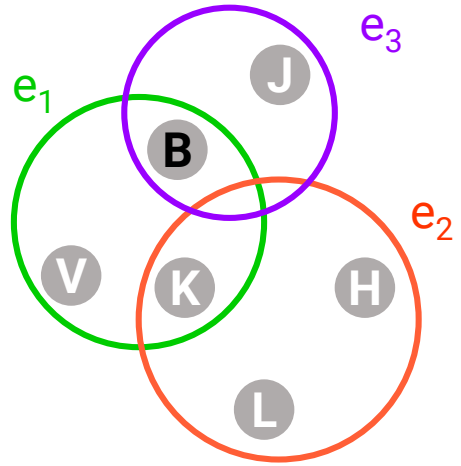
Austin Benson (**B**)	Daniel Huttenlocher (**H**)
Junteng Jia (**J**)	Jon Kleinberg (**K**)
Jure Leskovec (**L**)	Nate Veldt (**V**)

Publications (Hyperedges)

e_1: (V, B, K) NeruIPS'21

e_2: (L, H, K) WWW'10

e_3: (J, B) KDD'20



RWR on Hypergraphs: Challenges (cont.)

- In many real-world scenarios, RWR scores need to be calculated for **multiple queries**

*Which author is closest to **K**?*

Authors (Nodes)

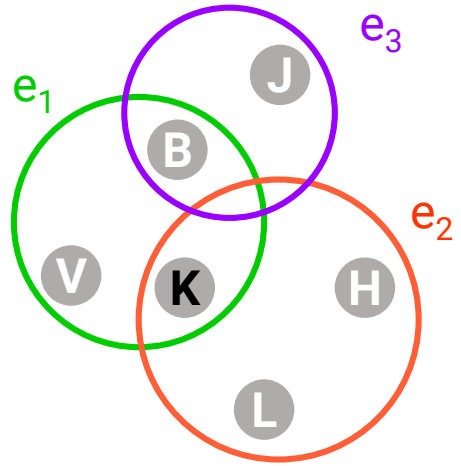
Austin Benson (**B**)	Daniel Huttenlocher (**H**)
Junteng Jia (**J**)	**Jon Kleinberg** (**K**)
Jure Leskovec (**L**)	Nate Veldt (**V**)

Publications (Hyperedges)

e_1: (V, B, K) NeruIPS'21

e_2: (L, H, K) WWW'10

e_3: (J, B) KDD'20



RWR on Hypergraphs: Challenges (cont.)

- In many real-world scenarios, RWR scores need to be calculated for **multiple queries**

- Q: Can we reduce the query cost?

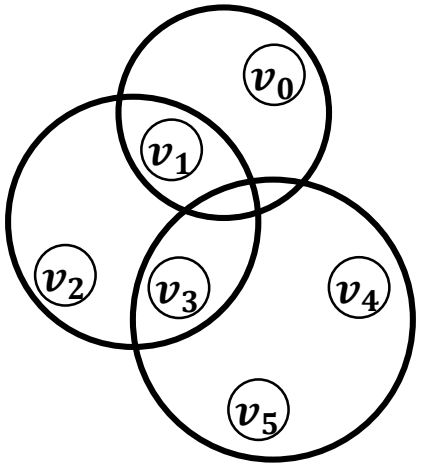
RWR on Hypergraphs: Challenges (cont.)

- In many real-world scenarios, RWR scores need to be calculated for **multiple queries**

- Q: Can we reduce the query cost?
- A: **Preprocessing method!**
  - Preprocessing methods for **graphs**
    - BEAR (SIGMOD'15)
    - BePI (SIGMOD'17)

Preprocessing Method

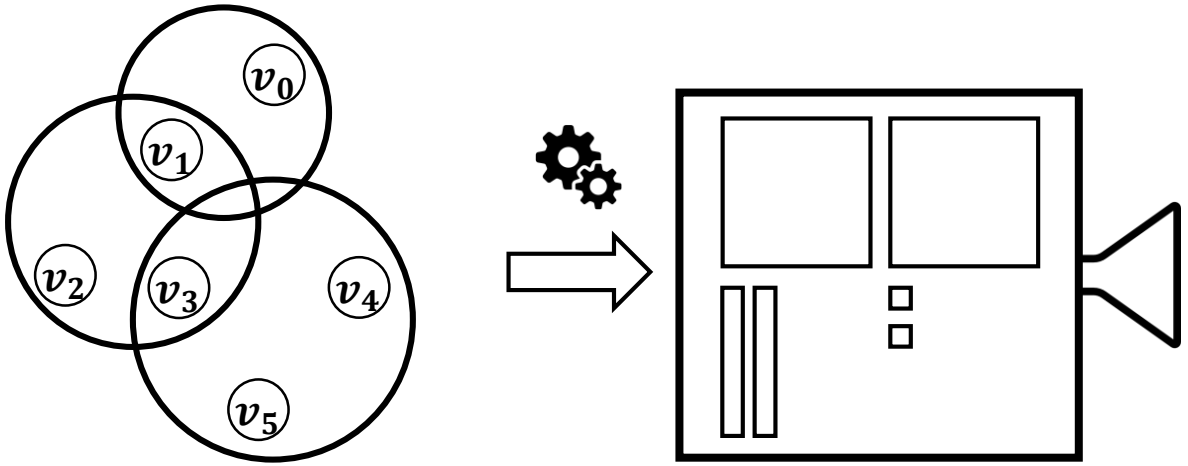
- Preprocess a given **hypergraph** into **parameters** for RWR
  - At the expense of preprocessing cost, lower the query cost



Hypergraph

Preprocessing Method (cont.)

- Preprocess a given **hypergraph** into **parameters** for RWR
  - At the expense of preprocessing cost, lower the query cost



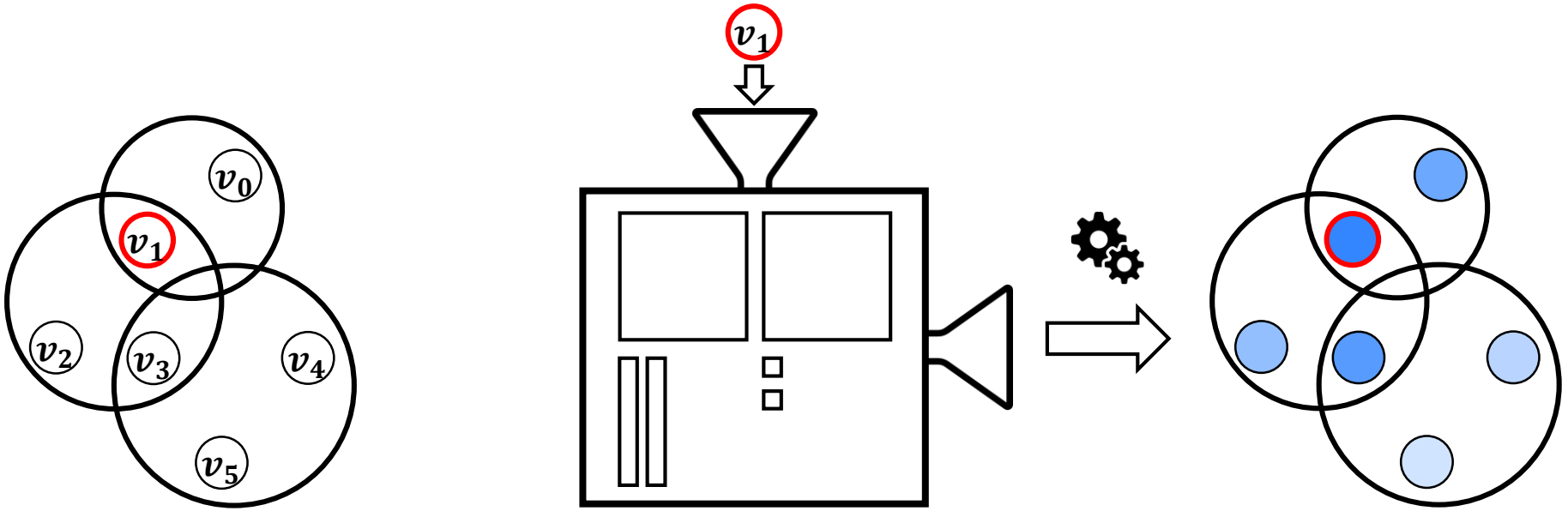
Hypergraph

Parameters

Preprocessing Method (cont.)

- Preprocess a given **hypergraph** into **parameters** for RWR
  - At the expense of preprocessing cost, lower the query cost

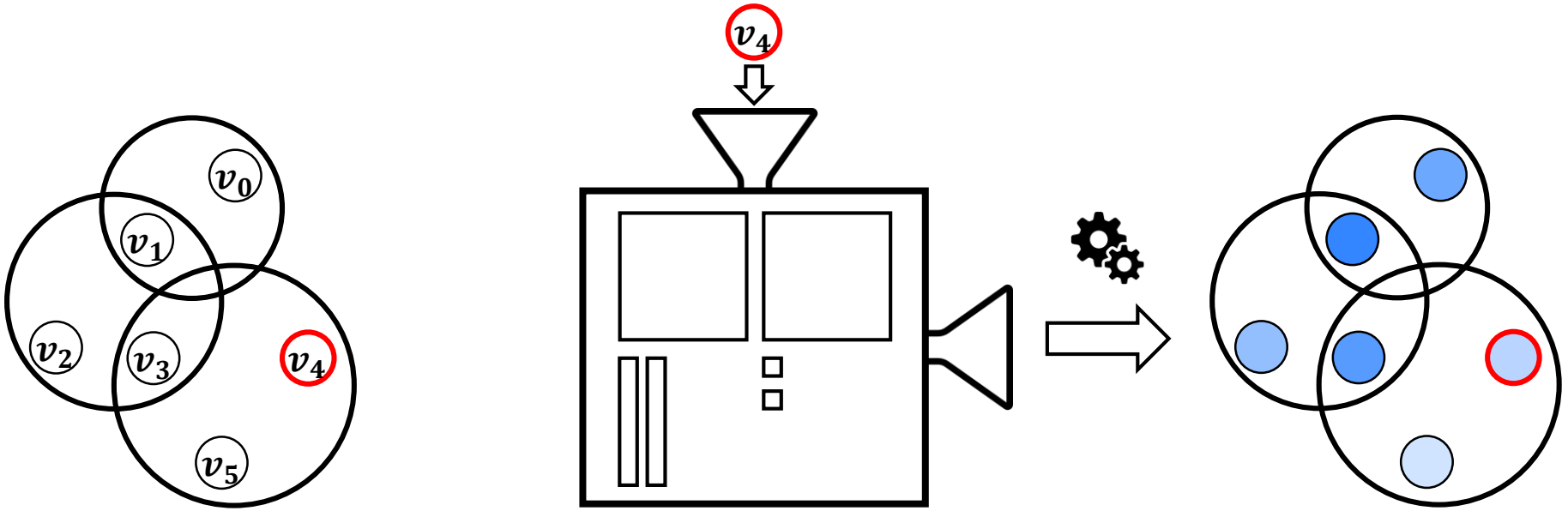
What is the proximity of nodes w.r.t. v_1*?*



Preprocessing Method (cont.)

- Preprocess a given **hypergraph** into **parameters** for RWR
  - At the expense of preprocessing cost, lower the query cost

What is the proximity of nodes w.r.t. v_4?



Preprocessing Method (cont.)

