Querying and mining complex graphs : uncertainty and multi-relations

Ke, Xiangyu

2019

Ke, X. (2019). Querying and mining complex graphs : uncertainty and multi-relations.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/137863>

<https://doi.org/10.32657/10356/137863>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0
International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 19:06:06 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

Querying and Mining Complex Graphs: Uncertainty and Multi-Relations

A thesis submitted
in partial fulfillment of the requirement
for the degree of Doctor of Philosophy
by

KE Xiangyu

School of Computer Science and Engineering
Nanyang Technological University

Dec 2019

Abstract

Graphs are important data representations that can ubiquitously model objects and their relations in a wide diversity of real-world applications. Effective and efficient graph analytics provide users with a deeper understanding of what is behind the data, thus can benefit a lot of useful applications such as node classification, node recommendation, link prediction, etc. Nowadays, besides the rapid growth in the volume of graphs, the inherent complexity on graphs has attracted great attention from data management community. This thesis will therefore focus on querying and mining **uncertain** and **multi-relation** graphs. Uncertain graphs can characterize the inherent uncertainty in the data due to many reasons, including noisy measurements, inference and prediction models, and explicit manipulations. For example, in a road network, each road junction can be represented as a node, and each road segment can serve as an edge with a blocking (e.g., traffic jam) probability. Meanwhile, the multi-relation graphs, where multiple edges may exist between a same pair of nodes, are even more expressive. Using the same example of a road network, the probability of traffic jam occurrence may vary with time. Other examples include interaction conditions and transformation probabilities from a protein to many others in a protein-protein interaction network, the topics and the corresponding probabilities that a user may re-tweet her friends' posts, etc. The uncertainty and the multi-relations over graphs bring about additional complexities to graph querying and mining. In this thesis, I investigated the following research problems.

First, I conducted an experimental analyses about six state-of-the-art s - t reliability algorithms. The s - t reliability measures the probability that a target node t is reachable from a source node s . We introduced the algorithms and provided several corrections or adaptations. Then I implemented all of them on a same code base, and conducted the evaluation with unified metrics, medium to large real-world datasets, and same query

workloads. We summarized their trade-offs, and provided guidelines for future researchers and practitioners.

Then, I examined a network modification problem, aiming at maximizing the network reliability through adding a small budget of new edges. We designed solutions to the basic two-terminal reliability case, then generalized them to the multiple-source-target case, and further studied a restricted version of only considering most reliable paths. The experimental results and the real-world applications well validated the effectiveness and the efficiency of our solutions.

Next, I studied a novel variant of well-known influence maximization problem by finding influential seed users and relevant topics simultaneously. Instead of using a fixed information diffusion probability between each pair of users, variations of diffusion probability in different topics were allowed. We formally characterized the problem and provided an iterative solution. The efficiency of the seed finding part was improved by up to 30%, while the accuracy of the topic finding part was improved by up to 30%.

Furthermore, I applied the uncertain graph to model the crowd errors in the solution to the crowdsourcing entity resolution problem. With the help of our novel metric, the “reliability” of a clustering, the next crowdsourcing questions could be selected wisely, which thus improved the accuracy by 15%, and reduced the crowdsourcing cost by 50%. I also developed a demo software for the proposed PERC system.

Finally, I presented our multi-relation graph summarization techniques. In our two-step baselines, a final summary was generated by properly aggregating the summary of each relation. I demonstrated their drawbacks and designed holistic solutions. Among them, the k -median clustering approach was able to return a summary with a provable bounded size.

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

[December 5, 2019]

.....

Date

[ ]

.....

[Xiangyu KE]

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

[December 5, 2019]

.....
Date

[*Arijit Khan*]

.....
[Arijit KHAN]

Authorship Attribution Statement

This thesis contains material from 4 papers published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Chapter 2 is published as [Xiangyu Ke](#), Arijit Khan, and Leroy Lim Hong Quan. *An In-Depth Comparison of s-t Reliability Algorithms over Uncertain Graphs*. Proceedings of the VLDB Endowment (PVLDB), Volume 12 Issue 8, April 2019, Pages 864-876. DOI: 10.14778/3324301.3324304.

The contributions of the co-authors are as follows:

- Asst. Prof. Arijit Khan discussed the initial idea with me.
- I designed and conducted the experimental evaluation, discussed the results with Asst. Prof. Arijit Khan, and drew the conclusions.
- I prepared the initial drafts. Asst. Prof. Arijit Khan helped edit the manuscript.
- Mr. Leroy Lim Hong Quan helped implement some codes.

Chapter 3 is published as [Xiangyu Ke](#), Arijit Khan, Mohammad Al Hasan, and Rojin Rezvansangsari. *Reliability Maximization in Uncertain Graphs*. IEEE Transactions on Knowledge and Data Engineering, (To appear) 2020.

The contributions of the co-authors are as follows:

- Asst. Prof. Arijit Khan discussed the initial idea with me.
- I theoretically analyzed the problem, designed the algorithms and conducted the experimental evaluation, discussed the results with Asst. Prof. Arijit Khan, and drew the conclusions.
- I prepared the initial drafts. Asst. Prof. Arijit Khan helped edit the manuscript.
- Ms. Rojin Rezvansangsari helped study the restricted problem: most-reliable paths based reliability maximization.

- Prof. Mohammad Al Hasan helped reviewed the manuscripts, and provided several important comments.

Chapter 4 is published as [Xiangyu Ke](#), Arijit Khan, and Gao Cong. *Finding Seeds and Relevant Tags Jointly: For Targeted Influence Maximization in Social Networks*. Proceedings of the 2018 International Conference on Management of Data (SIGMOD), Pages 1097-1111. DOI: 10.1145/3183713.3199670.

The contributions of the co-authors are as follows:

- Asst. Prof. Arijit Khan discussed the initial idea with me.
- I theoretically analyzed the problem, and designed the solutions. I had several discussions with Asst. Prof. Arijit during the procedure, and he provided useful feedbacks.
- I designed and conducted the experimental evaluation, discussed the results with Asst. Prof. Arijit Khan, and drew the conclusions.
- I prepared the initial drafts. Asst. Prof. Arijit Khan helped edit the manuscript.
- Prof. Gao Cong helped reviewed the manuscripts, and provided several important comments.

Chapter 5 is published as following two papers:

1. Vijaya Krishna Yalavarthi, [Xiangyu Ke](#), and Arijit Khan. *Select Your Questions Wisely: For Entity Resolution with Crowd Errors*. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM), Pages 317-326. DOI: 10.1145/3132847.3132876.
2. [Xiangyu Ke](#), Michelle Teo, Arijit Khan, and Vijaya Krishna Yalavarthi. *A Demonstration of PERC: Probabilistic Entity Resolution with Crowd Errors*. Proceedings of the VLDB Endowment (PVLDB), Volume 11 Issue 12, August 2018, Pages 1922-1925. DOI: 10.14778/3229863.3236225.

The contributions of the co-authors in the paper 1 are as follows:

- Asst. Prof. Arijit Khan provided the initial direction.
- Mr. Vijaya Krishna Yalavarthi and Asst. Prof. Arijit Khan analyzed the main challenges of the problem, and designed the framework. I joined the project, and participated in designing the algorithms.
- Mr. Vijaya Krishna Yalavarthi and I conducted the online crowdsourcing. In particular, Mr. Vijaya Krishna Yalavarthi designed the interface and initial the crowdsourcing procedure, then I collected and analyzed the data.
- Mr. Vijaya Krishna Yalavarthi and I implemented the codes, executed the experiments, and analyzed the results. Asst. Prof. participated in the discussion, and helped modify our experiment plan.
- Asst. Prof. prepared the initial drafts. Mr. Vijaya Krishna Yalavarthi and I edited the manuscript.

The contributions of the co-authors in the paper 2 are as follows:

- Prof. Arijit Khan and I decided to build a prototype system for paper 1.
- I designed the system architecture, and implement the initial interface.
- Mrs. Michelle Teo helped implement the detailed backend functions, and improved the interface.
- I finalized the system with core algorithms from paper 1 and prepared the data.
- Mrs. Michelle Teo created a demonstration video for our system.
- I prepared the initial draft. Asst. Prof. Arijit Khan and Mr. Vijaya Krishna Yalavarthi helped edit the manuscript.

[April 10, 2020]

[ ]

.....
Date

.....
[Xiangyu KE]

Acknowledgments

I would express my deepest gratitude to my supervisor, Prof. Arijit Khan in Nanyang Technological University, Singapore. He trained my research skills step-by-step, and gave me many useful suggestions that inspired me in many aspects of my research direction. Without his guidance, the achievements of my research work will never be possible.

I would like to thank all my co-authors for the projects we are working together, and my friends and colleagues for their warm companion and enlightening discussion. From them, I acquire strength and inspiration to march on. They include but not limited to: Prof. Gao Cong, Prof. Mohammad Al Hasan, Prof. Richard T.B. Ma, Dr. Francesco Bonchi, Dr. Hongxia Yang, Dr. Tianxing Wu, Dr. Yuxiang Wang, Xuan Liu, Vijaya Krishna Yalavarthi, Arkaprava Saha, Appan Rakaraddi, Rojin Rezvansangsari, Leroy Lim Hong Quan, and Michelle Teo.

Last but not least, special thanks to all my family. I show my immense gratitude and love to my beloved partner Yang Song for all the support he did for my study and life. I would also like to show my great appreciation to my parents, Chaohua Yang and Conghua Ke, for their constant encouragement, understanding, support and love. Without these, I would not have finished my Ph.D. works easily.

Contents

Abstract	i
Declaration	iv
Acknowledgments	viii
List of Figures	xiv
List of Tables	xvii
1. Introduction	1
1.1 Problems and Motivations	3
1.2 Approaches and Contributions	6
1.3 Outlines	9
2. Experiments and Analyses:	
<i>s-t</i> Reliability Algorithms in Uncertain Graphs	10
2.1 <i>s-t</i> Reliability in Uncertain Graphs	11
2.2 Reliability Estimation Algorithms	12
2.2.1 MC Sampling	12
2.2.2 Indexing via BFS Sharing	14
2.2.3 Recursive Sampling	15
2.2.4 Recursive Stratified Sampling	18
2.2.5 Lazy Propagation Sampling	18
2.2.6 Indexing via Probabilistic Trees	22
2.2.7 Other Related Work	24

2.3	Horizontal Comparison of Algorithms:	
	Strengths and Weaknesses	25
2.4	Experimental Results	27
2.4.1	Environment Setup	27
2.4.2	Estimator Variance and Convergence	30
2.4.3	Trade-off among Relative Error, Running Time, and Memory Usage	33
2.4.4	Estimator Accuracy	33
2.4.5	Estimator Efficiency	35
2.5	Summary and Recommendation	37
3.	Reliability Maximization in Uncertain Graphs	40
3.1	Preliminaries	41
3.1.1	Problem Formulation	41
3.1.2	Hardness of the Problem	42
3.1.3	Characterization of the Problem	46
3.2	Related Work	48
3.3	Baseline Methods	49
3.3.1	Individual Top- k Method	50
3.3.2	Hill Climbing Method	50
3.3.3	Centrality-based Method	50
3.3.4	Eigenvalue-based Method	51
3.4	A Simplified Problem: Improve The Most Reliable Path	51
3.5	Proposed Solution: Budgeted Reliability Maximization	55
3.5.1	Search Space Elimination	55
3.5.2	Top- k Edges Selection	57
3.6	Multiple-Source-Target Reliability Maximization	60
3.6.1	Maximizing the Average Reliability	61
3.6.2	Maximizing the Minimum Reliability	62
3.6.3	Maximizing the Maximum Reliability	63
3.7	Experimental Results	64
3.7.1	Experimental Setup	64
3.7.2	Single-source-target results	67

3.7.3	Multiple-source-target results	71
3.7.4	Case Studies	73
3.8	Summary	77
4.	Finding Seeds and Tags Jointly for Social Advertising	78
4.1	Preliminaries	79
4.1.1	Problem Formulation	79
4.1.2	Hardness of the Problem	82
4.1.3	Overview of Our Solution	85
4.2	Related Work	85
4.3	Algorithms for Finding Seeds	87
4.3.1	Targeted Reverse Sketching	87
4.3.2	Indexing for Targeted Reverse Sketching	89
4.4	Algorithms for Finding Tags	94
4.4.1	Individual Paths Selection	94
4.4.2	Shortcomings of Individual Paths Selection	95
4.4.3	Batch-Paths Selection	96
4.5	Proposed Iterative Algorithm	99
4.6	Initialization for Iterative Algorithm	100
4.7	Experimental Results	101
4.7.1	Experimental Setup	101
4.7.2	Accuracy and Efficiency Analysis	103
4.7.3	Case Study	104
4.8	Summary	106
5.	PERC System: Probabilistic Entity Resolution with Crowd Errors . .	107
5.1	Preliminaries	108
5.1.1	Background	108
5.1.2	Entity Resolution Problem	111
5.2	Related Work	111
5.3	Framework Overview	113
5.4	Core Algorithms	115

5.4.1	Entity Resolution Processor	115
5.4.2	Next Crowdsourcing Processor	116
5.5	Experimental Results	118
5.5.1	Environment Setup	118
5.5.2	Next Crowdsourcing Results	120
5.6	Demonstration	122
5.7	Summary	124
6.	Multi-Relation Graph Summarization	125
6.1	Problem and Related Work	126
6.1.1	Multi-relation graph summarization	128
6.2	Other related work	130
6.3	Baseline Methods: Aggregate Summaries Across Relations	131
6.3.1	Overview	131
6.3.2	Single-Relation Graph Summarization	132
6.3.3	Summary Aggregation	140
6.3.4	Limitation of Baselines Methods	143
6.4	Proposed Algorithms: Holistic Summary of All Relations	145
6.4.1	Holistic k -Median: k -Median ⁺	145
6.4.2	Holistic Greedy: Greedy ⁺	148
6.4.3	Holistic Randomized: Randomized ⁺	148
6.5	Finding Optimal Number of Supernodes	148
6.6	Experimental Results	150
6.6.1	Experimental Setup	150
6.6.2	Comparison with Two-Step Baselines	152
6.6.3	Exact Storage Cost	154
6.6.4	Scalability Analysis	155
6.6.5	Varying Number of Supernodes	156
6.6.6	Case Study on DBLP: Visualization	156
6.6.7	Case Study on Brain Network Dataset: Classification and Visual- ization	158
6.7	Summary	159

7. Conclusions and Future Works	160
7.1 Conclusions	160
7.2 Future Directions	161
Publications	163
References	164

List of Figures

1.1	Edge influence probabilities based on information topics	2
1.2	Running example: Edge probability denotes ratio of crowd workers voted YES for the respective records pair to be same entity.	5
2.1	The broad spectrum of reliability problem over uncertain graphs	11
2.2	Compact structure with five possible worlds: BFS sharing	14
2.3	Toy graph for Example 2.1	21
2.4	Reliability estimated by MC, Lazy Propagation (LP), and Lazy Propagation+ (LP+) at convergence. Reliability is reported as the average over 100 $s-t$ pairs. For experimental setting, see Sec. 2.4.	21
2.5	Example of FWD ProbTree ($w=2$) construction	23
2.6	Comparison of estimator variance and convergence. The symbol ρ_K on Y-axis denotes the ratio $\frac{V_K}{R_K} = \frac{\text{average variance at \#samples}=K}{\text{average reliability at \#samples}=K}$	30
2.7	Comparison of average reliability returned by each estimator in regards to average reliability returned by MC sampling at $K = 10\,000$ (shown as horizontal dashed line), <i>BioMine</i>	31
2.8	The trade-off between relative error and running time/memory usage, <i>lastFM</i>	32
2.9	The trade-off between relative error and running time/memory usage, <i>BioMine</i>	33
2.10	The decision tree for selecting a proper reliability estimator under different scenarios	38
3.1	Reduction from MAX k -COVER to single $s-t$ budgeted reliability maximization problem.	43
3.2	Example for non-submodularity and non-supermodularity	45

3.3	Example for the problem characterization	46
3.4	Run-through example for the proposed algorithm	56
3.5	Reliability gain and running time comparison with varying budget on #new edges k . $\zeta = 0.5$, $r = 100$ $l = 30$, #Sources=#Targets=100, <i>Twitter</i>	73
3.6	Improving the reliability from sensor 21 (right) to 46 (left) with 3 new links (marked by dotted lines).	74
3.7	Improving the reliability from sensor 15 to 40 (on the diagonal) with 3 new links (marked by dotted lines).	75
3.8	Influence spread comparison in <i>Databases</i> area. $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, <i>DBLP</i>	75
4.1	Hardness of top- r tags selection	83
4.2	Example of non-submodularity. $P(e c) = 0$ for all other tag and edge combinations that are not specified.	84
4.3	Running time of TRS (state-of-the-art) vs. LL-TRS (our method), <i>Twitter</i> , #tags=5, #targets=3K, $\epsilon=0.1$, $\alpha=1$, $\delta=0.01$, $h=3$	89
4.4	I-TRS indexing and querying: The original uncertain graph contains three tags $\{c_1, c_2, c_3\}$. $P(e c) = 0$ for all other tags that are not specified. The query considers the target set $T = \{D, E\}$. The selected tag set $C_1 = \{c_1, c_2\}$. We set $\theta = 4$, $\theta_c = 3$	90
4.5	Implication of Theorem 4.6, <i>Yelp</i> . #targets=3K, #seeds=10, $\delta = 0.01$, $\alpha = 1$. Optimal tags are used.	93
4.6	Tags selection. Seed set = $\{A, B, C\}$, target set = $\{G, H, I\}$	95
4.7	Batch-paths lattice for Figure 4.6. Dashed circles present redundant search space due to edges between seeds (removed).	97
4.8	Updates of batch-paths lattice on inclusion of tags $\{4, 5\}$	97
4.9	Accuracy and efficiency analysis to adjust seed set size k . $r = 10$, $ T = 3K$. 103	
4.10	Accuracy and efficiency analysis to adjust tag set size r . $k = 20$, $ T = 3K$. 103	
4.11	Influence spread comparison using cross-city tags. $k=10$ seeds, $r=10$ tags, Y-axis denotes the % of spread by 10 selected tags, over total spread by all 195 tags.	105

5.1	Possible worlds of an uncertain graph: Three possible worlds G_5 , G_6 , G_7 are not clusterings, as they are not transitively closed. For example, in G_5 , $A = C$ and $C = B$, but $A \neq B$, thus violating transitivity.	109
5.2	The architecture of PERC system	113
5.3	Workflow of PERC	115
5.4	Reliability of a clustering	117
5.5	Cost improvement: # next crowdsourcing questions required to reach a certain accuracy (F1-measure)	121
5.6	Accuracy improvement (F1-measure) for next crowdsourcing	121
5.7	Efficiency improvement: Computation time required to select a batch of next crowdsourcing questions	121
5.8	User interface of PERC	123
6.1	A simple graph and its summary.	127
6.2	A multi-relation graph and its summary.	129
6.3	The individual summary for each relation in Figure 6.2(a).	129
6.4	Example for single-relation graph summary methods.	138
6.5	Empirical comparison among five summary aggregation algorithms over <i>DBLP_10</i>	142
6.6	The example graph with 3 relations.	144
6.7	The individual summary for each relation.	144
6.8	(a) The graph with distances between nodes for summary aggregation computation. We omit node E here since its distance to any other node is 1. (b) The aggregated summary of the individual summaries. All the summary aggregation algorithms produce the same result in this example. (c) The potentially best summary S_2	144
6.9	Trade-off between relative size and running time. Sub-figure (f) presents breakup on #superedges and #correction-edges for summaries.	152
6.10	Scalability analysis.	155
6.11	Sensitivity analysis with #supernodes k , <i>Amazon</i>	156
6.12	Case study on <i>DBLP</i>	157
6.13	Case study on <i>Austism</i> . The partitionings of the same set of 14 nodes are quite different for ASD brain networks and for healthy brain networks. The numbers on superedges denote the number of relations having such superedges.	158

List of Tables

2.1	Stratum design for recursive stratified sampling	17
2.2	Properties of datasets	27
2.3	Comparison of relative error (RE): <i>LastFM</i>	34
2.4	Comparison of relative error (RE): <i>DBLP_0.05</i>	34
2.5	Comparison of running time: <i>AS_Topology</i>	35
2.6	Comparison of running time: <i>BioMine</i>	36
2.7	Summary and recommendation	37
3.1	Reliability gains of three possible solutions for the example in Figure 3.3 under different probability setting.	46
3.2	Reliability gain and running time comparison without search space elimination. $k = 10, \zeta = 0.5, lastFM$	52
3.3	Reliability gain and running time comparison after search space elimination. $k = 10, \zeta = 0.5, l = 30, r = 100, lastFM$. Additional time for search space elimination: 16 sec.	53
3.4	Properties of datasets, SPL denotes shortest path length, and C. Coe. denotes clustering coefficient	64
3.5	Comparison with the exact solution (ES), $k = 3, \zeta = 0.33, r = 54, l = 30, Intel Lab Data$	68
3.6	Single-source single-target budgeted reliability maximization on different datasets. $k = 10, \zeta = 0.5, r = 100, l = 30$	68
3.7	Single-source-target reliability maximization on different synthetic datasets. $k = 10, \zeta = 0.5, r = 100, l = 30$	69
3.8	Reliability gain and running time comparison with varying budget on #new edges k . $\zeta = 0.5, r = 100, l = 30, DBLP$	70

3.9	Reliability gain, running time comparison with varying probability ζ on new edges. $k = 10, r = 100, l = 30, AS_Topology$	71
3.10	Analysis with different probabilities on new edges. $k = 10, r = 100, l = 30, Twitter$	71
3.11	Reliability gain and running time comparison for multiple-source-target pairs. $k = 100, \zeta = 0.5, r = 100, l = 30, \frac{k_1}{k} = 10\%, Twitter (Min)$	72
3.12	Reliability gain and running time comparison for multiple-source-target pairs. $k = 100, \zeta = 0.5, r = 100, l = 30, \frac{k_1}{k} = 10\%, Twitter (Max)$	72
3.13	Reliability gain and running time comparison for multiple-source-target pairs. $k = 100, \zeta = 0.5, r = 100, l = 30, \frac{k_1}{k} = 10\%, Twitter (Avg)$	72
4.1	Accuracy comparison between TRS and I-TRS with optimal tags, <i>Yelp</i> . #targets=3K, $\delta = 0.01, \alpha = 1$	93
4.2	Characteristics of datasets.	101
4.3	Accuracy and efficiency for different initialization methods, <i>Yelp</i> . $r = 20, T = 3K$	104
4.4	Case study: top-10 tags found for influence maximization in three target cities.	104
5.1	Crowdsourcing cost reduction by PERC over state-of-the-art approaches: We present the number of crowdsourcing questions required to achieve a certain accuracy (0.8 F1-measure for <i>Cora</i> dataset, and 0.9 for others) for various methods.	108
5.2	Properties of datasets	118
6.1	Characterization of the most related papers.	130
6.2	Properties of datasets.	150
6.3	#supernodes, and (#superedges, #correction-edges) in obtained summaries via our holistic methods.	152
6.4	Actual storage cost (MB) for graphs and summaries. The actual storage cost for summaries includes the supernode mapping(s). We also report summary storage percentage w.r.t. original graph storage.	154

6.5	Actual storage cost (MB) for graphs and summaries with further compression: (super)edges between the same set of (super)nodes over multiple relations are stored as $\langle node_i, node_j, relation_x, relation_y, \dots, relation_z \rangle$. The actual storage cost for summaries includes the supernode mapping(s).	154
6.6	Breakup of the actual storage cost (MB) for summaries with further compression (shown in Table 6.5). E_S denotes superedges, C denotes correction edges, and M denotes the node mapping.	155

Chapter 1

Introduction

Graphs are important data representations that can ubiquitously model objects and their relations in a wide diversity of real-world applications. Besides the rapid growth in the volume of graphs in the big data era, how to handle the inherent complexities on graphs becomes a hot topic in data management community. In this thesis, the focuses will be on querying and mining **uncertain graphs** and **multi-relation graphs**.

In an uncertain graph (or, *probabilistic* graph), each edge is associated with a probability of existence. The uncertainty arises due to a variety of reasons: noise measurements [3], inference and prediction models [1], explicit manipulations (e.g., for privacy purposes [20]), and so on. For example, in a road network, each road junction can be represented as a node, and each road segment is an edge with a probability of blocking. In the *Twitter* social network, we consider each user account as a node, and their friendships as the edges. A user may have some probabilities to re-tweet her friends.

In a multi-relation graph (also known as *multi-layer*, *multiplex*, or *multi-dimensional* graph), multiple edges can exist between a same pair of nodes, representing different relations. For instance, given two time slots, *8AM-9AM* and *10AM-11AM*, the probability of a traffic jam on a same road segment could be quite different. Meanwhile, in the *Twitter* social network, the re-tweet probabilities between a same pair of users can vary among topics, as depicted in Figure 1.1. Other similar examples include interaction conditions in biological networks, research topics in co-authorship networks, with or without probabilities.

The uncertainty and the multi-relations bring about great challenges to graph querying and mining. For example, the query asking whether a target node t is reachable from

Figure 1.1: Edge influence probabilities based on information topics

a source node s returns either yes or not on deterministic graphs, and can be solved in polynomial time with simple breath-first search. However, under the uncertain graph setting, the target node t is reachable from the source node s with a probability p , which is known as the s - t (or, *two-terminal*) *reliability* on uncertain graphs. The s - t reliability computation is a prototypical $\#P$ -complete problem [10, 171]. If we are interested in the s - t reliability with a given set of labels (e.g., social topics), it becomes *conditional s-t reliability query* [98], and more complex.