- Preprocess a given **hypergraph** into **parameters** for RWR
  - At the expense of preprocessing cost, lower the query cost

- There is no preprocessing method for RWR on **hypergraphs**

Our Goal

- Develop a **fast preprocessing method** for **RWR on Hypergraphs**

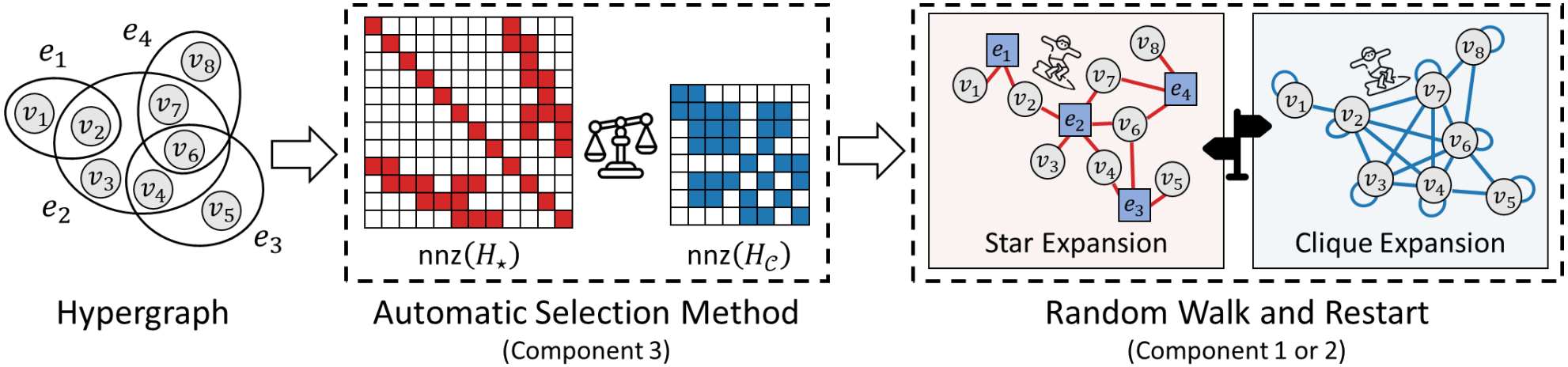
Roadmap

- Overview
- Random Walk with Restart (RWR) on Hypergraphs
- Proposed Algorithm: ARCHER <<
- Empirical Evaluation of ARCHER
- Application: Anomaly Detection
- Conclusion

Proposed Algorithm: ARCHER

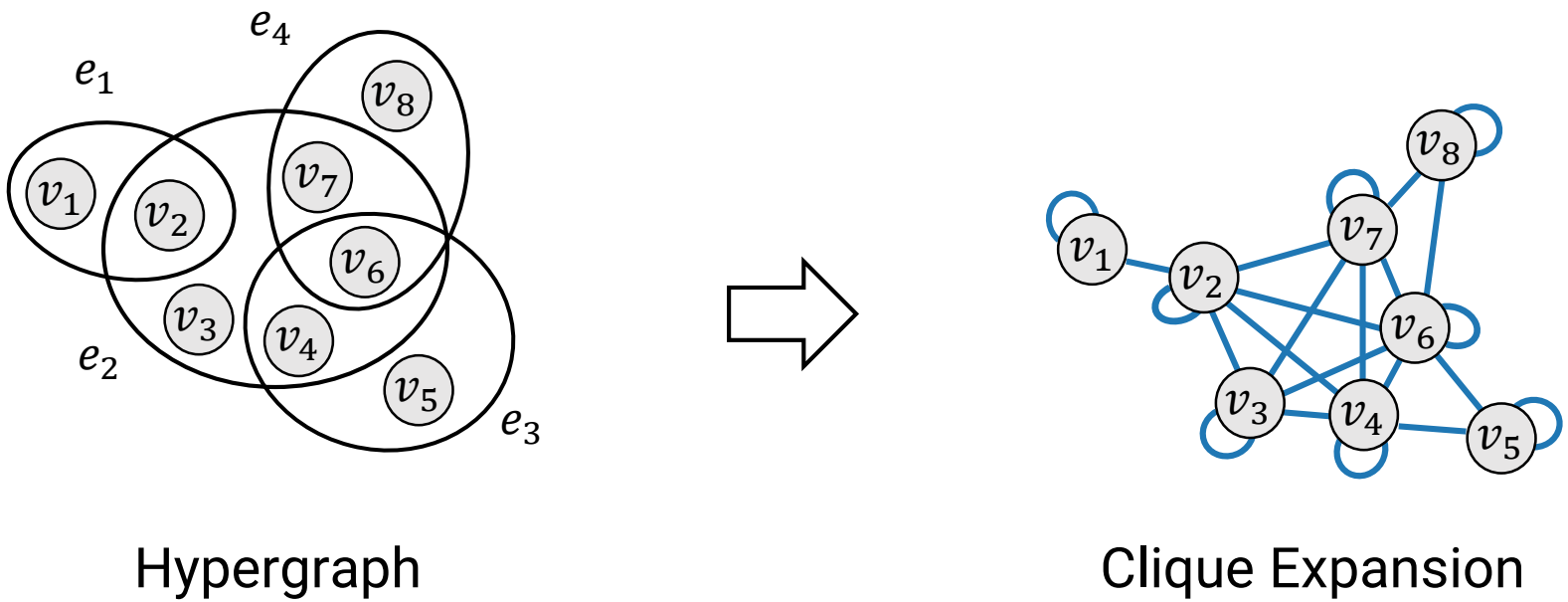
- We propose **ARCHER** (**A**daptive **R**WR **C**omputation on **Hyper**graphs)

- Component 1: Clique-expansion-based Method
- Component 2: Star-expansion-based Method
- Component 3: Automatic Selection Method



Comp. 1: Clique-expansion-based method

- Clique-expansion: a graph constructed from a hypergraph by replacing each original hyperedge with a clique

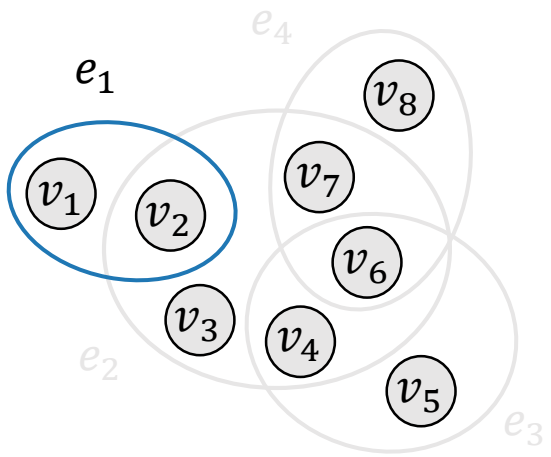


Hypergraph

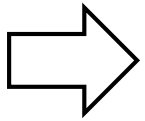
Clique Expansion

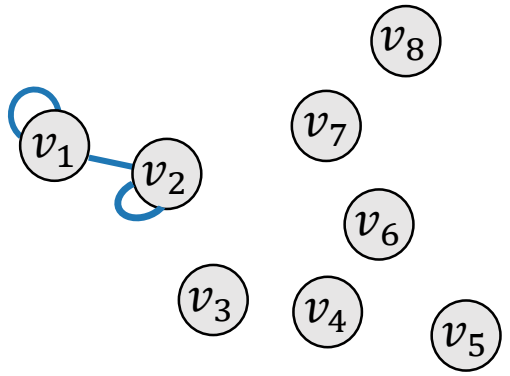
Comp. 1: Clique-expansion-based method (cont.)

- Clique-expansion: a graph constructed from a hypergraph by replacing each original hyperedge with a clique



Hypergraph

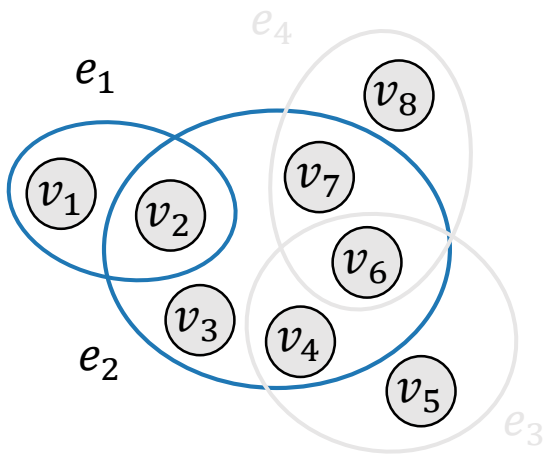




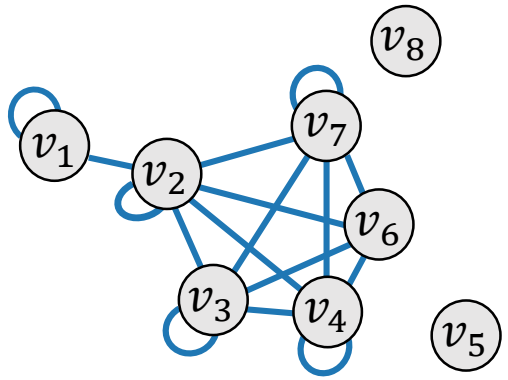
Clique Expansion

Comp. 1: Clique-expansion-based method (cont.)

- Clique-expansion: a graph constructed from a hypergraph by replacing each original hyperedge with a clique



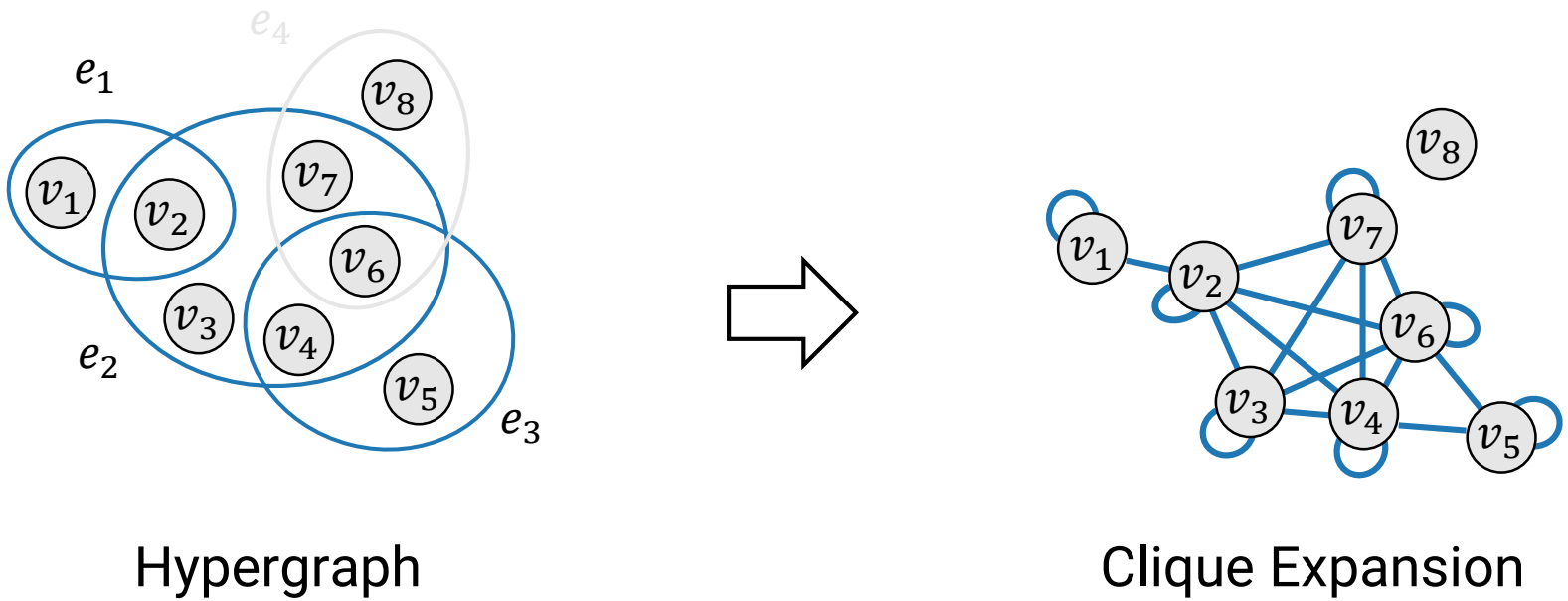
Hypergraph



Clique Expansion

Comp. 1: Clique-expansion-based method (cont.)

- Clique-expansion: a graph constructed from a hypergraph by replacing each original hyperedge with a clique

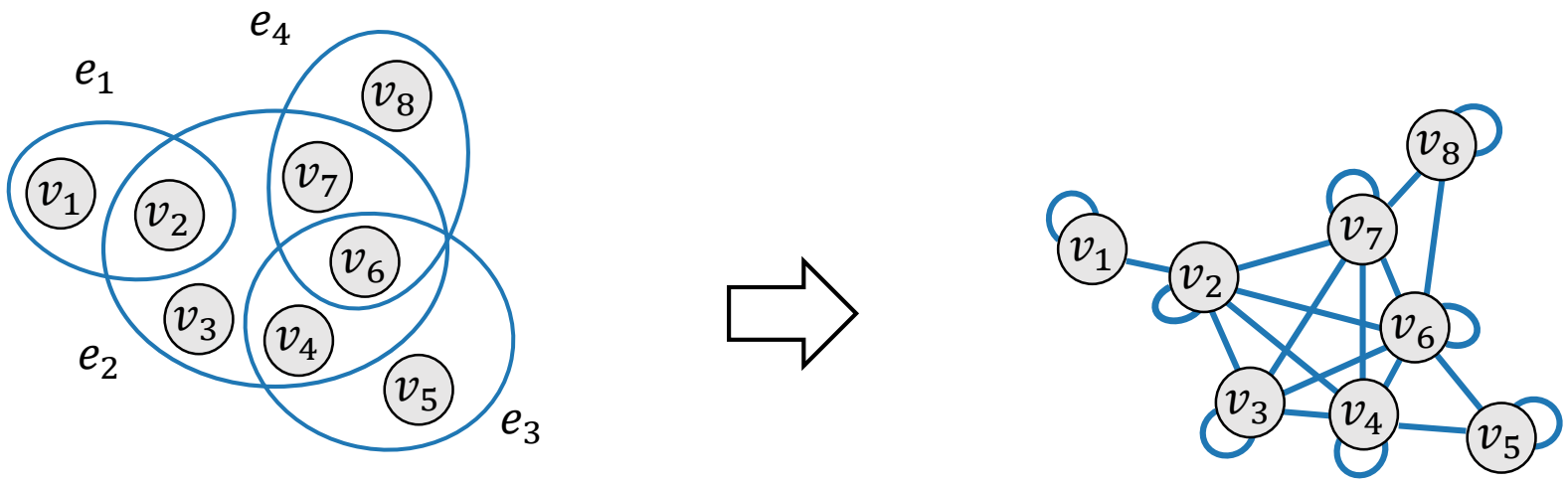


Hypergraph

Clique Expansion

Comp. 1: Clique-expansion-based method (cont.)