In this thesis, I started with an experimental analyses on the current algorithms for the aforementioned fundamental s - t *reliability query*. I summarized their trade-offs, and provided guidelines for researchers and practitioners. Then, I investigated a *network modification* problem, targeting at improving the s - t reliability in a network by adding a small budget of new edges. In the well-known *influence maximization* problem, the influence computation is a variant of s - t reliability computation. I further incorporated the concept of multi-relation graphs, and studied a novel problem of jointly finding the influential seed users and the relevant topics for viral marketing. Then, I designed a framework and developed a demo system, named PERC, to solve the *crowdsourcing entity resolution* problem by modeling the crowd answers as a uncertain graph. With the help of our novel concept, the *reliability of a clustering*, the next crowdsourcing questions could be selected wisely, which reduced the overall cost. Last but not least, I examined the problem of *multi-relation graph summarization*, aiming to reduce the large overhead of graph storage, and can help us discover some meaningful patterns across relations. In what follows, this chapter will briefly introduce the aforementioned problems and our proposed solutions. More details will be presented in the following chapters, respectively.

1.1 Problems and Motivations

Experimental Analyses for s - t Reliability Algorithms. A fundamental query in uncertain graphs is the so-called s - t *reliability query*, which asks to measure the probability that two given nodes are reachable [4]. The s - t reliability query has been used in many applications such as measuring the quality of connections between two terminals in a sensor network [71], finding other proteins that are highly probable to be connected with a specific protein in a protein-protein interaction (PPI) network [84], probabilistic path queries in a road network [80], and evaluating information diffusions in a social influence network [98], etc.

Due to the inherent complexity of the problem ($\#P$ -hard [10]), although the exact reliability detection has received attention in the past [4], the focus nowadays has mainly been on approximated and heuristic solutions over large-scale graphs [99], mostly via sampling and indexing. *With the wide range of algorithms available for estimating the s - t reliability over uncertain graphs, there is an urgent need to realize their trade-offs, and to employ the best algorithm for a given scenario.* In this experimental study, we were the first to compare 6 state-of-the-art methods [63, 84, 114, 116, 123, 194] with each other, summarize their trade-offs, and provide the guidelines for the practitioners.

Reliability Maximization in Uncertain Graphs. I investigated the novel problem of adding a small number of edges in an uncertain network for maximizing the reliability between a given pair of nodes. We referred to such edges as *shortcut edges* and the problem of identifying the best set of k edges as the *budgeted reliability maximization* problem. Our problem falls under the broad categories of uncertain networks design [177], optimization [139], and modification [133] problems, yet surprisingly this specific problem has not been studied in the past.

The budgeted reliability maximization problem is critical in the context of many physical networks, such as transportation and communication networks. In mobile ad-hoc networks, the connectivity between sensor nodes and devices is estimated using noisy measurements, thus leading to edges naturally associated with a probability of existence [71]. Road networks can be modeled as uncertain graphs because of unexpected traffic congestion [80]. In these networks, creating new connections between nodes (e.g., building

new roads, flyovers, adding Ethernet cables) is limited by physical constraints and budget. One can introduce only k new edges where k is decided based on resource constraints. Thus, our goal is to intelligently add k new edges such that the reliability between a pair of important nodes is maximized [177]. Furthermore, in social networks, finding k best shortcut edges could maximize the information diffusion probability from an early adopter to a target customer [36], thus the network host can actively recommend these links to the respective users. In protein-interaction networks, interactions are established for a limited number of proteins through noisy and error-prone experiments—each edge is associated with a probability accounting for the existence of the interaction. Therefore, finding the top- k shortcut edges can assist in de-noising protein-interaction networks [109].

Jointly Finding Influential Seed Users and Topics in Social Networks. The classical influence maximization problem [40, 94] identifies the top- k seed users in a social network such that the expected number of influenced users in a network, starting from those seeds and following an influence diffusion model, is maximized. Most of them assume fixed edge probabilities, that is, a user can influence another with certain probability. However, the information propagated in social medias is always associated with some topics, e.g., hashtags on Twitter, and users may have different preferences for them (Figure 1.1). Setting a small budget on the number of tags is critical, for example, (a) to avoid information overload against campaign effectiveness (e.g., if one wants to write an advertisement or a blog, she may want to focus on a small number of relevant topics), and (b) for cost-effective planning and due to other physical constraints (e.g., an album can accommodate a *limited* number of genres/ songs). Motivated by these timely demands, I revisited the classical influence maximization problem, and investigated a novel problem as follows: *Given a set of target customers, a small budget k on the number of seed users, and a small budget r on the number of relevant tags, what are the top- k seed nodes and the top- r most relevant tags that maximize the expected spread of the campaign within the target set?*

PERC System: Probabilistic Entity Resolution with Crowd Errors. *Entity Resolution (ER)* is the task of disambiguating manifestations of real-world entities in various records by linking and clustering [70]. For example, there could be different ways of addressing the same person in text, or several photos of a particular object. Also known

Figure 1.2: Running example: Edge probability denotes ratio of crowd workers voted YES for the respective records pair to be same entity.

as *Deduplication*, this is a critical step in data cleaning and analytics, knowledge base construction, comparison shopping, health care, and law enforcement, among many others. Comparing with machine-based methods, human workers can provide more insights in some complex tasks, e.g., video tagging and natural language processing [73], and are able to handle various types of data. However, human can still make mistakes due to various factors, thus, a fault-tolerant crowdsourcing system for ER task is on demand.

In this work, we applied the uncertain graph model to provide a solution. Consider a dataset of eight images shown in Figure 1.2. Records A , B belong to famous American actress and model, *Eva Mendes*; C , D to Bollywood star and lead actress of the American television series, *Quantico*, *Priyanka Chopra*; and E , F , G , H to Hollywood actor *Tom Cruise*. 80% of crowd workers voted YES that both records in each of the following pairs are same: $\langle A, B \rangle$, $\langle C, D \rangle$, $\langle E, F \rangle$, and $\langle G, H \rangle$. All crowd workers also answered NO for the edges between the following cluster pairs: $\langle C_1, C_3 \rangle$, $\langle C_2, C_3 \rangle$, $\langle C_1, C_4 \rangle$, and $\langle C_2, C_4 \rangle$. In this example, four clusters C_1 , C_2 , C_3 and C_4 are formed, as shown in Figure 1.2. Our objective is to identify the next question to crowdsource that maximizes the gain. It can be observed that asking a question between clusters C_3 and C_4 is more beneficial because all images in C_3 and C_4 belong to the same entity, and one more edge with probability greater than 0.5 helps in merging these two clusters.

Multi-Relation Graph Summarization. In big data era, the size of data graphs is

growing at an unprecedented rate, spanning millions and billions of nodes and edges. For example, Facebook has 800 million active users and related network data [110]. Graph-of-Things (GoT), which is a live knowledge graph system for Internet-of-Things, has been adding millions of records per hour, and roughly more than 10 billion RDF triples per month [140]. At the current rate of data volume increase, it becomes highly impractical to store, manage, process, and visualize such big graphs. A variety of graph summarization and compression techniques have been developed, including grouping of nodes and edges into supernodes and superedges [112, 134, 156], relabelling nodes [22, 43, 56], and encoding common structures (e.g., cliques, bipartite-cores, and stars) with fewer bits [27, 148]. These techniques are either *lossless* or *lossy* depending on whether the input graph can be reconstructed exactly from the summary. Surprisingly, little attention has been paid to the problem of summarizing multi-relation graphs. If understanding and querying single-relation graphs are difficult, would they not be even harder over multi-relation networks? Hence, there is a critical need to summarize multi-relation graphs for interactive query processing and visualization, as well as to discover complex, hidden, and correlated inter-dependencies across different entities in the dataset.

1.2 Approaches and Contributions

Experimental Analyses for s - t Reliability Algorithms. The state-of-the-arts algorithms for answering s - t reliability query have not been thoroughly compared with each other. In this work, we first investigated the s - t reliability estimation problem and summarized six state-of-the-art sequential algorithms, together with their time complexity and sampling variance. During which, we corrected certain issues in the past algorithms (e.g., accuracy of [116], time complexity analysis of [194]), and significantly improved their performance. Then, we re-implemented six state-of-the-art s - t reliability estimation methods in a common system and code base, using several medium and large-scale, real-world graph datasets, identical evaluation metrics, and query workloads, to provided a throughout evaluation. Finally, We reported their accuracy, efficiency, memory cost, summarized their trade-offs, and provided guidelines for researchers and practitioners.

Reliability Maximization in Uncertain Graphs. The budgeted reliability maximization problem is non-trivial. I theoretically investigated the **NP**-hardness of the

proposed problem, and showed that it was also hard to approximate. It does not admit any **PTAS**, and its underlying objective function is neither submodular, nor supermodular, which prevents us from using an iterative hill-climbing based greedy algorithm that maximizes the marginal gain at every iteration to obtain a solution with approximation guarantees. Our proposed solution systematically minimized the search space by only considering the missing edges between nodes that have reasonably high reliability from the source node and to the target node. Next, the solution extracted several highly-reliable paths between source and target nodes, after including those limited number of candidate edges in the input graph. This was motivated by the observation that what really matters in computing the reliability between two nodes was the set of paths connecting the source to the target, not the individual edges in the graph [41, 98, 102]. Our algorithm then iteratively selected these paths so as to achieve maximum improvement in reliability while satisfying the constraint on the number of new edges (k) to be added.

I also considered a *restricted* version of the problem, which approximates the reliability by considering *only* the most reliable path between the source and the target node [41, 102]. We proved that improving the probability of the most reliable path can be solved exactly in polynomial time, which yielded an efficient algorithm for the restricted version of our problem. Finally, we focused on the *generalizations* where multiple source and target nodes could be provided as input, thus opening the stage to a wider family of queries and applications, e.g., network modification for targeted influence maximization [36, 91].

Jointly Finding Influential Seed Users and Topics in Social Networks. In this work, we first proved the **NP**-hardness of proposed problem, and the **#P**-hardness of conditional influence spread estimation, together with some characteristics about approximation hardness. Then I revisited the state-of-the-art algorithms for both finding seeds in traditional influence maximization setting [164], and finding dominant tags for information diffusion between two sets of users [98]. I designed smart indexing scheme to reduce the running time of former by up to 30%, and improved the accuracy of the latter by up to 30% by refining the algorithm. By combining them with a provable sub-optimized iterative manner with smart initialization strategies, we were able to obtain

high quality results efficiently, demonstrated both with the experimental evaluation and the real-world case study.

PERC System: Probabilistic Entity Resolution with Crowd Errors. Many state-of-the-art works elude the severe concern of crowd errors by majority voting [173,175,182], that is, to ask the same question to multiple people and consider the majority answer; or by assigning a universal error rate for crowd taskers [172]. The majority voting is often unreliable because spammers and low-paid workers may collude to produce incorrect answers [120]. Besides, the tasker crowd is large, anonymous, transient, and it is usually difficult to establish a trust relationship with a specific worker [89]. Each batch of tasks is solved by a group of taskers who may be completely new, and one may not see them again, thereby making it unrealistic to assign a universal error rate for all workers over all crowdsourcing tasks. In our work, we applied the uncertain graph model to characterize the crowd errors. A novel metric called the “reliability” of a clustering was introduced to measure the connected -ness within and across clusters by following the notion of uncertain graphs, which enabled the wise selection of next crowdsourcing questions. Based on it, we designed an end-to-end solution, named by PERC, for crowdsourced ER. Our algorithm is parameter-free, that is, we do not require any user-defined threshold, and the apriori knowledge about the error rate of the crowd workers. Based on detailed empirical results with four real-world datasets, PERC improved the accuracy by 15%, reduces the crowdsourcing cost by 50%, and also decreased the next question selection time by an order of magnitude compared to state-of-the-art approaches. Furthermore, we implemented a demo system for the proposed pipeline.

Multi-Relation Graph Summarization. I first analyzed the difficulties for determining an optimal solution, and discussed why the existing approaches for some other similar graph problems could be inadequate in our scenario. Then, we designed baseline two-step algorithms, which first generated a lossless summary for each relation, then properly aggregated them to obtain one uniform summary. Among them, a k -median algorithm was able to produce a single-relation summary *with a provable bounded size*. I also demonstrated how the baseline approaches could suffer from effectiveness issues, and further proposed holistic algorithms for more compact and efficient summary generation. Our holistic k -median⁺ algorithm *maintained the same approximation guarantee* for the

multi-relation graph summarization. Our throughout experimental evaluation confirmed that our holistic algorithms could produce up to 5% more compact summaries, within half of the running time of baseline approaches. The real-world application on visualization demonstrated the effectiveness of our algorithms.

1.3 Outlines

This thesis is structured as follows. Chapter 2 presents our experimental analyses for six state-of-the-art s - t reliability algorithms on uncertain graphs. Chapter 3 describes how to maximize the reliability in uncertain graphs via adding a small budget of new edges. Chapter 4 discusses the problem of jointly finding the influential seed users and topics in social networks to enhance the viral marketing. Chapter 5 demonstrates our algorithm and system for error-tolerant crowdsourcing entity resolution. Chapter 6 provides our summary techniques for multi-relation graphs. Each of them generally consists of the detailed formulations, hardness characteristics, corresponding literature review, baseline and proposed solutions, some auxiliary strategies (e.g., initialization methods for the influence maximization work), throughout experimental evaluation, and some real-world case studies. At the end, Chapter 7 concludes the thesis, and presents some promising future directions.

Chapter 2

Experiments and Analyses: s - t Reliability Algorithms in Uncertain Graphs

In this chapter, we shall investigate two-terminal reliability: The probability that a target node t is reachable from a source node s in an uncertain graph, also denoted as the s - t reliability. Due to the inherent complexity of the problem ($\#P$ -hard) [10], although the exact reliability detection has received attention in the past [4], the focus nowadays has mainly been on approximate and heuristic solutions over large-scale graphs [99]. The large spectrum of the reliability problem is categorized in Figure 2.1. In this work, we shall focus on *sequential algorithms for the fundamental s - t reliability query*¹. Notice that we would not consider distributed algorithms [42, 196], other simplified versions of the s - t reliability problem [41, 104, 158], neither the reduction of uncertainty of a graph (e.g., by crowdsourcing) before s - t reliability estimation [64–66, 119]. However, our work can benefit the aforementioned studies in many aspects: (1) The distributed algorithms are usually designed based on the fundamental sequential algorithms involved in our paper, e.g., [196] distributed the procedure of possible world sample generation, which is exactly the distributed adoption of basic algorithm in [63]. We provide the trade-offs about the accuracy, efficiency and memory usages about these algorithms, which can help the future researchers to know them well, and design the proper distributed version. (2) Some index-based method [123] in our study is able to couple with various estimators, which

¹This work is published as [VLDB’19] Xiangyu Ke, Arijit Khan, and Leroy Lim Hong Quan, *An In-Depth Comparison of s - t Reliability Algorithms over Uncertain Graphs*.

Figure 2.1: The broad spectrum of reliability problem over uncertain graphs

is also applicable to the distributed algorithms. (3) For the simplified versions of the s - t reliability problem [41,104,158], the s - t reliability computation over simplified graphs still relies on basic estimators in our studies. If a method was designed for a specific kind of reliability query (e.g. distance-constrained [84]), it can be generalized to the fundamental s - t query, thus we include the algorithm in our study [84,194]. In particular, various sampling and indexing-based efficient algorithms were proposed in the literature. Estimation of reliability in uncertain graphs has its beginnings with the usage of Monte Carlo (MC) sampling [63]. Subsequently, more advanced sampling methods were proposed in the form of recursive samplings [84,114], sampling with equivalent distribution [116], and shared possible world samples [194], as well as other indexing methods [123]. *With the wide range of algorithms available for estimating the s - t reliability over uncertain graphs, there is an urgent need to realize their trade-offs, and to employ the best algorithm for a given scenario.*

2.1 s - t Reliability in Uncertain Graphs

An uncertain graph $\mathcal{G}$ is a triple (V, E, P) , where V is a set of n nodes, $E \subseteq V \times V$ is a set of m directed edges, and $P : E \rightarrow (0, 1]$ is a probability function that assigns a probability of existence to each edge in E .

The bulk of the literature on uncertain graphs and device networks reliability assumes the existence of the edges in the graph independent from one another, and interprets uncertain graphs according to the well-known *possible-world semantics* [24, 28, 78, 84, 141, 142, 154, 193]: an uncertain graph $\mathcal{G}$ with m edges yields 2^m possible deterministic

graphs, which are derived by sampling independently each edge $e \in E$ with probability $P(e)$. More precisely, a possible graph $G \sqsubseteq \mathcal{G}$ is a pair (V, E_G) , where $E_G \subseteq E$, and its sampling probability is:

$$Pr(G) = \prod_{e \in E_G} P(e) \prod_{e \in E \setminus E_G} (1 - P(e)) \quad (2.1)$$

For a possible deterministic graph G , we define an indicator function $I_G(s, t)$ to be 1 if there is a path in G from a source node $s \in V$ to a target node $t \in V$, and 0 otherwise. The probability that t is reachable from s in the uncertain graph $\mathcal{G}$, denoted by $R(s, t)$, is computed as:

$$R(s, t) = \sum_{G \sqsubseteq \mathcal{G}} I_G(s, t) Pr(G) \quad (2.2)$$

The number of possible worlds $G \sqsubseteq \mathcal{G}$ is exponential in the number of edges, which makes the exact computation of $R(s, t)$ infeasible even for modestly-sized networks. In fact, the s - t reliability computation is a prototypical $\#\mathbf{P}$ -complete problem [10, 171].

Due to intrinsic hardness, we tackle the reliability estimation problem from approximation and heuristic viewpoints. In particular, we shall examine six sampling and indexing-based efficient algorithms that were proposed in recent literature as follows: **(1)** Monte Carlo (MC) sampling [63]; **(2)** Indexing via BFS sharing [194]; **(3)** Recursive sampling [84]; **(4)** Recursive stratified sampling [114]; **(5)** Lazy propagation sampling [116]; and **(6)** Indexing via probabilistic trees [123].

2.2 Reliability Estimation Algorithms

2.2.1 MC Sampling

In the basic Monte Carlo (MC) sampling, we first sample K possible worlds $G_1, G_2, \dots, G_K$ of the uncertain graph $\mathcal{G}$ according to independent edge probabilities. We then compute the reachability in each sampled graph G_i , and define $I_{G_i}(s, t) = 1$ if t is reachable from s in G_i , and 0 otherwise. Given this, we have the MC sampling estimator:

$$R(s, t) \approx \hat{R}(s, t) = \frac{1}{K} \sum_{i=1}^K I_{G_i}(s, t) \quad (2.3)$$

This is also known as the hit-and-miss Monte Carlo. The basic sampling estimator $\hat{R}(s, t)$ is an *unbiased estimator* of the s - t reliability, i.e., $E(\hat{R}(s, t)) = R(s, t)$, and its variance can be determined due to Binomial distribution $\sim B(K, R(s, t))$ [63, 84].

$$\begin{aligned} \text{Var}(\hat{R}(s, t)) &= \frac{1}{K} \cdot R(s, t) \cdot (1 - R(s, t)) \\ &\approx \frac{1}{K} \cdot \hat{R}(s, t) \cdot (1 - \hat{R}(s, t)) \end{aligned} \quad (2.4)$$

It is possible to derive bounds on the number of MC samples needed to provide a good estimate for the s - t reliability problem. It was shown in [141] by applying the Chernoff bound that with number of samples $K \geq \frac{3}{\epsilon^2 R(s, t)} \ln\left(\frac{2}{\lambda}\right)$, we can ensure the following.

$$\Pr\left(\left|\hat{R}(s, t) - R(s, t)\right| \geq \epsilon R(s, t)\right) \leq \lambda \quad (2.5)$$

The time complexity to generate K possible worlds is $\mathcal{O}(mK)$. In each possible world, the reachability can be determined by performing a breadth-first search (BFS) from the source node. Each BFS requires $\mathcal{O}(m+n)$ time. Therefore, the overall time complexity of MC sampling based reliability estimation is $\mathcal{O}(K(m+n))$. In essence, one may combine MC sampling with BFS from the source node for improved efficiency [84, 97]. It means that an edge in the current possible world is sampled only upon request. This avoids sampling of many edges in parts of the graph that are not reached with the current BFS, thus increasing the chance of an early termination.

In practise, MC sampling can be inefficient over large-scale networks due to two reasons.

- For each s - t reliability query, we need to generate K possible worlds via sampling. Based on empirical evidences from state-of-the-art works [84, 94, 141] as well as according to our own experimental results, K can be in the order of thousands to achieve a reasonable accuracy. However, as correctly pointed out in [137, 194], this sampling procedure does not contribute to the reliability estimation process directly. For example, one can pre-compute these K possible worlds in an offline manner to further improve the efficiency of online s - t reliability estimation.
- There could be a significant overlap in structures of different possible worlds [84, 193, 194]. Unfortunately, the reliability estimation via basic MC sampling performs a separate BFS over each possible world, therefore it cannot take advantage of the common substructure across various possible worlds.

Figure 2.2: Compact structure with five possible worlds: BFS sharing

2.2.2 Indexing via BFS Sharing

Zhu et al. [194] developed an *offline sampling* method to generate K possible worlds: $G_1, G_2, \dots, G_K$. In order to minimize the storage overhead, they proposed a bit-vector based compact structure, as depicted in Figure 2.2. It essentially stores only one graph $G = (V, E)$ with the same set of nodes and edges as the input uncertain graph $\mathcal{G}$. However, each edge e in G has a bit-vector of size K — its i -th bit represents whether the edge e is present in the sampled graph G_i or not.

Given an s - t reliability query, [194] performs BFS over this compact graph structure, which is equivalent to doing BFS traversals in parallel across the pre-computed possible worlds. We attach an additional bit vector I_v with each node v that keeps track of the possible worlds in which v is reachable from s . Initially, $I_s = [1 \ 1 \ \dots \ 1]$ and $I_v = [0 \ 0 \ \dots \ 0]$ for all $v \neq s$. Let us also denote by U the set of visited nodes based on BFS. Initially $U = \{s\}$. At each step, when we find an unexplored node v that is an out-neighbor of at least one node u in U , we insert v into U . We update the bit vector I_v to include the possible worlds where all such v 's are reachable from s . Before proceeding to the next step of BFS, one may note that if v has some out-neighbor w that is already in U , we may need to update I_w . Specifically, let G_i be a possible world that is currently in I_v , but not in I_w . Then, we should also include G_i in I_w . Such changes in I_w may in turn affect I_z , where z is an out-neighbor of w , and z is also in U . In general, we proceed to the next step of BFS only after finishing these cascading updates. Finally, the number of 1's in I_t , divided by K , provides the MC estimation of s - t reliability.

Clearly, BFS sharing has the same variance as the basic MC. However, by generating the K possible worlds in an offline manner, [194] reduces the online s - t reliability estimation time. On the other hand, *unlike the MC sampling approach, in this case*

no early termination of BFS is possible, even when the target node t is reached earlier. This is due to performing the required cascading updates in this method. It often makes BFS Sharing even more time consuming than MC, which is evident in our experimental results.

Note that the original algorithm in [194] was developed to identify the top- k target nodes having the maximum reliability from a given source node. However, as we discussed above, one can trivially update their technique to estimate the reliability of a given pair.

Our correction in complexity analysis. The offline index building time complexity is $\mathcal{O}(Km)$, and index storage space is $\mathcal{O}(n + Km)$, where $\mathcal{O}(Km)$ is the storage due to edge bit vectors. We load all edge bit vectors in memory during online query processing to improve efficiency. Besides, n node bit vectors of size $\mathcal{O}(Kn)$ are created during online query processing. BFS sharing method has online reliability estimation time complexity $\mathcal{O}(K(m + n))$: Due to cascading updates, each node and edge can be visited at most K times. Notice that in the original paper [194], it was stated that the online query time is independent of K . As we reasoned, however, the running time increases linearly with K due to cascading updates. This is also confirmed by our empirical results (e.g., notice in Tables 2.5 and 2.6, the running times of BFS sharing increases for larger K over the same dataset).

2.2.3 Recursive Sampling

Recursive sampling, which was proposed in [84], improves on MC sampling by considering the two following factors.

- When some edges are missing in a possible world, the presence of other edges might no longer be relevant with respect to certain s - t reliability queries. Hence, those edges can be skipped from sampling and query evaluation process.
- Many possible worlds share a significant portion of existing or missing edges. Hence, the reachability checking cost could be shared among them.

The basic approach, which follows a divide-and-conquer technique, is given below. A very similar algorithm, called the Dynamic MC sampling, was developed in [193].

We start from the source node s , and say that s is already *reached*. An edge e is *expandable* if it starts from a *reached* node. We randomly pick an extendable edge e , then sample the existence of e for K iterations. The next step is to divide the samples into two groups: one group with e existing and another group with e not existing. In the first group, we may reach a new node w via e , and in that case, more edges become *expandable*. For both groups, we repeat the process of picking a random *expandable* edge, sampling its existence, and dividing the group into smaller batches.

Formally, assume that $E_1 \subseteq E$ be the set of included edges and $E_2 \subseteq E$ be the set of not-included edges in one group (referred to as a *prefix group* in [84]) at some intermediate stage of our method. Let us denote this group by $\mathcal{G}(E_1, E_2)$, i.e., the set of possible worlds of $\mathcal{G} = (V, E, p)$ which contains all edges in E_1 , and no edges in E_2 . Clearly, $E_1 \cup E_2 \subseteq E$ and $E_1 \cap E_2 = \phi$. The *generating probability* of the group $\mathcal{G}(E_1, E_2)$ can be defined as:

$$Pr(\mathcal{G}(E_1, E_2)) = \prod_{e \in E_1} P(e) \prod_{e' \in E_2} (1 - P(e')) \quad (2.6)$$

The s - t reliability of a group $\mathcal{G}(E_1, E_2)$ is defined as the probability that t is reachable from s conditioned on the existence of the group $\mathcal{G}(E_1, E_2)$, i.e.,

$$R_{\mathcal{G}(E_1, E_2)}(s, t) = \sum_{G \subseteq \mathcal{G}(E_1, E_2)} I_G(s, t) \times \frac{Pr(G)}{Pr(\mathcal{G}(E_1, E_2))} \quad (2.7)$$

Next, one may verify that the following holds.

$$R(s, t) = R_{\mathcal{G}(\phi, \phi)}(s, t) \quad (2.8)$$

Also, for any edge $e \in E \setminus (E_1 \cup E_2)$,

$$\begin{aligned} R_{\mathcal{G}(E_1, E_2)}(s, t) \\ = P(e)R_{\mathcal{G}(E_1 \cup \{e\}, E_2)}(s, t) + (1 - P(e))R_{\mathcal{G}(E_1, E_2 \cup \{e\})}(s, t) \end{aligned} \quad (2.9)$$

We terminate aforementioned recursive procedure when either E_1 contains an s - t path with $R_{\mathcal{G}(E_1, E_2)}(s, t) = 1$, or E_2 contains an s - t cut with $R_{\mathcal{G}(E_1, E_2)}(s, t) = 0$.