- Clique-expansion: a graph constructed from a hypergraph by replacing each original hyperedge with a clique

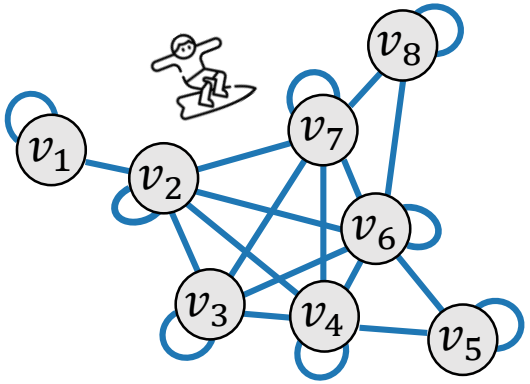


Hypergraph

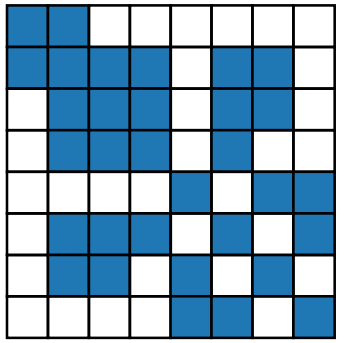
Clique Expansion

Comp. 1: Clique-expansion-based method (cont.)

- **Theorem (*Clique-Expansion Equality*):** RWR on a hypergraph can be reduced to RWR on its clique-expanded graph with proper edge weights



Clique Expansion



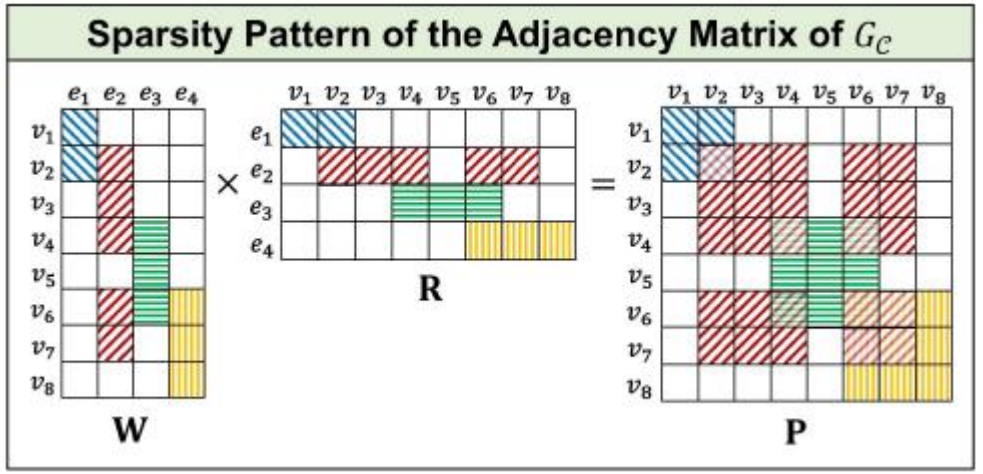
Adjacency Matrix of the Clique-Expanded Graph

Comp. 1: Clique-expansion-based method (cont.)

Details

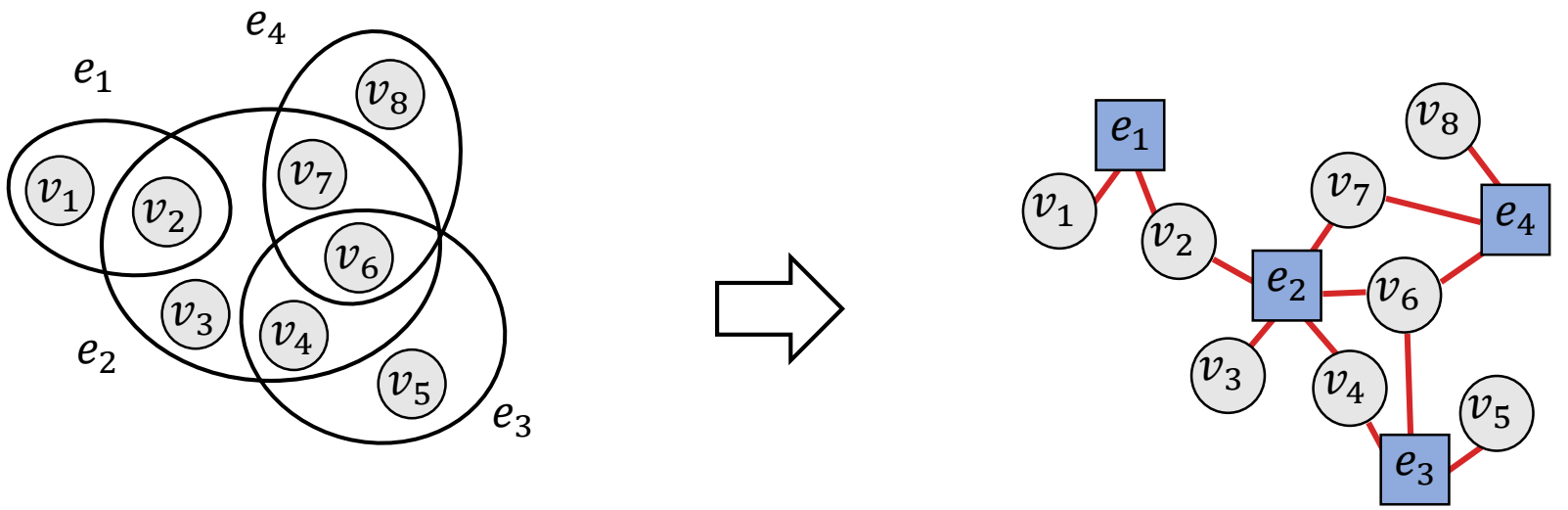
- **Theorem (*Clique-Expansion Equality*):** RWR on a hypergraph can be reduced to RWR on its clique-expanded graph with proper edge weights

- Input
  - Row-normalized hyperedge-weight matrix $\widetilde{\boldsymbol{W}}$
  - Row-normalized node-weight matrix $\widetilde{\boldsymbol{R}}$
  - RWR query vector $\boldsymbol{q}$
  - Restart probability c
- RWR score vector $\boldsymbol{r}$ can be obtained by solving
  - $\boldsymbol{r} = (1-c)\left(\widetilde{\boldsymbol{W}}\widetilde{\boldsymbol{R}}\right)^{\mathsf{T}}\boldsymbol{r} + c\boldsymbol{q}$



Comp. 2: Star-expansion-based method

- Star-expansion: a graph constructed from a hypergraph by
  - (1) aggregating nodes and hyperedges into new set of nodes
  - (2) adding edges between each incident node and hyperedge pair

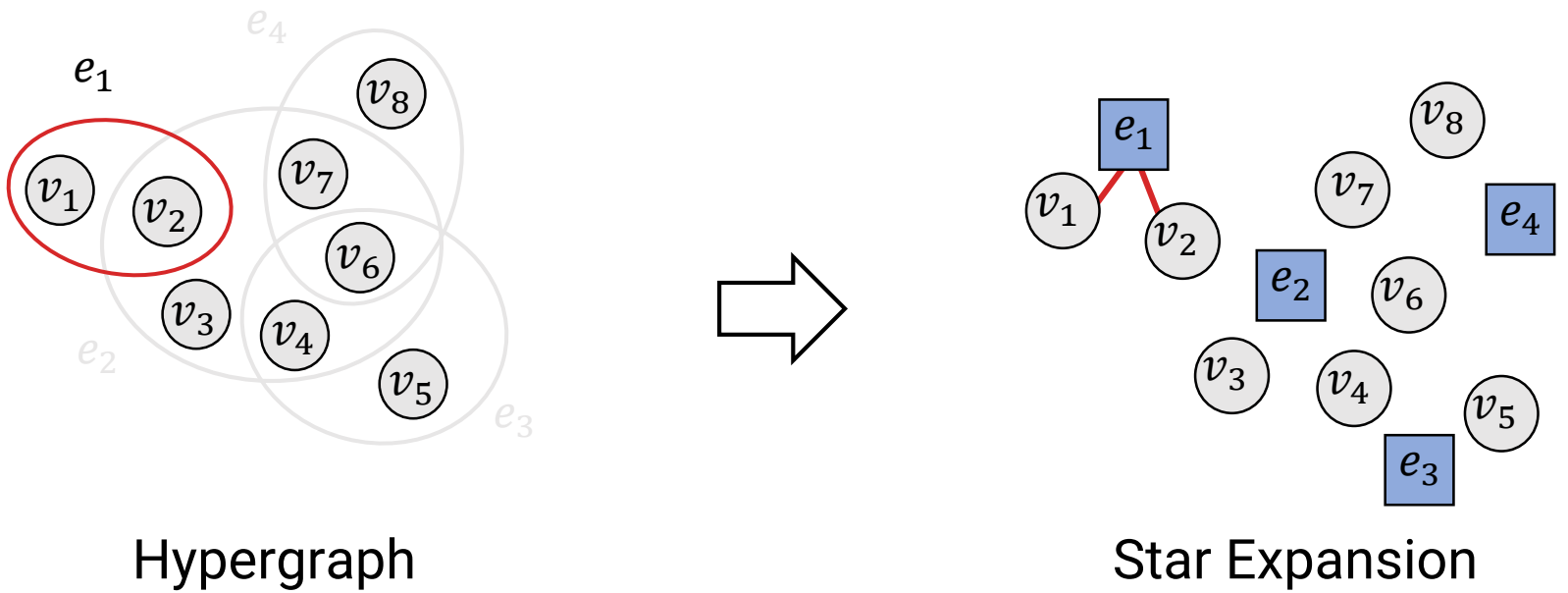


Hypergraph

Star Expansion

Comp. 2: Star-expansion-based method (cont.)

- Star-expansion: a graph constructed from a hypergraph by
  - (1) aggregating nodes and hyperedges into new set of nodes
  - (2) adding edges between each incident node and hyperedge pair

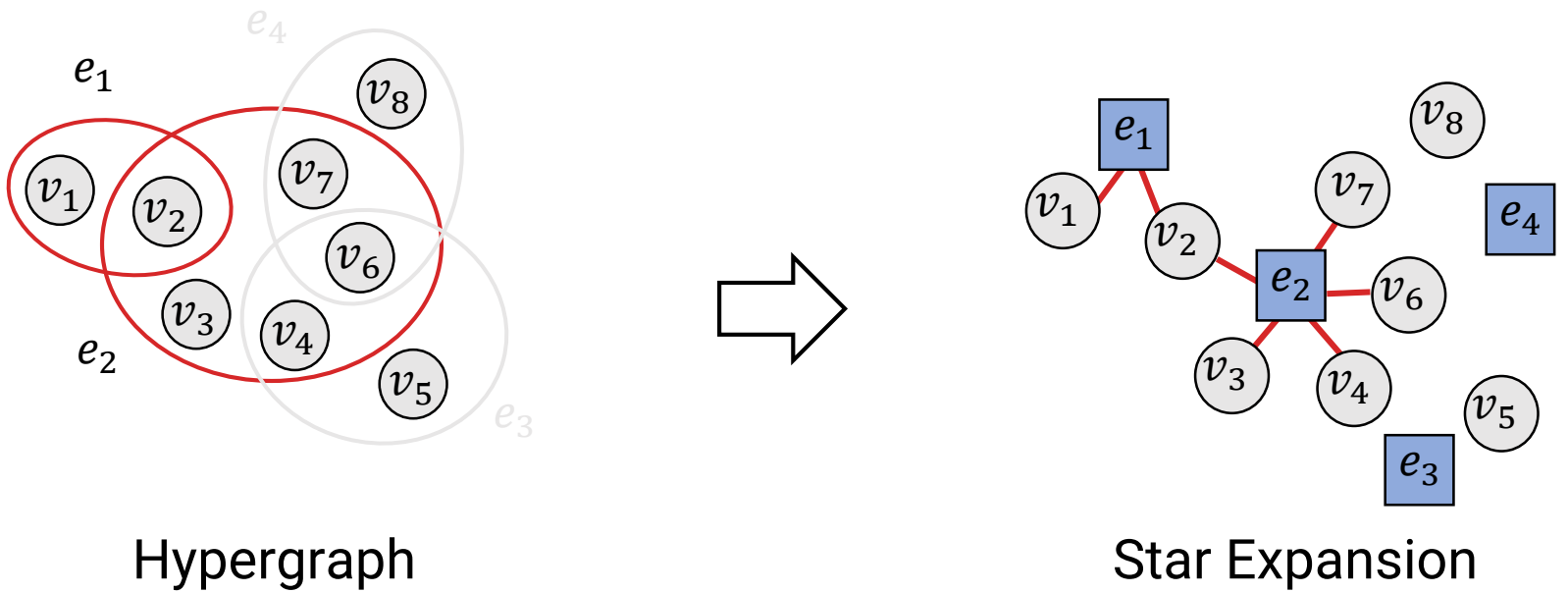


Hypergraph

Star Expansion

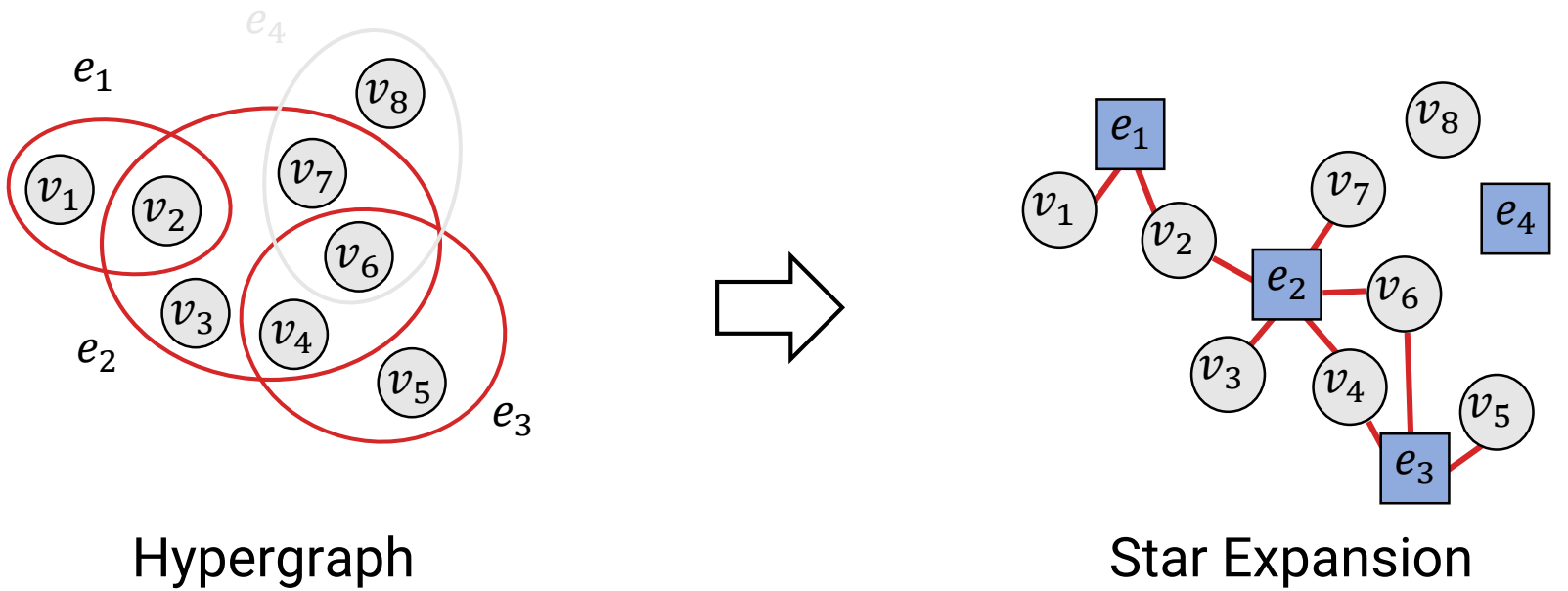
Comp. 2: Star-expansion-based method (cont.)

- Star-expansion: a graph constructed from a hypergraph by
  - (1) aggregating nodes and hyperedges into new set of nodes
  - (2) adding edges between each incident node and hyperedge pair



Comp. 2: Star-expansion-based method (cont.)

- Star-expansion: a graph constructed from a hypergraph by
  - (1) aggregating nodes and hyperedges into new set of nodes
  - (2) adding edges between each incident node and hyperedge pair

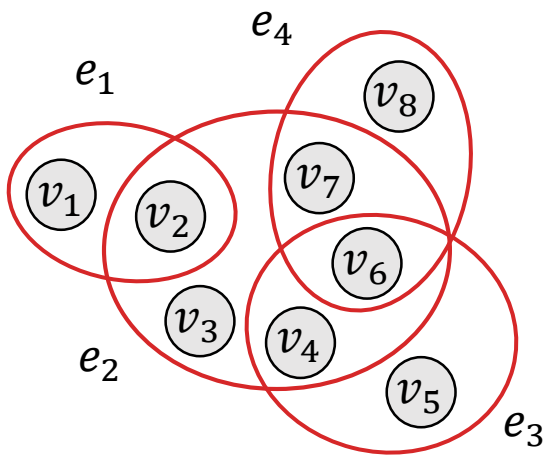


Hypergraph

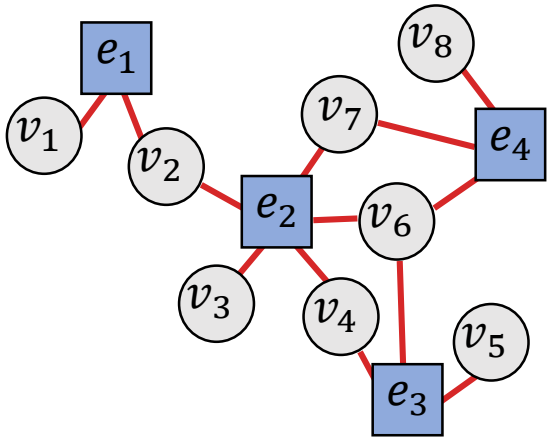
Star Expansion

Comp. 2: Star-expansion-based method (cont.)

- Star-expansion: a graph constructed from a hypergraph by
  - (1) aggregating nodes and hyperedges into new set of nodes
  - (2) adding edges between each incident node and hyperedge pair



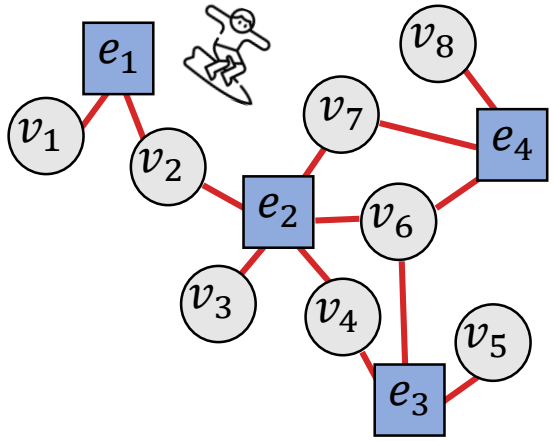
Hypergraph



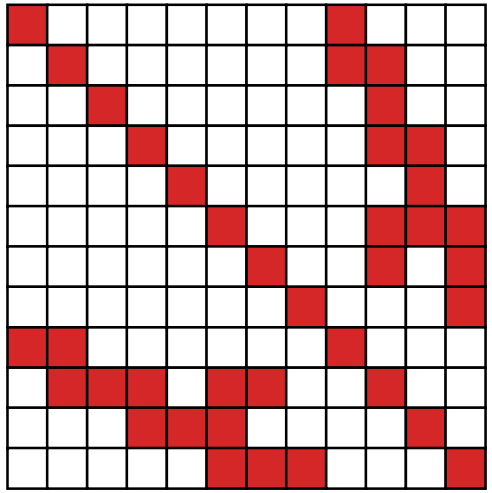
Star Expansion

Comp. 2: Star-expansion-based method (cont.)

- **Theorem (*Star-Expansion Equality*):** RWR on a hypergraph can be reduced to RWR on its star-expanded graph with proper edge weights



Star Expansion



Adjacency Matrix of the Star-Expanded Graph

Comp. 2: Star-expansion-based method (cont.)

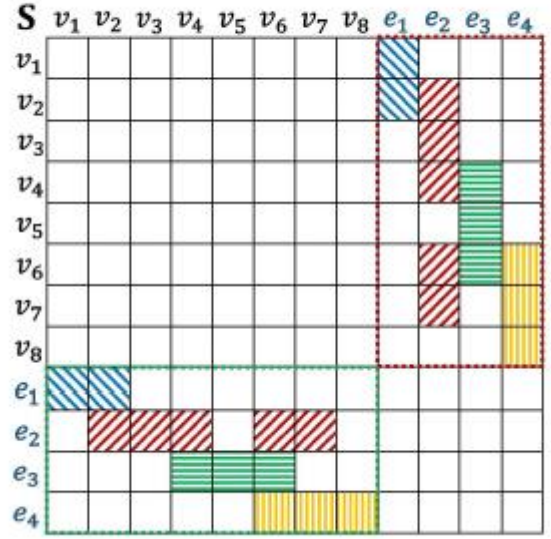
Details

- **Theorem (*Star-Expansion Equality*):** RWR on a hypergraph can be reduced to RWR on its star-expanded graph with proper edge weights

- Input
  - Row-normalized hyperedge-weight matrix $\widetilde{\boldsymbol{W}}$
  - Row-normalized node-weight matrix $\widetilde{\boldsymbol{R}}$
  - RWR query vector $\boldsymbol{q}$
  - Restart probability c
- RWR score vector $\boldsymbol{r}$ can be obtained by solving
  - $\boldsymbol{r}_\star = (1-c_\star)\left(\begin{bmatrix}\mathbf{0} & \widetilde{\boldsymbol{W}} \\ \widetilde{\boldsymbol{R}} & \mathbf{0}\end{bmatrix}\right)^{\mathsf{T}} \boldsymbol{r}_\star + c_\star \begin{bmatrix}\boldsymbol{q} \\ \mathbf{0}\end{bmatrix}$,

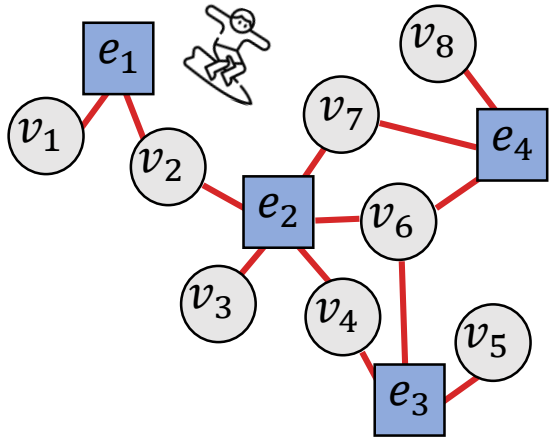
    $\boldsymbol{r} = \frac{c}{c_\star}\boldsymbol{r}_\star[1:|V|]\ \left(c_\star = 1-\sqrt{1-c}\right)$

Sparsity Pattern of the Adjacency Matrix of $G_\star$

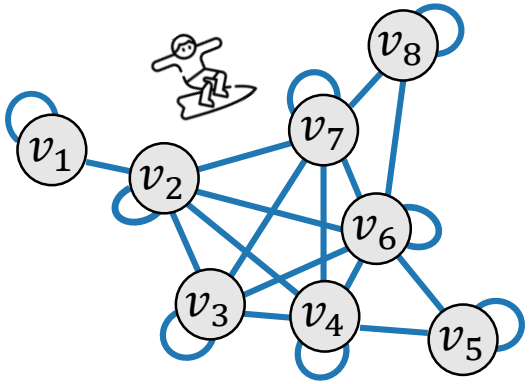


Comp. 3: Automatic selection method

- How can we choose between clique- and star-expansion?



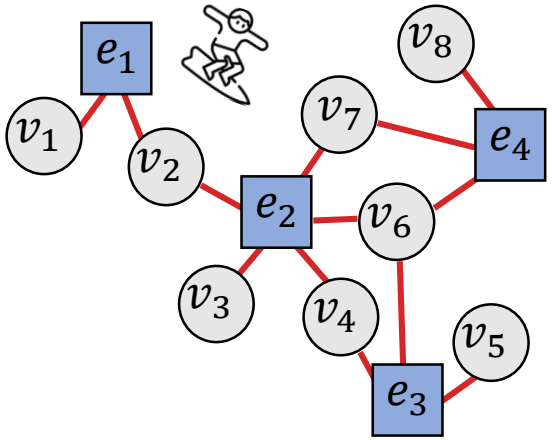
Star Expansion



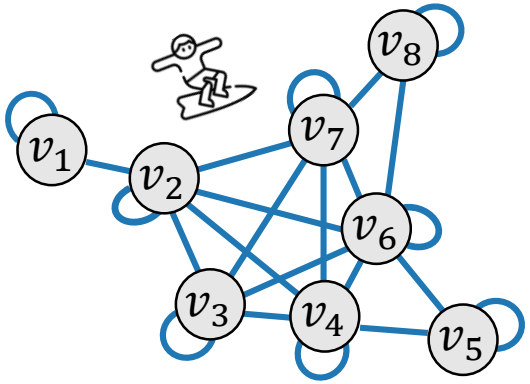
Clique Expansion

Comp. 3: Automatic selection method (cont.)