The efficiency can further be improved by selecting the “best” *expandable* edge (i.e., edge e in Equation 2.9) at each iteration. In particular, by following the experimentally

Table 2.1: Stratum design for recursive stratified sampling

Stratum	e_1	e_2	e_3	$\dots$	e_r	e_{r+1}	$\dots$	e_m	Prob space
Stratum 0	0	0	0	$\dots$	0	*	$\dots$	*	Ω_0
Stratum 1	1	*	*	$\dots$	*	*	$\dots$	*	Ω_1
Stratum 2	0	1	*	$\dots$	*	*	$\dots$	*	Ω_2
$\dots$					$\dots$				$\dots$
Stratum r	0	0	0	$\dots$	1	*	$\dots$	*	Ω_r

optimal strategy in [84], we employ depth-first search (DFS) for the next edge expansion. We also find that this strategy works well in our experiments. Starting from the source node s , we start to explore its first neighbor (its next neighbor is explored only if there is no path to t which can be found going through the earlier ones), and then recursively visit the neighbors of this neighbor.

The aforementioned recursive sampling process has the same variance as the basic MC sampling. The variance can be reduced by eliminating the “uncertainty” of the existence of edge e in Equation 2.9. Let $\pi = R_{\mathcal{G}(E_1, E_2)}(s, t)$, $\pi_1 = R_{\mathcal{G}(E_1 \cup \{e\}, E_2)}(s, t)$, and $\pi_2 = R_{\mathcal{G}(E_1, E_2 \cup \{e\})}(s, t)$. Now, instead of directly sampling both the children nodes from the root (as suggested in Equation 2.9), we consider to estimate both π_1 and π_2 independently, and then combine them together to estimate π . Specifically, for K total samples in the root, we deterministically allocate K_1 of them to the left subtree (prefix group that includes edge e), and K_2 of them to the right subtree (prefix group that excludes edge e). It was shown in [84] that when the sample size allocation is proportional to the edge inclusion probability, i.e., $K_1 = P(e) \cdot K$ and $K_2 = (1 - P(e)) \cdot K$, the variance of the earlier recursive estimator can be reduced. Moreover, when a prefix group size K is below a pre-defined threshold, we use a non-recursive sampling method, such as the basic Monte Carlo or more sophisticated Hansen-Hurwitz estimator to sample the remaining edges [84].

The time complexity of recursive sampling estimator is $\mathcal{O}(na)$, where a is the average recursion depth, and is bounded by the diameter of the graph. Note that in the original paper by Jin et al. [84], recursive sampling was proposed to estimate distance-constrained reliability, that is, the probability that s is reachable to t within an input distance d . In this work, we adapted the proposed approach to compute the s - t reliability without any distance constraint.

2.2.4 Recursive Stratified Sampling

Li et al. [114] developed an alternative divide-and-conquer approach to measure reliability, called the recursive stratified sampling (RSS). By using a stratification method that partitions the probability space Ω into $r + 1$ non-overlapping subspaces $(\Omega_0, \dots, \Omega_r)$ via selecting r edges, they proceed to fix the states for certain edges in each stratum. For example, as seen in Table 2.1, in stratum 0, we set the status of r selected edges as 0, while leaving the rest of edges as undetermined. Subsequently, in stratum i ($1 \leq i \leq r$), we set the status of edge i to 1, the status of those before it as 0, and those after it as undetermined.

Let T be the set of selected r edges via BFS from the source node s , and $X_{i,j}$ be the corresponding status vector of the j -th ($1 \leq j \leq r$) selected edge in stratum i . Then, the probability of a possible graph in stratum i is given by:

$$\begin{aligned} \pi_i &= Pr[G_P \in \Omega_i] \\ &= \prod_{e_j \in T \wedge X_{i,j}=1} P(e_j) \prod_{e_j \in T \wedge X_{i,j}=0} (1 - P(e_j)) \end{aligned} \quad (2.10)$$

We then set the sample size of stratum i to $K_i = (\pi_i \cdot K)$, where K being the total sample size. The algorithm recursively applies the sample size to each stratum and simplifies the graph. It terminates when the sample size of a stratum is smaller than a given threshold. Reliability is then calculated by finding the sum of the reliabilities in all subspaces. The pseudocode is presented in our extended version [93].

It was shown in [114] that the time complexity of recursive stratified sampling is same as that of the MC sampling, i.e., $\mathcal{O}(K(m + n))$, while the variance of the estimator is significantly reduced.

2.2.5 Lazy Propagation Sampling

Li et al. proposed the lazy propagation sampling [116] that aims to bypass MC sampling's requirement of probing a large number of edges. In particular, if an edge has a low probability, it will remain un-activated in most of the samples; and therefore, those probings could be "unnecessary". In contrast, lazy propagation sampling estimates reliability by avoiding unnecessary probing of edges as much as possible. To achieve this,

Algorithm 1 Lazy Propagation Sampling

Input: source node s , target node t in uncertain graph $\mathcal{G} = (V, E, P)$, $K = \# \text{samples}$

Output: reliability $R(s, t)$

```

1:  $r \leftarrow 0$ , set all nodes not initialized
2: for  $i = 1$  to  $K$  do
3:   if  $s == t$  then
4:      $r \leftarrow r + 1$ 
5:     goto line 29
6:   end if
7:    $h \leftarrow \phi$ , set all nodes not visited
8:    $h \leftarrow \{s\}$  /* set of visited nodes */
9:   mark  $s$  visited
10:  while  $h \neq \emptyset$  do
11:     $v \leftarrow h.pop()$ 
12:    if  $v$  is not initialized then
13:       $c_v \leftarrow 0$ ,  $h_v \leftarrow \emptyset$ , mark  $v$  initialized
14:      for  $nbr \in v$ 's neighbor sets do
15:         $X(nbr) \leftarrow$  geometric r.v. instance
16:         $h_v \leftarrow h_v \cup \{\langle nbr, X(nbr) + c_v \rangle\}$ 
17:      end for
18:    end if
19:    while  $h_v.top() == c_v$  do
20:       $\langle nbr, X(nbr) \rangle \leftarrow h_v.pop()$ 
21:       $h \leftarrow h \cup \{nbr\}$  if  $nbr$  is not visited
22:      mark  $nbr$  visited
23:       $X'(nbr) \leftarrow$  geometric r.v. instance
24:       $h_v \leftarrow h_v \cup \{\langle nbr, X'(nbr) + c_v + \mathbf{1} \rangle\}$  /* our correction in bold */
25:      if  $nbr == t$  then
26:         $r \leftarrow r + 1$ 
27:        goto line 29
28:      end if
29:    end while
30:     $c_v \leftarrow c_v + 1$ ;
31:  end while
32: end for
33: return  $\frac{r}{K}$ 

```

the algorithm employs a geometric distribution per edge, and probes an edge only when it will be activated. The geometric random instance determines, based on the original edge probability, the number of sampling instances before the edge is activated. This

means that the algorithm will predict how many k worlds are sampled before an edge exists. In doing so, the number of times an edge is probed is reduced by a factor of $\frac{1}{p(e)}$ in expectation. It was proved in [116] that there is no statistical difference between using lazy sampling and the classic MC sampling. In other words, lazy sampling has the same variance as that of MC sampling, while improving on the efficiency.

As the original algorithm [116] was developed for personalized social influential tags exploration, we adapt it to measure reliability over uncertain graphs. We show the complete procedure in Algorithm 1. It initializes a heap h_v for every visited node v , and pushes v 's each out-neighbor nbr with a geometric random instance, i.e., $\langle nbr, X(nbr) \rangle$ into h_v (lines 12-18). It also maintains a counter c_v to keep track of the number of times v has been visited. Once a random instance in h_v is equal to c_v , the corresponding neighbor will be probed, and then a new random instance is generated to decide the next time to probe the neighbor (lines 19-29).

Our correction in the algorithm. In the original paper [116], it is $h_v \leftarrow h_v \cup \{\langle nbr, X'(nbr) + c_v \rangle\}$ at line 24, which, in fact, incurs error in reliability estimation². This is because node nbr is probed in the current round (since $h_v.top() == c_v$ was true at line 19), and now line 23 aims at assigning a new geometric random instance $X'(nbr)$ with nbr . This new geometric random instance $X'(nbr)$ indicates that after how many times of failure, nbr will be visited again following node v , starting from the next round. Therefore, at line 24, the value of counter c_v shall be that of the next round, i.e., one needs to add c_v+1 with $X'(nbr)$ (as opposed to adding only c_v specified in the original paper [116]).

We shall demonstrate the error in [116] and our correction with the following example.

Example 2.1. *Considering the graph in Figure 2.3: Node 1 is the source node, and node 3 the target node. In the first round, $c_1 = 0$ and we initialize node 1, push its neighbor node 2 into h_1 . Suppose that the geometric random instance $X(2)$ for 2 is generated as 0. Then, node 2 exists in the first possible world (i.e., the condition in line 19 holds true for node 2), and the algorithm proceeds to line 20-24. Now, if we follow $h_v \leftarrow h_v \cup \{\langle nbr, X'(nbr) + c_v \rangle\}$ at line 24 as in the original paper [116], we shall encounter at least one of the two following errors.*

²Yuchen Li, first author of [116], acknowledged this issue in email communication.

Figure 2.3: Toy graph
for Example 2.1

Figure 2.4: Reliability estimated by MC, Lazy Propagation (LP), and Lazy Propagation+ (LP+) at convergence. Reliability is reported as the average over 100 s - t pairs. For experimental setting, see Sec. 2.4.

(1) Assume that at line 24, the new geometric random instance $X'(2) > 0$, e.g., $X'(2) = 1$. It indicates that in the next possible world, edge $1 \rightarrow 2$ must not exist. However, when following the original algorithm as in [116], $\langle \text{Node } 2, 1 \rangle$ will be stored in h_1 , since $c_1 = 0$ at present. When sampling the next possible world, $c_1 = 1$ and node 2 will be probed again, which is incorrect. Clearly, this is an overestimation, and it will result in higher estimated reliability from source to target.

(2) In the other case, when the new geometric random instance $X'(2) = 0$ at line 24, it means that in the next possible world, edge $1 \rightarrow 2$ must exist. By following the original algorithm as in [116], $\langle \text{Node } 2, 0 \rangle$ will be pushed in the heap h_1 and would be placed at the top. However, when sampling all subsequent possible worlds, $c_1 \geq 1$ and $\langle \text{Node } 2, 0 \rangle$ at the top of h_1 will always stop the algorithm to enter lines 20-24, which makes node 1 not expandable anymore, and thus incurs an underestimation error.

In practice, the first type of error, i.e., overestimation happens in most cases, and the original method in [116] estimates much higher reliability. Thus, we update the algorithm at line 24, as shown in Algorithm 1 to avoid such errors. We denote it as LP+, and the original one as LP. Our experimental results in Figure 2.4 confirms that LP indeed estimates much higher reliability than Monte Carlo (MC), whereas LP+ estimation is close to that of MC.

2.2.6 Indexing via Probabilistic Trees

Maniu et al. [123] proposed ProbTree index to improve the efficiency of s - t reliability queries. The method builds a tree index structure, called ProbTree, from the input uncertain graph G . Next, given an s - t reliability query q , an equivalent graph $G(q)$ is created from the ProbTree index structure, and MC sampling is performed on $G(q)$. Clearly, if $G(q)$ can be generated quickly, and if $G(q)$ is smaller than G , then executing the query on $G(q)$ would be faster.

In [123], three index structures were developed. Among them, we employ FWD tree (an abbreviation for fixed-width tree decomposition) because (a) its index building time, space, and query processing time are linear in the input graph size, and (b) the index structure is lossless (i.e., produces good-quality results) for a tree decomposition width $w \leq 2$. In fact, as reported in [123], as well as found in our experiments, FWD tree with $w=2$ produces high-quality results, and exhibits good efficiency.

FWD tree index building has three phases: fixed-width tree decomposition, building of the FWD index tree, and pre-computation of reliability. The first phase is an adaption of the algorithms in [161, 181], which performs a *relaxed* tree decomposition with a fixed width w . At each step, a node having degree at most w is chosen, marked as *covered*. A bag is created to contain this node and its neighbors, along with the probabilistic edges between them in G . Then, the covered node is removed from G and a clique between its neighbors is added into G . This process repeats until there are no such nodes left. Finally, the rest of the uncovered nodes and the remaining edges are copied in the root of the tree.

The second phase is the creation of the FWD index tree. Each bag is visited in its creation order, and their parents are defined as the bag whose node set contains all uncovered nodes of the visited bag. If no such bag exists, then the parent of the bag will be the root of the tree. In the final phase, we need to compute $R(v_1, v_2)$ for each pair (v_1, v_2) in each bag. It follows a bottom-up manner. For each bag B , it collects the computed reliability from B 's children, and combine them with information within current bag to compute the current reliability for all node pairs in B . When $w \leq 2$, it is possible to pre-compute the correct probability distributions between node pairs, thereby making the index structure lossless.