- How can we choose between clique- and star-expansion?
- Naïve answer: preprocess both, choose after running



Star Expansion



Clique Expansion

Comp. 3: Automatic selection method (cont.)

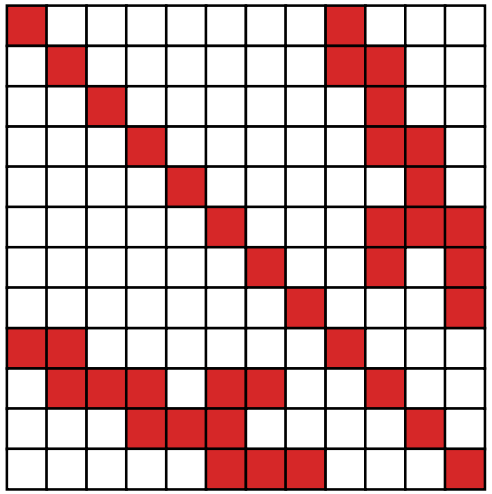
- How can we choose between clique- and star-expansion?
- Naïve answer: preprocess both, choose after running
- Cost of preprocessing vary depending on the dataset
  - Empirical results
    - Case 1: star-expansion-based method is efficient
      - Up to 137.6x less time
      - Up to 16.2x less space
    - Case 2: clique-expansion-based method is efficient
      - Up to 6.4x less time
      - Up to 9.6x less space

*Is there a way to choose **without actually executing them**?*

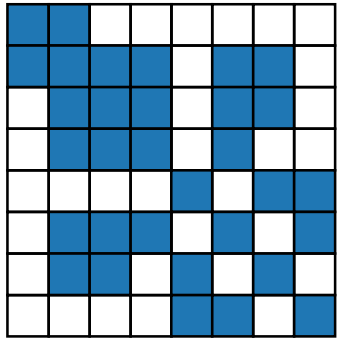
Comp. 3: Automatic selection method (cont.)

- ***Hint***: empirically preprocessing cost depends on **the count of non-zeros in the adjacency matrix**

Star Expansion



Clique Expansion



Comp. 3: Automatic selection method (cont.)

- Computing the **non-zero count** is much lighter than preprocessing!

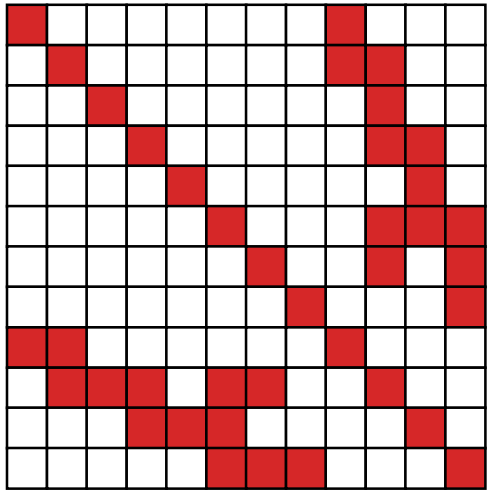
- Computing cost of **counting non-zeros**
  - Star expansion
    - Time complexity: $O(1)$
    - Space complexity: $O(1)$
  - Clique expansion
    - Time complexity: $O(\sum_{e \in E} |e|^2)$
    - Space complexity: $O(n)$

※Worst case preprocessing cost※
- Star expansion
  - Time: $O\left((|V| + |E|)^3\right)$
  - Space: $O\left((|V| + |E|)^2\right)$
- Clique expansion
  - Time: $O(|V|^3)$
  - Space: $O(|V|^2)$

Comp. 3: Automatic selection method (cont.)

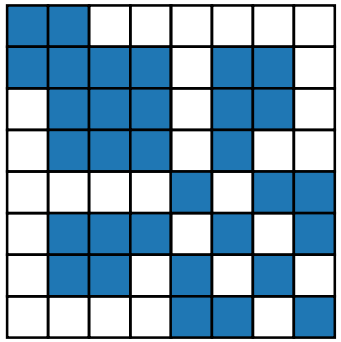
- Choose the method with **less non-zeros**

Star Expansion



nonzero(star)

Clique Expansion

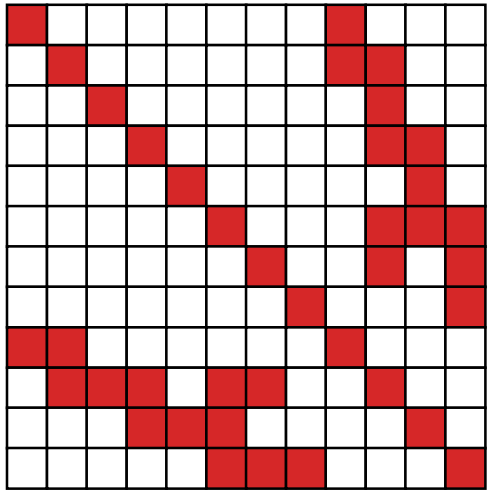


nonzero(clique)

Comp. 3: Automatic selection method (cont.)

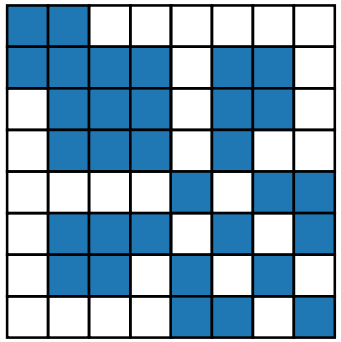
- Choose the method with **less non-zeros**

Star Expansion

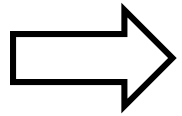


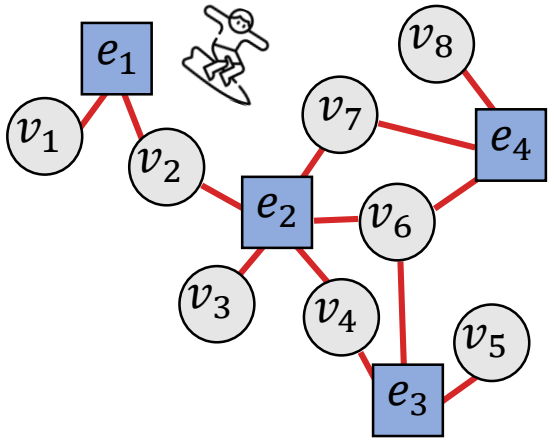
nonzero(star) <

Clique Expansion



nonzero(clique)



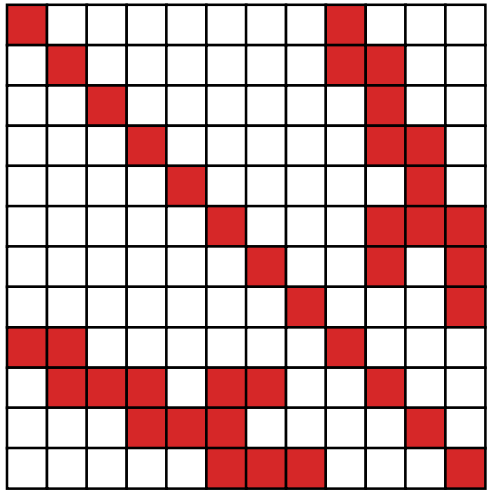


Star Expansion

Comp. 3: Automatic selection method (cont.)

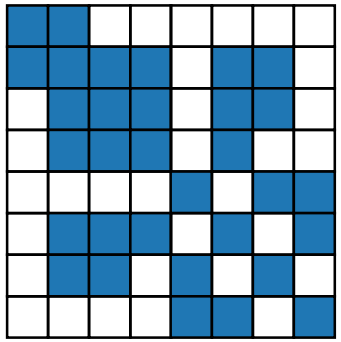
- Choose the method with **less non-zeros**

Star Expansion

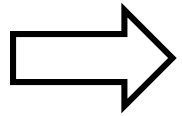


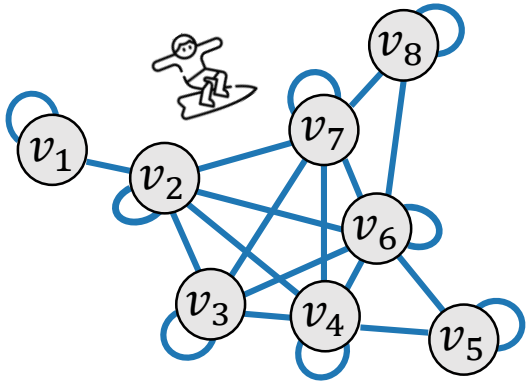
nonzero(star) >

Clique Expansion



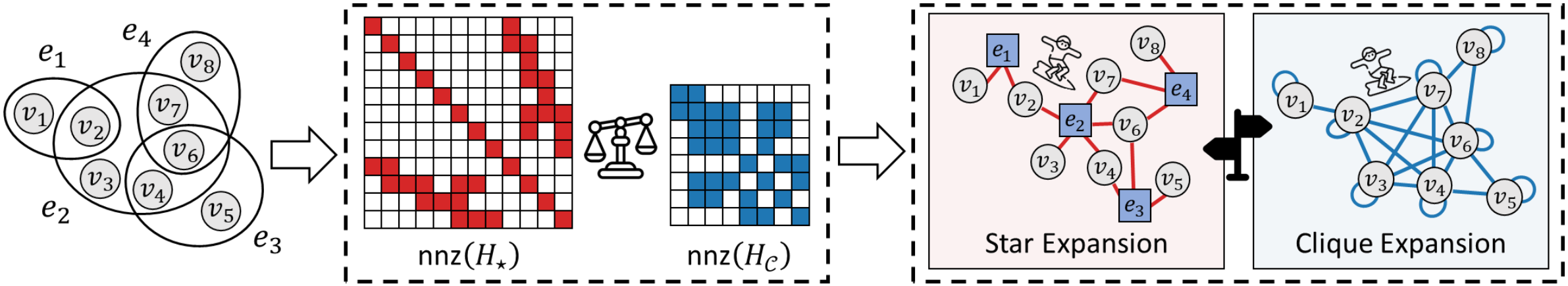
nonzero(clique)





Clique Expansion

Ultimate Framework: ARCHER



Hypergraph

Automatic Selection Method
(Component 3)

Random Walk and Restart
(Component 1 or 2)

Roadmap

- Overview
- Random Walk with Restart (RWR) on Hypergraphs
- Proposed Algorithm: ARCHER
- Empirical Evaluation of ARCHER <<
- Application: Anomaly Detection
- Conclusion

Experimental Settings

- Datasets: 18 real-world hypergraphs from 10 different domains
  - # nodes: 143 – 2.7M
  - # hyperedges: 6,038 – 11M

Dataset	n	m	$\text{avg}_{e\in\mathcal{E}}\lvert e\rvert$	$\max_{e\in\mathcal{E}}\lvert e\rvert$	Density	Overlapness
EEN	143	10,883	2.47	37	76.12	188.21
EEU	998	234,760	2.39	40	234.09	559.80
SB	294	29,157	7.96	99	99.17	789.62
HB	1,494	60,987	20.47	399	40.82	835.79
WAL	88,860	69,906	6.59	25	0.79	5.81
TRI	172,738	233,202	3.12	85	1.35	4.21
AM	55,700	105,655	8.12	555	1.90	15.41
YP	25,252	25,656	18.2	649	1.02	18.50
TW	81,305	70,097	25.2	1,205	0.86	21.75
COH	1,014,734	1,812,511	1.32	925	1.75	2.32
COG	1,256,385	1,590,335	2.80	284	1.26	3.53
COD	1,924,991	3,700,067	2.79	280	1.92	5.35
THU	125,602	192,947	1.80	14	1.54	2.76
THM	176,445	719,792	2.24	21	4.08	9.13
THS	2,675,955	11,305,343	2.23	67	4.22	9.56
ML1	3,533	6,038	95.3	1,435	1.71	162.83
ML10	10,472	69,816	84.3	3,375	6.67	562.02
ML20	22,884	138,362	88.1	4,168	6.05	532.93

- **Email**: EEN, EEU
- **Co-sponsorship of bills**: SB, HB
- **Co-purchased products**: WAL
- **“Click-out” accommodation set**: TRI
- **Co-authorship**: COD, COG, COH
- **User group of thread**: THS, THM, THU
- **Product review**: AM
- **Location review**: YP
- **Group on Twitter**: TW
- **Movie review**: ML1, ML10, ML20

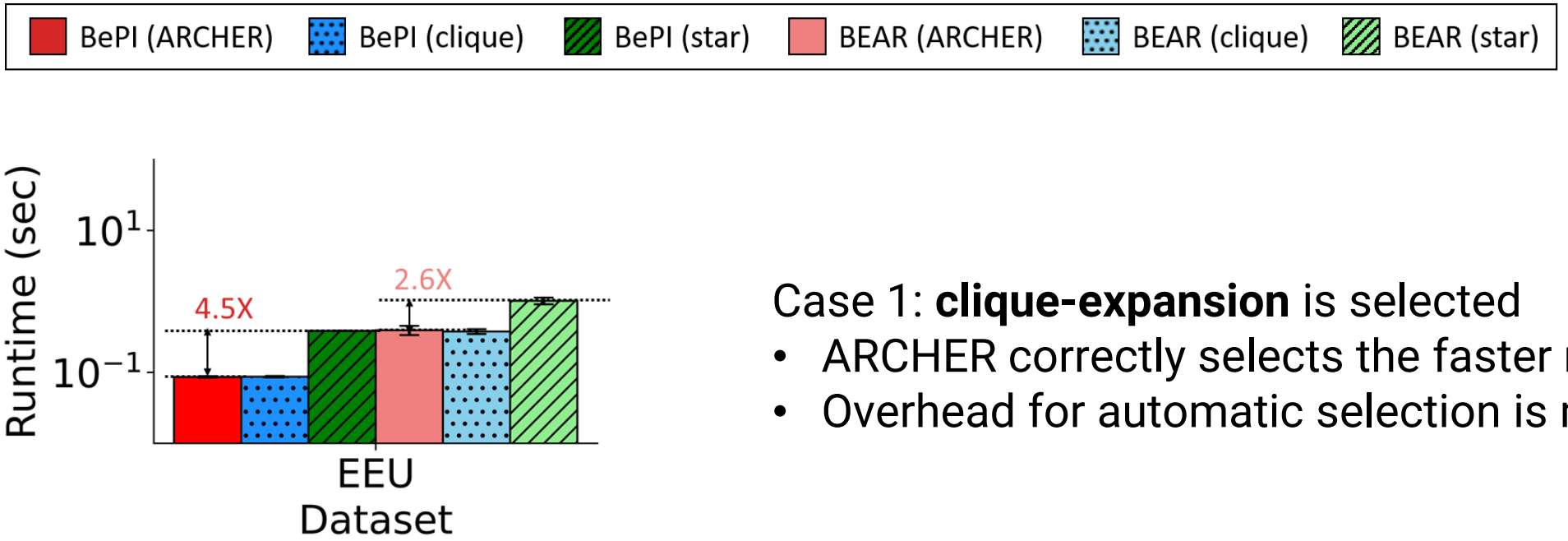
Experimental Settings (cont.)

- RWR computation methods for hypergraphs:
  - ARCHER
  - Always clique-expansion-based method
  - Always star-expansion-based method

- RWR computation methods for clique- and star-expanded graphs:
  - Preprocessing methods
    - BEAR (SIGMOD'15)
    - BePI (SIGMOD'17)
  - Power iteration

Q1. Preprocessing Time

- **Q1.** How **long** do ARCHER and the baselines take for preprocessing?

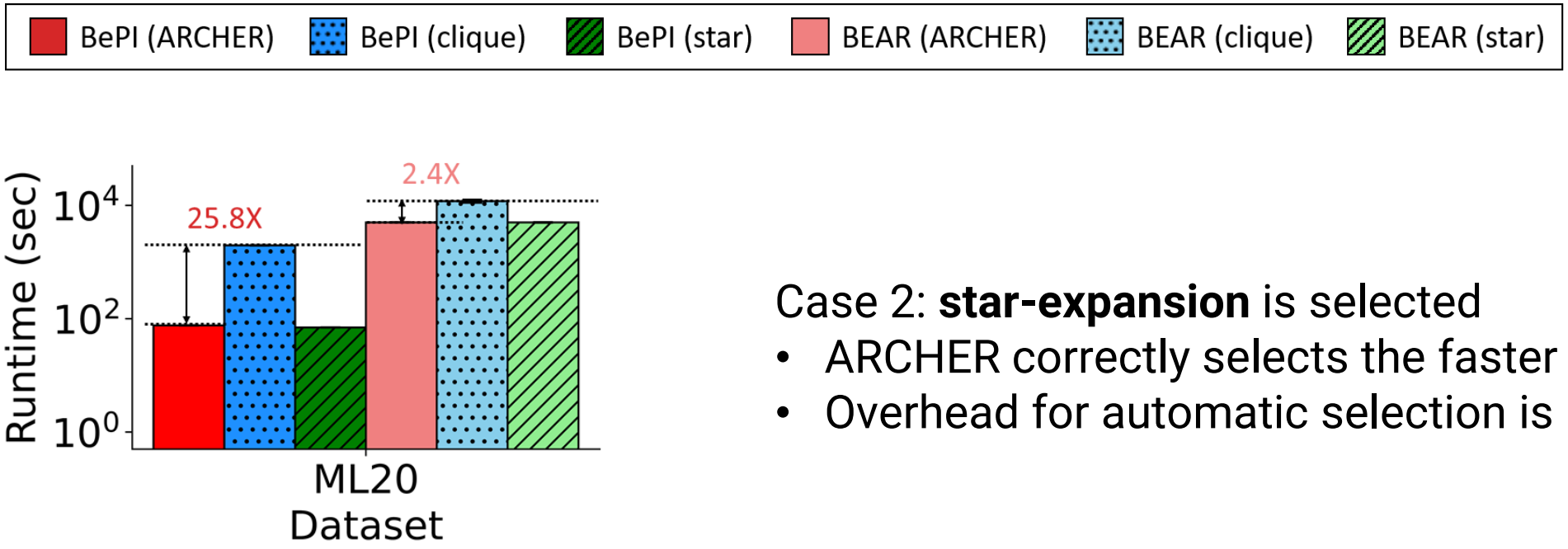


Case 1: **clique-expansion** is selected

- ARCHER correctly selects the faster method
- Overhead for automatic selection is marginal

Q1. Preprocessing Time (cont.)

- **Q1.** How **long** do ARCHER and the baselines take for preprocessing?

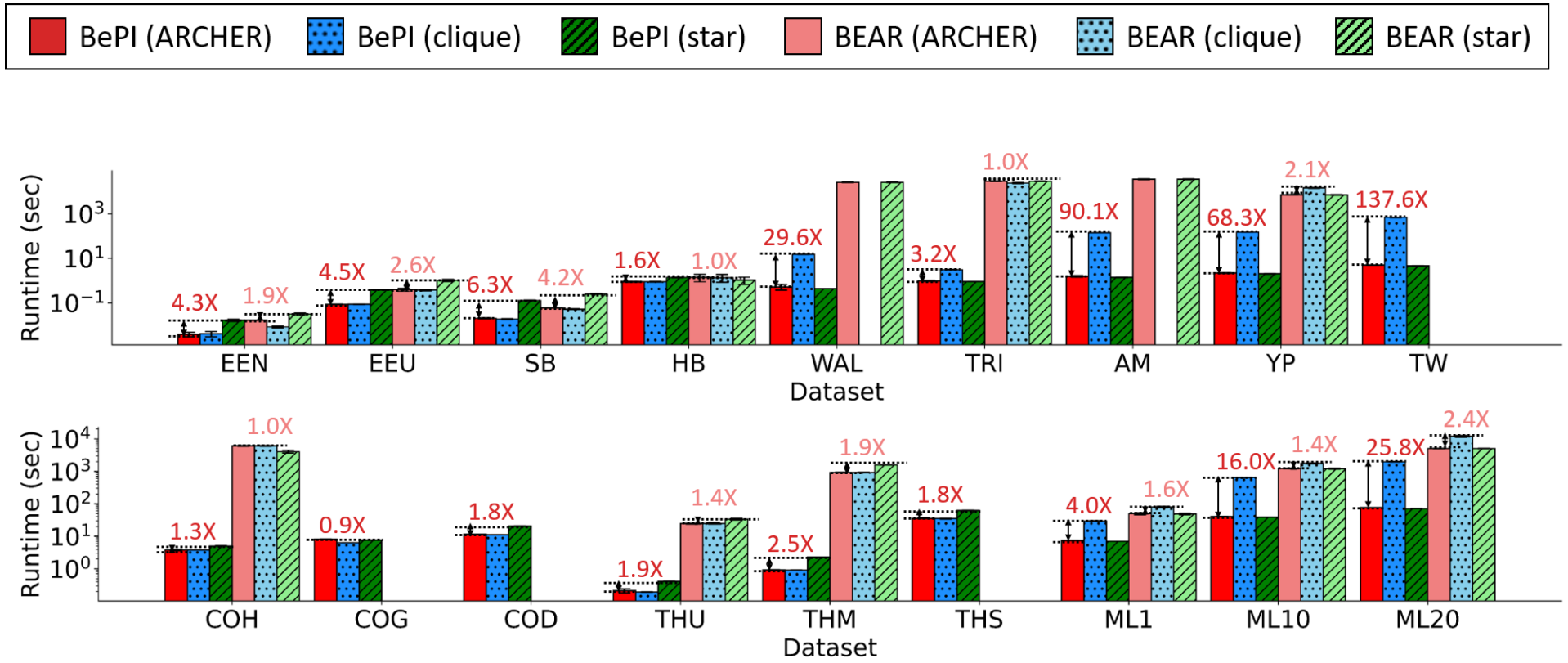


Case 2: **star-expansion** is selected

- ARCHER correctly selects the faster method
- Overhead for automatic selection is marginal

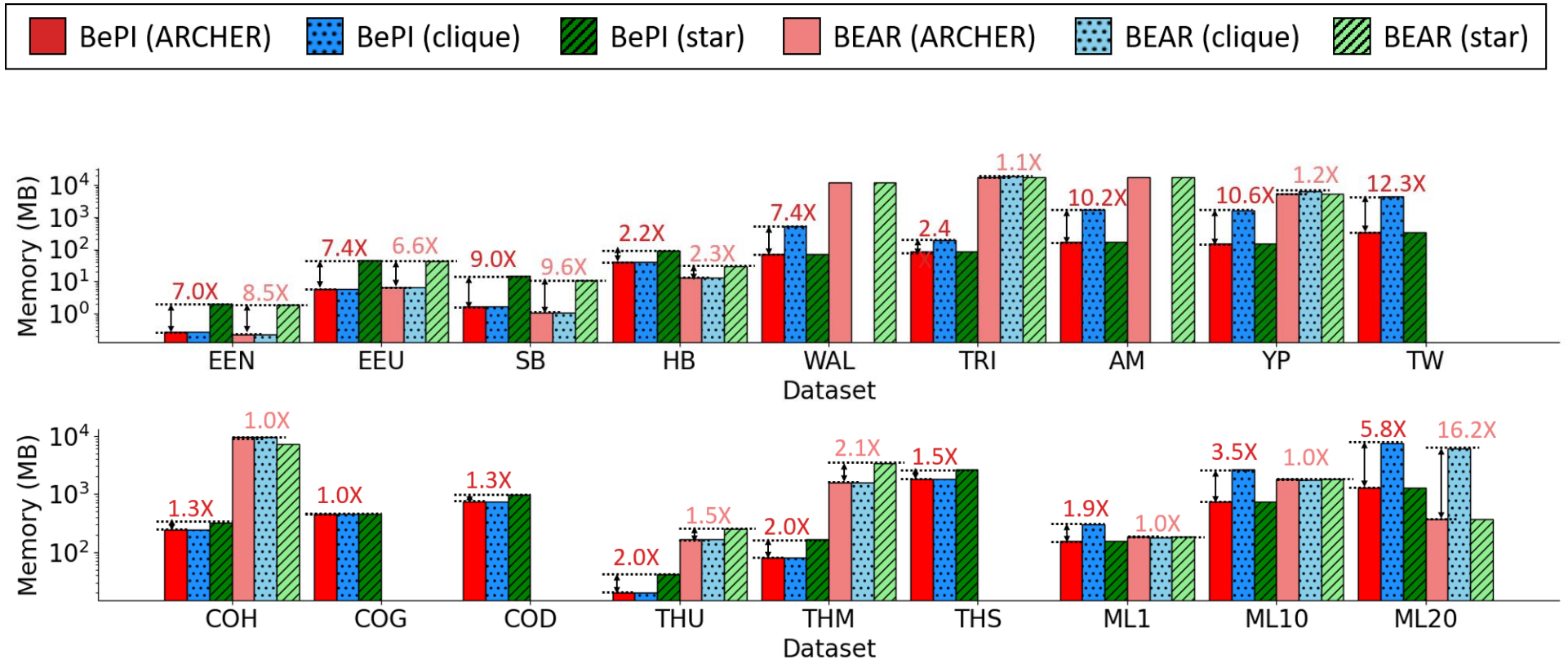
Q1. Preprocessing Time (cont.)