Figure 2.5: Example of FWD ProbTree ($w=2$) construction

When conducting an s - t query, if the root contains both s and t , the root is the query graph. Otherwise, it searches from the bottom to find the bags containing s or t as the covered node, respectively. Then it propagates them up all the way to the root, and merges them as a combined graph for query answering. By doing so, all irrelevant branches will not be included in the graph returned. An example of index building and query processing is presented below.

Example 2.2. In Figure 2.5(b), we show the FWD index tree for the input uncertain graph in Figure 2.5(a). At first, node 3 has the smallest degree 1 and is chosen. A bag (E) is created to contain it, together with its neighbor, node 4. Then, we put all the unmarked edges between them into this bag and mark them. After the construction of this bag, we remove node 3 (the selected node) from the original graph. Now, no node has degree 1, then we consider degree 2. Due to the removal of node 3, node 4 has degree 2 and can be selected. We treat it in the same way and obtain the second bag (labelled as (C) in Figure 2.5(b)). Since there is no edge between node 0 and node 6 (neighbors of node 4) in the original graph, a new edge (0,6) is added. We repeat the aforementioned procedure until no more bag can be created. Then, we traverse them in the creation order (E), (C), (F), (D), (B), (A), and decide their parent-child relationships using lines 20-24.

For computing reliability in each bag, it follows a bottom-up manner as described before. For example, in bag (D), the reliability from node 6 to node 1 is collected from its child, bag (F), and is computed as: $1 - (1 - 0.75) * (1 - 0.5 * 0.5) = 0.8125$.

For reliability from node 1 to node 2, we find bags (B) and (D) (which have these nodes as cover), and finally bags (A), (B), and (D) are merged as the query graph. Bags (C) and (E) are irrelevant to this query. Bag (F)’s information about the query node is already contained in bag (D). Thus, it will not be involved as well.

Following [123], the time complexity of FWD probtree index construction is linear in the number of nodes in the graph, and the complexity of pairwise reliability computations in each bag is $\mathcal{O}(w^2d)$, where w is the tree width ($w = 2$ to ensure loseless index), and d is the diameter of graph. The space complexity of FWD probtree index is $\mathcal{O}(|E|)$. During online s - t query processing, the cost of retrieving s and t from the FWD probtree is linear in the tree depth. The reliability estimating cost is the same as MC sampling, $\mathcal{O}(K(n' + m'))$, but with smaller number of nodes and edges.

Our adaptation in complexity. The original *ProbTree* index is designed to support multiple kinds of s - t queries, e.g., reliability query, shortest path query, etc. Therefore, it pre-computes the distance probability distribution between each node in every bag. However, if only considering s - t reliability query, we can just compute and store the edge probability information, regardless of distance. Since $w = 2$, for each pair of nodes in the current bag, there can exist at most 2 paths in its children. One can easily collect and aggregate them by $1 - (1 - p_1)(1 - p_2)$, where p_1 and p_2 are the probability of these two paths. An illustration can be found in Example 2.2. The complexity of pairwise reliability computations in each bag can be reduced from $\mathcal{O}(w^2d)$ to $\mathcal{O}(w^2)$. Empirically, on our largest *BioMine* dataset, we can reduce the index building time from 4062 seconds to 2482 seconds.

2.2.7 Other Related Work

The large spectrum of the reliability problem is categorized in Figure 2.1. Further research on reliability emphasized on polynomial-time upper/lower bounds to reliability problems [11, 24, 28, 69, 105, 142]. Recently, efficient *distributed* algorithms have been developed for reliability estimation [42, 196]. Some orthogonal directions include finding

one “good” possible world [137, 158], considering the most probable path [41, 104], as well as adaptive edge testing [64–66] and crowdsourcing [119] for reducing uncertainty. As discussed earlier in Sections 2.2.1 – 2.2.6, in this work we focus on sampling and indexing based sequential algorithms for s - t reliability estimation.

Other related queries over uncertain graphs include nearest neighbors [115, 141], shortest paths [188, 195], reliable set [97, 194], conditional reliability [98], distance-constrained reachability [84], probabilistic road network queries [80], and influence maximization [94], among several others. Many of the efficient sampling and indexing strategies that we investigate in this work can also be employed to answer such advanced queries.

2.3 Horizontal Comparison of Algorithms: Strengths and Weaknesses

Estimator variance. The mean squared error (MSE) measures the quality of an estimator. The lower the MSE, the better is an estimator. For unbiased estimator, the MSE is equal to the variance of this estimator. *All estimators compared in this work are proved to be unbiased in original papers. Statistically, BFS Sharing and Lazy Propagation shares same variance as basic MC Sampling. The two recursive methods, RHH and RSS, reduce the estimator variance by dividing the sample size according to either edge probability (e.g., RHH), or stratum probability (e.g., RSS).* We refer to Theorem 2 in [84], and Theorems 4.2 and 4.3 in [114] as the theoretical basis for variance reduction in RHH and RSS, respectively.

Running time per sample. For each sample, MC applies a BFS search from the source node s , each edge encountered exists with the probability on that edge. The BFS search terminates when the target node t is found (early termination), or no new node can be expanded. In contrast, while BFS Sharing follows the same BFS search strategy; no early termination is allowed. The existence of each edge in every sample has been determined offline (via indexing). Therefore, *BFS Sharing tends to sample more edges than MC in each sample, while the time cost of checking the existence of each edge is saved due to indexing.*

Lazy propagation employs geometric distribution to determine the existence of an edge in samples, that is, each time when an edge is probed during the BFS search in

one sample, it decides after how many samples this edge will exist again. Therefore, unnecessary edge probing is avoided.

ProbTree decomposes the graph and pre-computes the reliability information beforehand, thus allowing running online BFS on a smaller graph to improve efficiency. RHH and RSS recursively simplify the graph, and apply MC sampling on the resulted graph when sample size is smaller than a threshold, or the graph is fully simplified. The improvement of the running time per sample depends on how much the graph can be simplified. Additional time cost may arise from graph simplification task and the recursive procedure. In all, *Lazy Propagation and ProbTree can reduce the time cost of investigating a sample when compared with MC, while BFS Sharing, RHH and RSS have both improvements and additional costs.* The detailed experimental study can be found in Section 2.4.5.

Total running time. To achieve the same estimator variance (e.g., same quality), MC, BFS Sharing, Lazy Propagation, and ProbTree would require same sample size, thus the Lazy Propagation and ProbTree shall improve the efficiency (due to lower running time per sample) compared to MC. For RHH and RSS, less samples are needed, thereby also improving the efficiency compared to MC. *With all these, the final efficiency comparison for Lazy Propagation, ProbTree, RHH, and RSS is unknown. Therefore, our experimental analysis is critical.*

Memory usage. In addition to the original graph, the only memory cost of MC is the current sample, and the count of samples where t is reachable from s . For BFS Sharing, instead of original graph, the whole index shall be kept in main memory. Each edge in BFS Sharing index is a vector of Boolean values, while in the original graph only a single value is stored. Compared with MC, Lazy Propagation additionally requires a global counter for each node, and a geometric random instance heap for its neighbors. The ProbTree index size is linear in the original graph size. Both RHH and RSS store the whole recursive stack, and simplified graph instances in main memory, which makes them more memory intensive.

Indexing cost, re-computation, and adaptability. Both BFS Sharing and ProbTree index sizes are linear in the original graph size. ProbTree index is generally comparable to the original graph size, and is independent of the sample size K . However, BFS

Table 2.2: Properties of datasets

Dataset	#Nodes	#Edges	Edge Prob: Mean, SD, Quartiles
<i>LastFM</i>	6 899	23 696	0.29 ± 0.25 , {0.13, 0.20, 0.33}
<i>NetHept_uniform</i>	15 233	62 774	0.04 ± 0.04 , {0.001, 0.01, 0.10}
<i>AS_Topology</i>	45 535	172 294	0.23 ± 0.20 , {0.08, 0.21, 0.31}
<i>DBLP_0.2</i>	1 291 298	7 123 632	0.33 ± 0.18 , {0.18, 0.33, 0.45}
<i>DBLP_0.05</i>	1 291 298	7 123 632	0.11 ± 0.09 , {0.05, 0.10, 0.14}
<i>BioMine</i>	1 045 414	6 742 939	0.27 ± 0.21 , {0.12, 0.22, 0.36}

Sharing is also about linear in the sample size K . Thus, *BFS Sharing index is usually larger than ProbTree index, and BFS Sharing also requires more loading time into main memory. On the other hand, BFS Sharing index is easier to build than ProbTree index.* Only K boolean values shall be generated based on the edge probability for each edge. The ProbTree index requires decomposing the original graph and pre-computing the reliability information.

To ensure the independence among queries, index shall be updated between two successive queries for BFS Sharing, this is not necessary for ProbTree. Moreover, ProbTree has the adaptability to support estimators other than MC sampling, while BFS Sharing has no such potential.

2.4 Experimental Results

We conduct experiments to compare six state-of-the-art reliability estimation algorithms, and report the number of samples required for their convergence, estimator accuracy, variance, running time, and memory usage using medium and large-scale, real-world network datasets. We obtain the source code of ProbTree [123] from the authors, which is written in C++, and we further optimize their index building method as discussed in the paragraph “Our improvement in complexity” in Section 2.2.6. We implement and well-optimize other five algorithms in C++, and perform experiments on a single core of a 100GB, 2.40GHz Intel Xeon E5-4650 v2 server. *Our datasets and source code are available at: <https://github.com/5555lan/RelComp>.*

2.4.1 Environment Setup

We downloaded five real-world networks (Table 5.2). Many of them have been extensively used in past research on uncertain graphs, including reliability estimation [84, 97, 114,

116, 141, 193, 194]. The detail descriptions about these datasets and the edge probability models can be found in our full paper [93].

2.4.1.1 Parameters Setting

For each dataset, 100 distinct s - t pairs are generated as follows. For a specific graph, we first select 100 different source nodes, uniformly at random. For a source node, we next perform BFS up to 2 hops. Among these visited nodes we select one target node that is 2-hop away from s , uniformly at random. These 100 s - t pairs are used consistently for all six competing methods over that dataset. All our results are averaged over these 100 s - t pairs.

Notice that we select s - t pairs with shortest-path distance = 2 hops, because if they are closer, their reliability would usually be higher. On the other hand, if some s - t pairs are far apart, their reliability would be naturally small. Nevertheless, we also demonstrate experimental results by varying the shortest-path distance between s - t pairs in our extended version [93].

The initial K , i.e., #samples considered in all algorithms is 250. We then increase K by a step of 250 till convergence is reached. (we refer to Section 2.4.1.2 for discussion on convergence).

For recursive estimators [84, 114], a few additional parameters need to be defined. For recursive sampling, we set the prefix group size threshold to be 5 as per [84]. For recursive stratified sampling, we set $r = 50$ (i.e., #selected edges in each stratum) as recommended in [114]. We find that these parameter values also work well in our experimental setting (For details, we refer to our sensitivity analysis results in the extended version [93]). Following [84, 114], we refer to recursive sampling and recursive stratified sampling as RHH and RSS, respectively.

2.4.1.2 Performance Metrics

Variance: Unlike MC-based estimators [63, 116, 123, 194], which report reachability from s to t as 1 or 0 in each sample; both recursive sampling algorithms [84, 114] estimate reliability in a holistic manner by considering *all* K samples. Thus, following [84, 114], we compute variance of an estimator by repeating experiments with the current number

of samples (K). Given an estimator and K , we repeat querying each s - t pair, e.g., s_i - t_i for T times; and we obtain T estimation results for each pair s_i - t_i : $R_1(s_i, t_i, K)$, $R_2(s_i, t_i, K)$, $\dots$, $R_T(s_i, t_i, K)$. We calculate the variance as follows.

$$V(s_i, t_i, K) = \frac{1}{T-1} \sum_{j=1}^T (R_j(s_i, t_i, K) - \bar{R}(s_i, t_i, K))^2 \quad (2.11)$$

Here, $\bar{R}(s_i, t_i, K)$ is the average value of these T estimation results, for a fixed K . Following [84], we set $T = 100$. We compute this variance for all 100 s - t pairs, and the average variance for this estimator, with given K , is:

$$V_K = \frac{1}{100} \sum_{i=1}^{100} V(s_i, t_i, K) \quad (2.12)$$

Analogously, we define average reliability for an estimator, with given K , as:

$$R_K = \frac{1}{100} \sum_{i=1}^{100} \bar{R}(s_i, t_i, K) \quad (2.13)$$

Based on our experiments, we find that as one increases K , the average variance V_K monotonically decreases. However, we also notice that, with same K , V_K varies for different datasets and estimators. This is primarily because the average reliability R_K is different for different datasets and estimators, even for same K . Thus, it is *difficult* to fix a uniform threshold on the average variance V_K which could define convergence in all cases.