- **Q1.** How **long** do ARCHER and the baselines take for preprocessing?
- **A1.** ARCHER takes up to **137.6x** less time



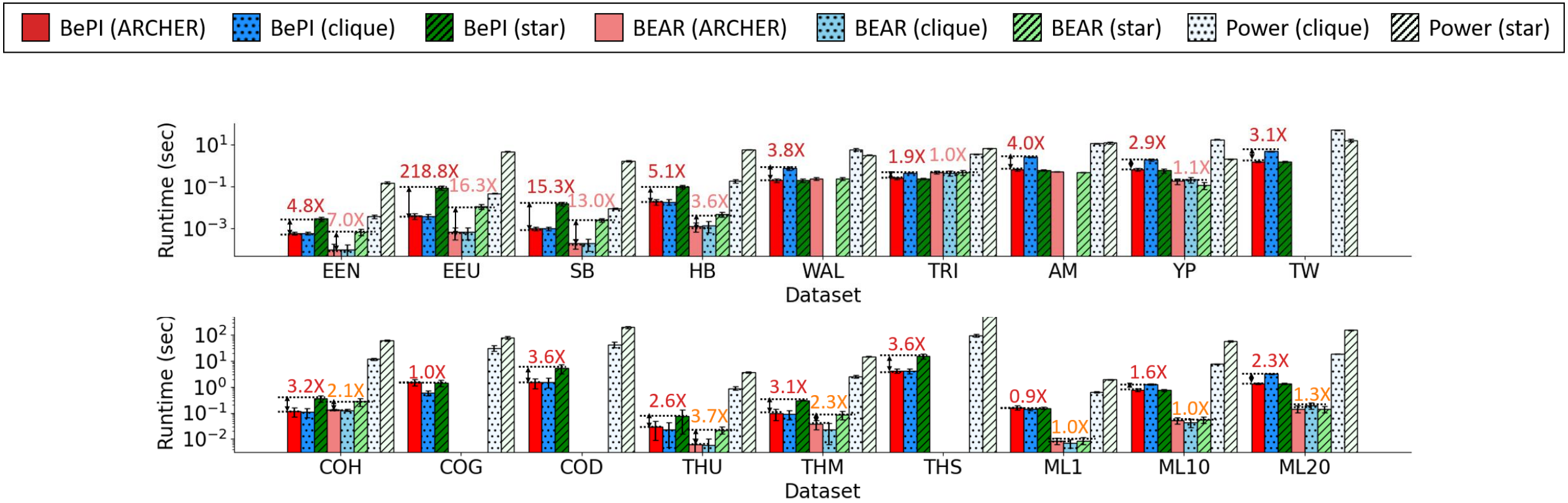
Q2. Space Cost

- **Q2.** How much **memory space** do they take for preprocessing?
- **A2.** ARCHER takes up to **16.2x** less space



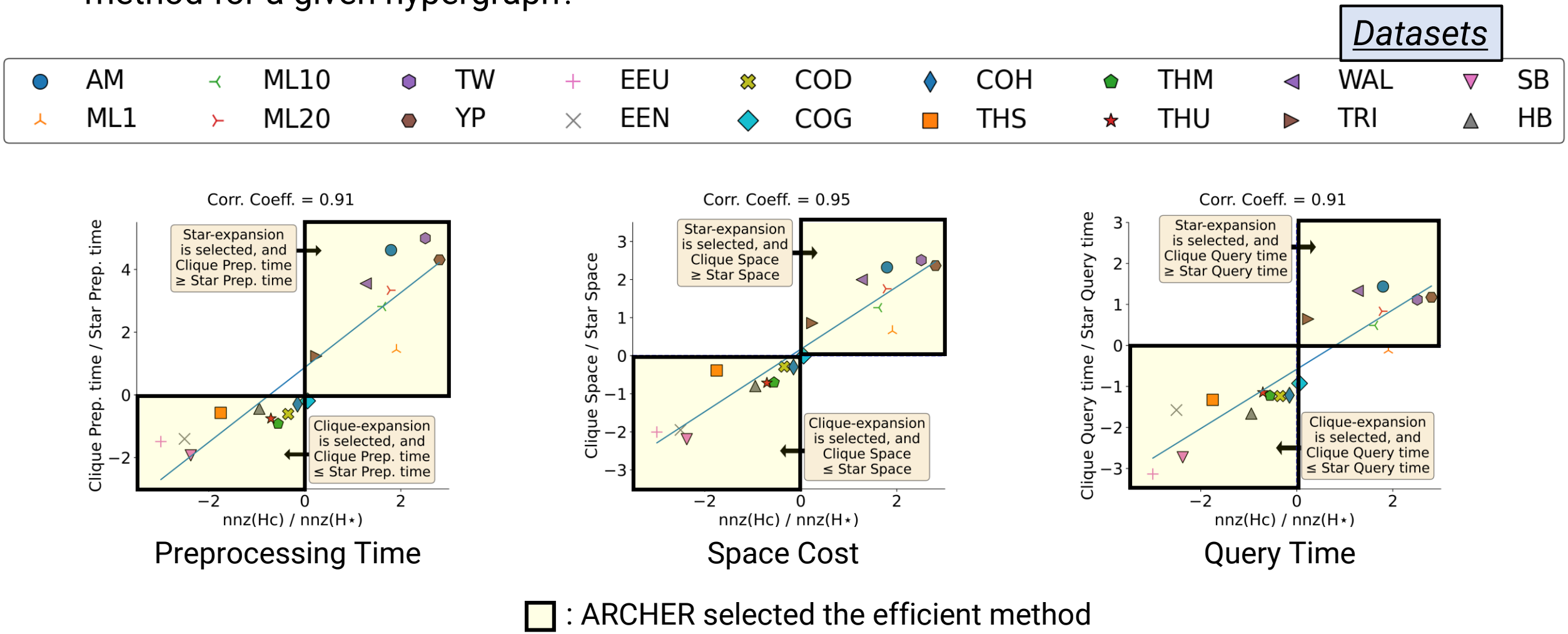
Q3. Query Time

- **Q3.** How quickly do they process RWR queries?
- **A3.** ARCHER takes up to **218.8x** less time



Q4. Automatic Selection Method

- **Q4.** How precisely does our automatic selection strategy decide more efficient computation method for a given hypergraph?



Q4. Automatic Selection Method (cont.)

- **Q4.** How precisely does our automatic selection strategy decide more efficient computation method for a given hypergraph?

- We compare proposed non-zero ratio with 3 baselines

Measure	Accuracy (18 datasets)
Non-zero ratio (proposed)	**0.944**
Density: $\|E\|/\|V\|$	0.722
Overlapness: $\sum_{e\in E}\|e\|/\|V\|$	0.667
Average hyperedge size: $\sum_{e\in E}\|e\|/\|E\|$	0.889

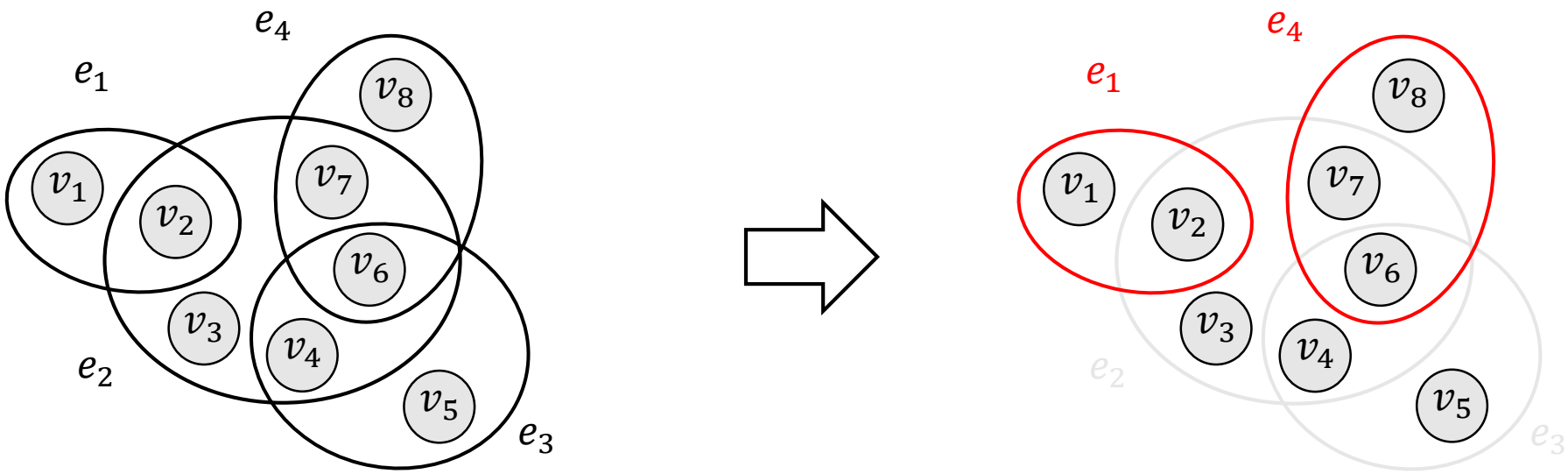
Roadmap

- Overview
- Random Walk with Restart (RWR) on Hypergraphs
- Proposed Algorithm: ARCHER
- Empirical Evaluation of ARCHER
- Application: Anomaly Detection <<
- Conclusion

Application: Anomaly Detection

Task Description

- Given a hypergraph, detect **anomalous** hyperedges that deviate from ordinary hyperedges



Hypergraph

Anomaly Detection

Application: Anomaly Detection (cont.)

- We expanded anomaly detection on bipartite graphs (ICDM'05) to hypergraphs
- Intuition: if a hyperedge e is **normal, proximity between the nodes** in e should be high
- Normality score

$$ns(e) = \frac{1}{|e|(|e|-1)} \sum_{u \in e} \sum_{v \in e \setminus \{u\}} \boldsymbol{r}_{u \to v}$$

$\boldsymbol{r}_{u \to v}$: RWR score of node v w.r.t. the query node u

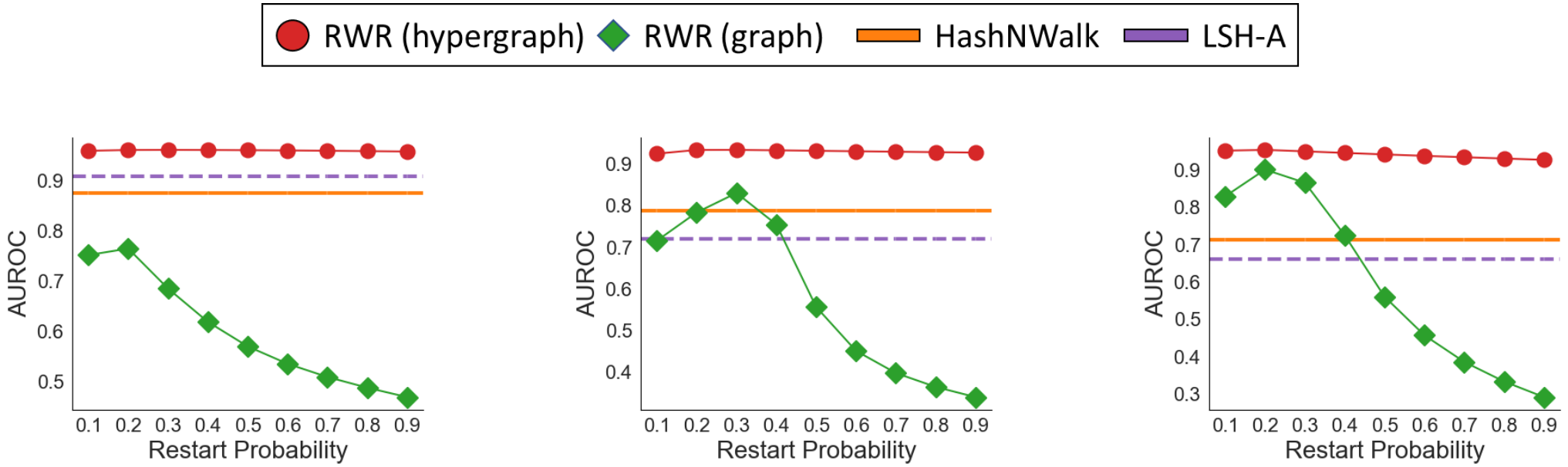
- **ARCHER** accelerates the computation of the normality score

Application: Anomaly Detection (cont.)

- Experimental Settings
- Datasets: EEN (email-Enron), SB (senate-bills), HB (house-bills)
- Anomalous hyperedge injection: unexpected hyperedges (IJCAI’22)
- Methods
  - RWR on hypergraphs
  - RWR on (pair-wise) graphs
  - LSH-A (COMPLEX NETWORKS’17)
  - HashNWalk (IJCAI’22)

Application: Anomaly Detection (cont.)