Instead, we systematically consider the ratio $\rho_K = \frac{V_K}{R_K}$ to decide convergence of an estimator over a given dataset. The ratio of variance to mean is also known as the **index of dispersion**, which is a *normalized* measure of the dispersion of a dataset. Dispersion is close to zero if all the data are similar, and increases as the data become more diverse. Given a dataset and an estimator, we keep increasing K (in steps of 250), and when $\rho_K = \frac{V_K}{R_K} < 0.001$, we say that the convergence has been reached for that estimator and over that dataset. For that estimator and dataset, we report average reliability at that specific value of K , since this estimation is robust.

Relative error: By following [84, 123, 194], we report relative error (RE) of reliability estimation for an estimator with respect to MC sampling. This is computed as follows.

$$RE_K = \frac{1}{100} \sum_{i=1}^{100} \frac{|\bar{R}(s_i, t_i, K) - \bar{R}_{MC}(s_i, t_i, Convergence)|}{\bar{R}_{MC}(s_i, t_i, Convergence)} \quad (2.14)$$

Figure 2.6: Comparison of estimator variance and convergence. The symbol ρ_K on Y-axis denotes the ratio $\frac{V_K}{R_K} = \frac{\text{average variance at \#samples}=K}{\text{average reliability at \#samples}=K}$.

Here, $\bar{R}_{MC}(s_i, t_i, \text{Convergence})$ denotes the reliability of the pair s_i-t_i , as returned by MC sampling at convergence. On the other hand, $\bar{R}(s_i, t_i, K)$ denotes the reliability of the same pair, returned by the specific estimator, at $\#samples=K$.

Online and offline efficiency and memory usage: In regards to online querying, we report (a) total time cost to answer an s - t reliability query, (b) average time cost per sample (i.e., total time cost/ $\#samples$), and (c) memory usage for all algorithms. In addition, for index-based methods, i.e., BFS sharing [194] and ProbTree [123], we report their (i) time costs for creating index, (ii) time costs for loading index in main memory, and (iii) index sizes.

2.4.2 Estimator Variance and Convergence

We report our empirical findings on estimator variance and convergence in Figure 2.6 and 2.7, which are summarized below.

(1) Figure 2.6 depicts that among six competing estimators, four MC-based estimators: Monte Carlo (MC) sampling, BFS Sharing, ProbTree, and Lazy Propagation (LP+) exhibit nearly similar characteristics in estimator variance. The other two recursive estimators: Recursive Sampling (RHH) and Recursive Stratified Sampling (RSS) share similar estimator variance between them.

Figure 2.7: Comparison of average reliability returned by each estimator in regards to average reliability returned by MC sampling at $K = 10\,000$ (shown as horizontal dashed line), *BioMine*.

In particular, BFS Sharing performs the sampling of MC offline by building indexes, and this does not change the estimator variance compared to MC. Another index-based method, ProbTree decomposes the original graph into “bags” and re-organizes them in a tree structure. When processing an online query, it generates a smaller but equivalent graph from the tree index, by ignoring irrelevant branches. We apply MC (as the original paper [123] did) to estimate the s - t reliability on this smaller but equivalent graph. Hence, the estimator variance remains same. For LP+, it utilizes the geometric distribution to decide the existence of an edge in samples, which is statistically equivalent [116] to MC; therefore results in similar variance. Overall, our empirical results confirm them.

For recursive estimators, [114] discussed that RHH is a special case of RSS when $r = 1$ (r denotes the number of selected edges in RSS). In Figure 2.6, we observe that the curves of RSS and RHH are close, but RSS always has lower variance than RHH.

(2) Our experimental results in Figure 2.6 confirm that *recursive estimators have lower variance than MC-based estimators*. By dividing the sample size according to either edge probability (e.g., RHH), or stratum probability (e.g., RSS), the uncertainty during sampling is reduced, which results in lower variance. It can be noted that *some later proposed methods, e.g., ProbTree and LP+ do not outperform RHH and RSS in estimator variance*.

(3) As discussed in Section 2.2.1, larger K ensures more accurate result by MC sampling. In Figure 2.7, we compare the reliability returned by each estimator with respect to that of MC sampling with very large K , e.g., $K = 10\,000$ (shown as horizontal dashed line) [179,187]. Figure 2.7 clearly presents that *the reliability estimated at variance*

Figure 2.8: The trade-off between relative error and running time/memory usage, *lastFM*.

convergence is very close to that at very large K , which indicates that estimator variance convergence can help find a sample size for high quality estimation.

(4) Past works on uncertain graphs generally employed some standard K (i.e., #samples) in their experiments, such as $K=500$ [141], 1 000 [84, 123], 2 000 [114, 194], or even 10 000 [94], while mentioning that they observed convergence for those specific K in their experimental setting. For future researchers and practitioners, we would like to emphasize that *there is no single K such that all estimators achieve convergence across all datasets*, which can be confirmed from our empirical findings. The K at convergence for every estimator on each dataset is listed in Tables 2.3-2.6 (observed from Figure 2.6). Interestingly for MC-based methods, K values at convergence are almost different over various datasets. Recursive estimators reach convergence with smaller K , e.g., $K=250$ on three out of five datasets. However, they require around 750 and 1000 samples on *NetHept* and *BioMine* datasets, respectively, for convergence. *Recursive methods generally converge with about 500 less samples than MC-based methods on the same dataset.*

(5) With a closer look, we find that *ProbTree has a slight improvement in estimator variance compared to three other MC-based methods*. This is because the ProbTree index pre-computes the reliability information contained in a bag’s children subtrees, and stores it in the current bag. By directly applying such pre-computed probabilities, one can reduce the uncertainty of sampling, and lower the estimator variance. In our experimental results on *NetHept* and *Biomine*, ProbTree requires around 250 less samples for convergence, when compared with other MC-based estimators.

Figure 2.9: The trade-off between relative error and running time/memory usage, *BioMine*.

2.4.3 Trade-off among Relative Error, Running Time, and Memory Usage

Figures 2.8 and 2.9 demonstrate the trade-off among estimation error, running time, and memory usage. The estimation error is provided as the relative error with respect to the reliability returned by MC sampling at variance convergence (as discussed in Section 2.4.1.2).

As shown in each subfigure (a) of Figures 2.8 and 2.9, when reaching the estimator variance convergence, the relative error rates of all six methods are (1) very close to each other; (2) below 2%; and (3) also converge. The estimator accuracy can benefit little by further increasing the sample size K . However, *the running time of all estimators grows about linearly with the sample size K . The memory usages of estimators are not very sensitive to the sample size.* Memory usage of MC, ProbTree, and LP+ nearly remains the same all the way. With larger number of samples, more indexes are required to be loaded into main memory by BFS Sharing, thus slightly increasing its memory cost. For recursive methods, larger K can allow larger recursion depths, thus more memory is consumed. *In summary, the estimator variance convergence can help us to find the sweet point which balances the estimator error and running time/memory consumption.*

More detailed accuracy and efficiency comparison can be found in Sections 2.4.4 and 2.4.5.

2.4.4 Estimator Accuracy

We compare relative errors (with respect to MC Sampling at variance convergence, Section 2.4.1.2) of all algorithms. Relative errors are reported (a) at convergence for that estimator, and (b) at $K=1000$. We report relative errors also at $K=1000$, since many

Table 2.3: Comparison of relative error (RE): *LastFM*

Estimator	At Convergence			At $K=1000$	
	K	R_K	RE (%)	R_K	RE (%)
MC	1000	0.1025	0.00	0.1025	0.00
BFS Sharing	1000	0.1030	0.97	0.1030	0.97
ProbTree	1000	0.1007	1.77	0.1007	1.77
Lazy Propagation	1000	0.1052	1.84	0.1052	1.84
Recursive (RHH)	250	0.1041	1.91	0.1040	1.88
Recur. Stratified (RSS)	250	0.1018	1.06	0.1020	1.07
Pairwise Deviation			0.53		0.52

Table 2.4: Comparison of relative error (RE): *DBLP_0.05*

Estimator	At Convergence			At $K=1000$	
	K	R_K	RE (%)	R_K	RE (%)
MC	750	0.2128	0.00	0.2136	0.17
BFS Sharing	750	0.2133	1.26	0.2134	1.15
ProbTree	750	0.2156	1.37	0.2162	1.40
Lazy Propagation	750	0.2154	1.52	0.2153	1.49
Recursive (RHH)	250	0.2114	1.39	0.2108	1.32
Recur. Stratified (RSS)	250	0.2124	1.36	0.2131	1.37
Pairwise Deviation			0.11		0.15

prior works [84, 114, 123, 194] did the same, irrespective of whether an estimator has converged or not. Our results are given in Tables 2.3-2.4 (more results can be found in our complete paper [93]).

(1) *When the value of K at convergence is larger than 1000 for a method, its relative error at convergence is smaller than that at $K=1000$ (e.g., see the results over *NetHept* and *BioMine*). In contrast, if the value of K at convergence is smaller than 1000 for a method, the relative error nearly remains the same when increasing K to 1000. Therefore, $K=1000$ is not a fair setting to compare the estimator accuracy across all estimators and datasets. Rather, K at convergence for that estimator ensures higher accuracy. Similarly, if the estimator has already converged at some $K < 1000$, the relative error would not reduce further, instead only the running time will increase for larger K .*

(2) *At convergence, relative errors for all six methods are low ($< 2\%$) and comparable (no common winner exists), which indicates that our approach of finding convergence (based on **index of dispersion**) ensures high accuracy. The best relative error rate on each dataset is below 1.5%.*

Table 2.5: Comparison of running time: $AS_Topology$

Estimator	At Convergence		At $K=1000$	Time Per
	K	Time (sec)	Time (sec)	Sample (ms)
MC	500	166	327	332
BFS Sharing	500	641	1235	1282
ProbTree	500	19	38	38
Lazy Propagation	500	20	39	40
Recursive (RHH)	250	12	45	48
Recur. Stratified (RSS)	250	14	55	56

Previous work [84] compared the relative error at $K=1000$, and concluded that RHH had better accuracy over MC-based methods. However, this does not hold when we consider K at convergence for respective methods. Empirically we find that $K=1000$ is sufficient for RHH and RSS to achieve convergence, while other methods may still require more samples (e.g., on *NetHept* and *BioMine*, in [93]) for their convergence. Only considering $K=1000$ is unfair to them. We notice that at convergence, there does not exist a common winner in regards to relative error among these six estimators.

We further calculate the pairwise deviation (D) of relative errors (RE) across different estimators on each dataset.

$$D = \frac{1}{5 * 6} \sum_{i=1}^6 \sum_{j=1}^6 |RE(i) - RE(j)| \quad (2.15)$$

Here, $RE(i)$ denotes the relative error of method i . One can observe that if a dataset requires more than 1000 samples for convergence (e.g., *NetHept* and *BioMine* in our full paper [93]), the pairwise deviation of relative errors among estimators over that dataset significantly decreases when increasing K from 1000 to that at convergence of respective estimators. These results further demonstrate that *at convergence, relative errors for all six methods are low and comparable; and measuring relative error at a specific value of K (e.g., $K=1000$) across all methods can be unfair to certain estimators.*

2.4.5 Estimator Efficiency

We present online running times for all methods both at convergence and at $K=1000$ in Tables 2.5-2.6 (more results can be found in our full paper [93]). The time cost per sample is reported in milliseconds at convergence. We summarize our observations below.

(1) *At convergence, RSS and RHH are the faster estimators.* RSS is similar or even faster than RHH on large datasets. However, it is slower than RHH on smaller datasets.

Table 2.6: Comparison of running time: *BioMine*

Estimator	At Convergence		At $K=1000$	Time Per
	K	Time (sec)	Time (sec)	Sample (ms)
MC	1500	4070	2678	2660
BFS Sharing	1500	12723	8644	8482
ProbTree	1250	600	482	480
Lazy Propagation	1500	770	520	513
Recursive (RHH)	1000	389	389	389
Recur. Stratified (RSS)	1000	375	375	375

ProbTree and *LP+* are in the middle range, and are often comparable to RHH. *MC* and *BFS Sharing* are much slower. *BFS Sharing* consumes about $4\times$ running time when compared with MC.

High efficiency of RHH and RSS (one s - t query finishes within 400 seconds for all datasets) is due to the following reasons. (i) When finally conducting sampling in RHH and RSS, the graph has been simplified. This is because certain parts of the graph will no longer be connected to the source node with the absence of some selected edges. Moreover, these estimators can avoid sampling if there exists a path connecting s and t with selected edges. (ii) Due to faster convergence, the sample size K required in *RHH* and *RSS* is smaller than that for other methods.

ProbTree and LP+ reduces about 80% of the running time of MC. ProbTree simplifies the graph with pre-computed indexes. LP+ speeds up the sampling procedure by avoiding unnecessary probing of edges, which is implemented with geometric distribution. Unlike RSS and RHH, they still require larger number of samples; and hence, their running times are in the middle range.