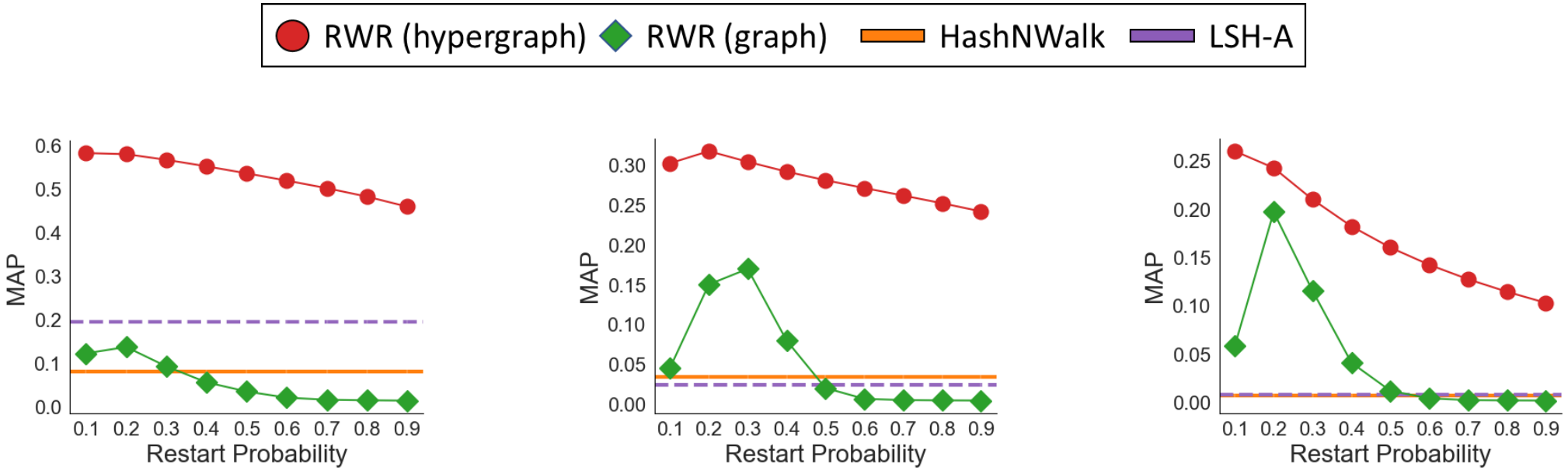
- **Q5.** Can we achieve more accurate anomaly detection on hypergraphs using RWR scores, compared to existing approaches?



AUROC in EEN | AUROC in SB | AUROC in HB

Application: Anomaly Detection (cont.)

- **Q5.** Can we achieve more accurate anomaly detection on hypergraphs using RWR scores, compared to existing approaches?



MAP in EEN

MAP in SB

MAP in HB

Additional Contents

- There are some additional contents in the paper including:
  - **Edge-dependent node weight (EDNW)**
  - Another application: node retrieval

Additional Contents: EDNW

- Let's suppose a co-authorship hypergraph

Authors (Nodes)

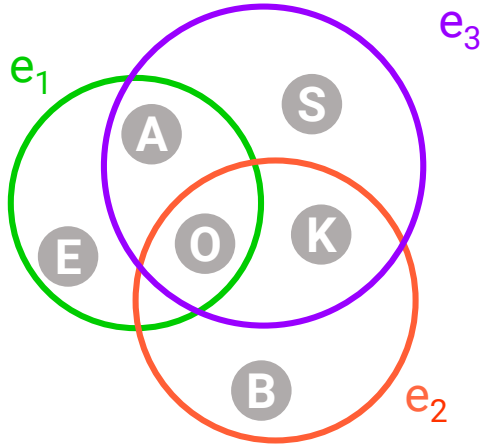
Elsa **(E)**	Anna (**A**)
Olaf (**O**)	Kristoff (**K**)
Briggs (**B**)	Sven (**S**)

Publications (Hyperedges)

e_1: (O, A, E)

e_2: (B, O, K)

e_3: (A, K, S, O)



Additional Contents: EDNW (cont.)

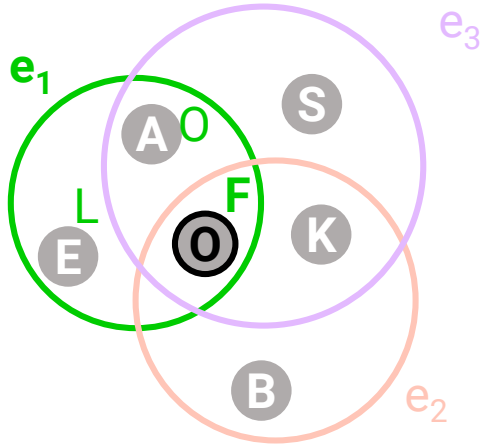
- Let's suppose a co-authorship hypergraph

e_1

Olaf (O), Anna (A), Elsa (E)

First Author (F)	Other Authors (O)	Last Author (L)

Contribution of a node may vary depending on the hyperedge



Additional Contents: EDNW (cont.)

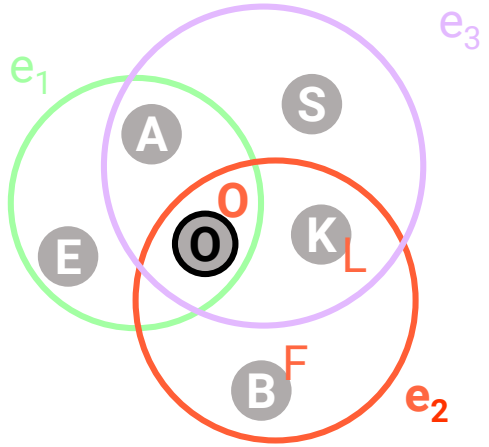
- Let's suppose a co-authorship hypergraph

e_2

Briggs (B), **Olaf (O)**, Kristoff (K)

First Author (F)	**Other Authors (O)**	Last Author (L)

Contribution of a node may vary depending on the hyperedge



Additional Contents: EDNW (cont.)

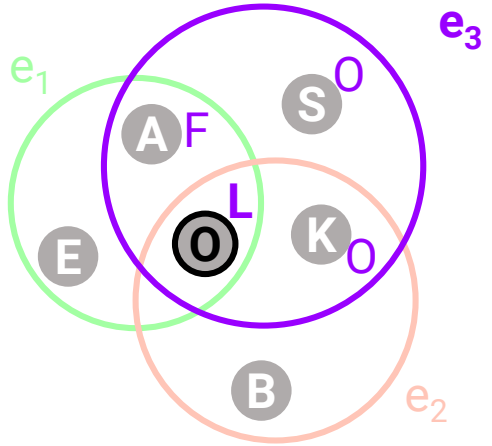
- Let's suppose a co-authorship hypergraph

e_3

Anna (A), Kristoff (K), Sven (S), **Olaf (O)**

First Author (F)	Other Authors (O)	**Last Author (L)**

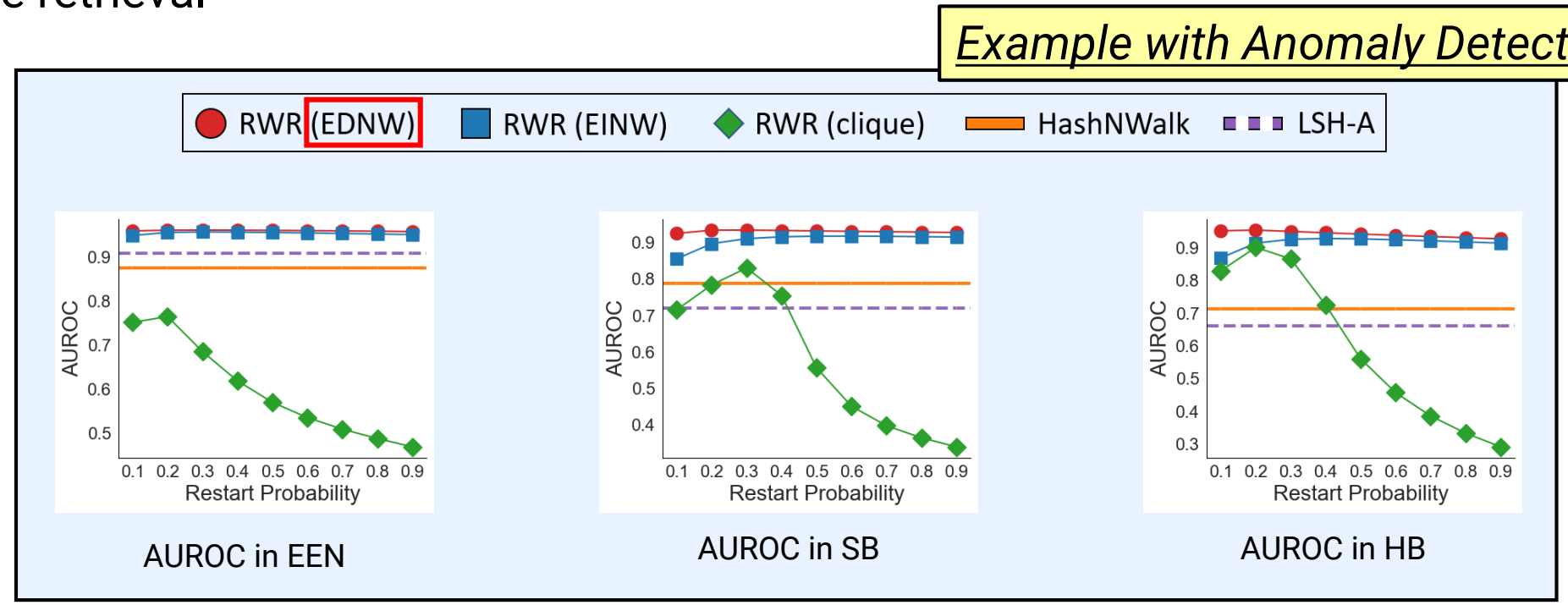
Contribution of a node may vary depending on the hyperedge



Additional Contents: EDNW (cont.)

- **ARCHER** can compute RWR on hypergraphs with **EDNW**
- We also show usefulness of **EDNW** in **applications**
  - Anomaly detection
  - Node retrieval

Example with Anomaly Detection



Roadmap

- Overview
- Random Walk with Restart (RWR) on Hypergraphs
- Proposed Algorithm: ARCHER
- Empirical Evaluation of ARCHER
- Application: Anomaly Detection <<
- Conclusion <<

Conclusions

- Our contributions are summarized as follows:

✓ We proposed **two RWR computation methods** on hypergraphs

✓ We proposed **ARCHER**, which adaptively selects between the two computation methods efficiently

✓ We proposed **an application to anomaly detection** using RWR

Paper: https://doi.org/10.1007/s10618-023-00995-9

Github repo: https://github.com/jaewan01/ARCHER

References

- **[Page'99]:** Page L, Brin S, Motwani R, et al. "The pagerank citation ranking: Bringing order to the web". Tech. rep., Stanford InfoLab. 1999.

- **[SIGMOD'15]**: Shin, Kijung, et al. "Bear: Block elimination approach for random walk with restart on large graphs." Proceedings of the 2015 ACM SIGMOD international conference on management of data. 2015.

- **[SIGMOD'17]**: Jung, Jinhong, et al. "Bepi: Fast and memory-efficient method for billion-scale random walk with restart." Proceedings of the 2017 ACM International Conference on Management of Data. 2017.

- **[ICESS'19]**: Zhang, Ying, Zhiqiang Zhao, and Zhuo Feng. "Towards scalable spectral sparsification of directed graphs." 2019 IEEE International Conference on Embedded Software and Systems. IEEE, 2019.

References

- **[FOCS'16]**: Cohen, Michael B., et al. "Faster algorithms for computing the stationary distribution, simulating random walks, and more." 2016 IEEE 57th annual symposium on foundations of computer science. IEEE, 2016.

- **[ICDM'05]**: Sun, Jimeng, et al. "Neighborhood formation and anomaly detection in bipartite graphs." Fifth IEEE international conference on data mining. IEEE, 2005.

- **[CIKM'20]**: Hayashi, Koby, et al. "Hypergraph Random Walks, Laplacians, and Clustering." Proceedings of the 29th ACM international conference on information & knowledge management. 2020.

References

- **[ICDM'16]:** Jung, Jinhong, et al. "Personalized ranking in signed networks using signed random walk with restart." 2016 IEEE 16th international conference on data mining. IEEE, 2016.

- **[IJCAI'22]**: Lee, Geon, Minyoung Choe, and Kijung Shin. "HashNWalk: Hash and Random Walk Based Anomaly Detection in Hyperedge Streams." 31st International Joint Conference on Artificial Intelligence. 2022.

- **[COMPLEX NETWORKS'17]**: Ranshous, Stephen, Mandar Chaudhary, and Nagiza F. Samatova. "Efficient outlier detection in hyperedge streams using minhash and locality-sensitive hashing. "Complex Networks & Their Applications VI: Proceedings of Complex Networks 2017. Springer International Publishing, 2018.