MC can be terminated early when the target node is visited. But for *BFS Sharing*, no early termination is possible (see our discussion in Section 2.2.2). In summary, *though BFS Sharing was proposed after MC, it can be $4\times$ slower than MC in regards to s - t reliability estimation. Moreover, ProbTree and LP+ were developed after recursive methods, but both recursive methods are generally more efficient due to their faster convergence.*

(2) When $K=1000$, there is no common winner in running time. RHH, RSS, ProbTree, and LP+ are comparable, and each of them wins once or twice out of all our datasets. This is because the advantage of requiring less samples (to reach convergence) does not hold here for RHH and RSS. Therefore, *comparing these methods at a fixed K*

Table 2.7: Summary and recommendation

Online Query Processing				
Method	Variance	Accuracy	Running Time	Memory
MC	★	★★★★	★★	★★★★★
BFS Sharing	★	★★★★	★	★★
ProbTree	★	★★★★	★★★★	★★★★
LP+	★	★★★★	★★★★	★★★★★
RHH	★★★★★	★★★★★	★★★★★	★
RSS	★★★★★	★★★★★	★★★★★	★
Index-related				
Method	Time (build)	Time (load)	Time (update)	Size
BFS Sharing	★★★★★	★★★★	★	★★★★
ProbTree	★★★★	★★★★★	★★★★★	★★★★★

(e.g., $K=1000$ as it is done in [123]) is unfair to both RHH and RSS. We notice that even at a fixed $K=1000$, MC and BFS Sharing are much slower than other estimators.

(3) We observe that *except for BFS Sharing, the time cost per sample is about the same at convergence and at $K=1000$, which indicates that their running time is linear in the sample size K* . For BFS Sharing, a small decrease in running time per sample can be viewed with increasing K , which implies that sharing BFS among samples help reduce the impact of sample size K . However, its online running time is still not independent of K . *With larger K , the total running time of BFS Sharing still increases (which is evident in Tables 2.5-2.6)*. Therefore, we do not agree with the claim in [194] that their online running time is independent of K . In fact, as we already discussed, its theoretical time complexity is still $\mathcal{O}(k(n+m))$, which is due to cascading updates (see the paragraph “Our correction in complexity analysis” in Section 2.2.2).

More experimental evaluation about index cost, online memory usage, adaptability, and sensitivity can be found in our full paper [93].

2.5 Summary and Recommendation

Summary. In this work we investigated six state-of-the-art sequential algorithms for s - t reliability estimation, corrected certain issues in these algorithms to further improve their performance, and conducted a thorough experimental evaluation.

Figure 2.10: The decision tree for selecting a proper reliability estimator under different scenarios

For estimator variance, both recursive methods: RHH and RSS exhibit significantly better performance than other four MC-based approaches: MC, BFSSharing, ProbTree, and LP+. Methods in the same category share very similar variance. In general, RSS is the best regarding variance, and achieves fastest convergence. To achieve convergence, there is no single sample size K that can be used across various datasets and estimators. Usually recursive methods require about 500 less samples than MC-based methods on the same dataset.

For accuracy, all methods have similar relative error ($<1.5\%$) at convergence. If K is set as 1000 for all estimators, some of them might not reach convergence, thus their relative errors can further be reduced by using larger K until convergence.

For efficiency, RHH and RSS are the fastest when running time is measured at convergence. When K is set as 1000, there is no common winner in terms of running time. Overall, the running times of RHH, RSS, ProbTree, and LP+ are comparable. BFSSharing is $4\times$ slower than MC, since it estimates all nodes' reliability from the source node.

The memory usage ranking (in increasing order of memory) is: $MC < LP+ < ProbTree < BFSSharing < RHH \approx RSS$.

Recommendation. Table 2.7 summarizes the recommendation level of each method according to different performance metrics. The scale is from 1 to 4 stars, and larger star number stands for higher ranking. Clearly, there is no single winner. Considering

various trade-offs, in conclusion we recommend ProbTree for *s-t* reliability estimation. It provides good performance in accuracy, online running time, and memory cost. Its index can help slightly reduce the variance, compared to other MC-based estimators. Notably, we adopted MC as ProbTree’s reliability estimating component (as the original paper [123] did). However, one can replace this with any other estimator (e.g., recursive estimators: RHH and RSS) to further improve ProbTree’s efficiency and to reduce its variance.

The decision tree shown in Figure 2.10 demonstrates our recommended strategy for estimator selection under different constraints. Following the branch with red tick, the better estimator(s) under the current condition can be determined. Notice that the path from the root to the leaf of ProbTree consists of all red ticks, indicating its applicability and trade-off capacity considering various factors.

Chapter 3

Reliability Maximization in Uncertain Graphs

In an uncertain graph setting, *Network Reliability* is a well-studied problem [10, 46, 142, 149, 171], which requires to measure the probability that a target node is reachable from a source node. In this work, I investigate the novel problem of adding a small number of edges in an uncertain network for maximizing the reliability between a given pair of nodes¹. Such edges are referred as *shortcut edges*, and the problem of identifying the best set of k edges as the *budgeted reliability maximization* problem. Our problem falls under the broad categories of uncertain networks design [177], optimization [139], and modification [133] problems, yet surprisingly this specific problem has not been studied in the past.

The budgeted reliability maximization problem is critical in the context of many physical networks, such as transportation and communication networks. In mobile ad-hoc networks, the connectivity between sensor nodes and devices is estimated using noisy measurements, thus leading to edges naturally associated with a probability of existence [71]. Road networks can be modeled as uncertain graphs because of unexpected traffic congestion [80]. In these networks, creating new connections between nodes (e.g., building new roads, flyovers, adding Ethernet cables) is limited by physical constraints and budget. One can introduce only k new edges where k is decided based on resource constraints. Thus, our goal is to intelligently add k new edges such that the reliability between a pair of important nodes is maximized [177].

¹This work is to appear in TKDE 2020, entitled *Reliability Maximization in Uncertain Graphs*. It is a joint work with Arijit Khan, Mohammad Al Hasan, and Rojin Rezvansangsari. I am the first author.

3.1 Preliminaries

3.1.1 Problem Formulation

An uncertain graph $\mathcal{G}$ is a triple (V, E, p) , where V is a set of n nodes, $E \subseteq V \times V$ is a set of m directed edges, and $p(e) \in [0, 1]$ is the probability that the edge $e \in E$ exists. Following bulk of the literature on uncertain graphs [10, 84, 98, 101, 141, 171], we assume that edge probabilities are independent of each other. Therefore, we employ the well-established *possible world* semantics: The uncertain graph $\mathcal{G}$ yields 2^m deterministic graphs $G \subseteq \mathcal{G}$. Each possible world $G = (V, E_G)$ is a certain instance of the uncertain graph $\mathcal{G}$, where $E_G \subseteq E$ and is obtained by independent sampling of the edges. Its probability of being observed is given as:

$$Pr(G) = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)) \quad (3.1)$$

Given a source node $s \in V$, and a target node $t \in V$, the *reliability* $R(s, t, \mathcal{G})$, also known as the s - t reliability, is defined as the probability that t is reachable from s in $\mathcal{G}$. Formally, for a possible graph $G \subseteq \mathcal{G}$, let $I_G(s, t)$ be an indicator function taking the value 1 if there exists a path from s to t in G , and 0 otherwise. $R(s, t, \mathcal{G})$ is computed as follows.

$$R(s, t, \mathcal{G}) = R(s, t, (V, E, p)) = \sum_{G \subseteq \mathcal{G}} [I_G(s, t) \times Pr(G)] \quad (3.2)$$

The problem that we study in this work is stated below.

Problem 3.1. [Single-source-target budgeted reliability maximization] *Given an uncertain graph $\mathcal{G} = (V, E, p)$, a source node $s \in V$, a target node $t \in V$, a probability threshold $\zeta \in (0, 1]$, and a small positive integer k , find the top- k edges to add in $\mathcal{G}$, each with probability $p(e) = \zeta$, such that the reliability from s to t is maximized.*

$$\begin{aligned} E^* &= \arg \max_{E_1 \subseteq V \times V \setminus E} R(s, t, (V, E \cup E_1, p)) \\ s. \ t. \quad &|E_1| = k; \quad \text{and} \quad p(e) = \zeta \quad \forall e \in E_1 \end{aligned} \quad (3.3)$$

For simplicity, we adopt a fixed probability threshold ζ on new edges. The intuition is that when establishing a new edge, generally we consider a connection with the

best/average possible reliability, e.g., an Ethernet cable with the highest reliability in case of LAN, or the average link reliability of sensor edges as we have used in the discussed case study. However, note that, if the user provides probability values for the missing edges as part of input, our proposed algorithm (§3.5) will work smoothly as it can simply use those values instead of ζ when finding the most reliable paths (see our experimental evaluation in Table 3.9, §3.7.2).

Remarks. Due to various physical and resource constraints, in practice, it might not be possible to consider all missing edges in the input graph as candidate edges to add. In a social network it is often realistic to recommend new connections between users who are within 2-3 hops. In a communication network, a new edge can be added only if the two nodes are within a certain geographical distance. While we analyze the complexity of our problem and develop algorithms for the generalized case, that is, all missing edges can potentially be candidate edges, in our solution as well as in experiments we provision for a threshold distance h : Two nodes can be added by a new edge only if they are within h -hops away. Note that (1) when h is the diameter of the graph (i.e., maximum shortest-path distance between any pair of nodes), this is essentially equivalent to the generalized case. (2) Smaller values of h reduces search space, thereby improving efficiency. Notice that all our following theoretical analyses are provided for the original problem statement. Our proposed solutions (Chapters 3.3 to 3.5) can work well with or without the physical constraint h discussed here. In our experiments, we analyze scalability of our methods for different values of h .

3.1.2 Hardness of the Problem

Problem 3.1 depends on reliability computation in uncertain graphs, which is #P-complete [10, 171]. Thus, single-source-target budgeted reliability maximization problem is hard as well. However, as reliability can be *estimated* in polynomial time via Monte Carlo (MC) sampling [63], or more sophisticated recursive stratified sampling [114], the key question is whether Problem 3.1 remains hard even if polynomial-time reliability estimation methods are employed. Due to combinatorial nature of our problem, and assuming $\mathcal{O}(n^2)$ missing edges in a sparse graph, one can design an *exact* solution that compares the s - t reliability gain for $\binom{n^2}{k}$ possible ways of adding k new edges, and then

Figure 3.1: Reduction from MAX k -COVER to single s - t budgeted reliability maximization problem.

reports the best one. However, this is clearly infeasible for large networks. We, in fact, prove that our problem is **NP**-hard, and it does not admit any **PTAS**. Moreover, Problem 3.1 is neither submodular, nor supermodular with respect to the inclusion of edges.

Theorem 3.1. *Problem 3.1 is NP-hard in the number of newly added edges, k .*

Proof. We prove **NP**-hardness by a reduction from the MAX k -COVER problem, which is **NP**-hard. In MAX k -COVER, there is a collection of subsets $S = \{S_1, S_2, \dots, S_h\}$ of a ground set $U = \{u_1, u_2, \dots, u_r\}$, where $S_i \subseteq U$ for all $i \in [1..h]$. The objective is to find a subset $S^* \subset S$ of size k such that maximum number of elements in U can be covered by S^* , i.e., so as to maximize $|\bigcup_{S_i \in S^*} S_i|$. For an instance of MAX k -COVER, we construct an instance of our Problem 3.1 in polynomial time as follows (Figure 3.1).

We create an uncertain graph $\mathcal{G}$ with a source node s and a target node t . For each element u_i in U , we add a node in $\mathcal{G}$. Each u_i is connected to t by an edge with probability p , such that $0 < p < 1$. Then, we also add each S_i in S as a node of $\mathcal{G}$. Node S_i is connected to node u_j with probability 1 if and only if $u_j \in S_i$. All other edges in $\mathcal{G}$ except those from s to all S_i have probability 0.

Thus, the candidate set of edges to add in $\mathcal{G}$ for maximizing reliability from s to t are those edges from s to all S_i . Without them, there is no path from s to t with non-zero probability. Let $\zeta = 1$, after k of these edges are selected, q out of r elements in U are now reachable from s , then the s - t reliability $= 1 - (1 - p)^q$, which monotonically increases

with larger q . This implies that Problem 3.1 and MAX k -COVER are equivalent here. If there exists a polynomial time solution for Problem 3.1, the MAX k -COVER can be solved in polynomial time too. The theorem follows. $\square$

Theorem 3.2. *Problem 3.1 does not admit any PTAS, unless $\text{Poly} = \text{NP}$.*

Proof. A problem is said to admit a *Polynomial Time Approximation Scheme* (**PTAS**) if the problem admits a polynomial-time constant-factor approximation algorithm for every constant $\beta \in (0, 1)$. We prove the theorem by showing that one can find at least one value of β such that, if a β -approximation algorithm for Problem 3.1 exists, then we can solve the well-known SET COVER problem in polynomial time. Since SET COVER is **NP**-hard, this can happen only if $\text{Poly} = \text{NP}$.

In SET COVER, there is a collection of subsets $S = \{S_1, S_2, \dots, S_h\}$ of a ground set $U = \{u_1, u_2, \dots, u_r\}$, where $S_i \subseteq U$ for all $i \in [1..h]$. The decision version of SET COVER asks if there is a subset $S^* \subset S$ of size k such that *all* elements in U can be covered by S^* .

Given an instance of SET COVER, we construct an instance of our problem in polynomial time by following the same method as in **NP**-hardness proof (Figure 3.1). In the SET COVER instance, if there is a solution with k subsets, then the optimal solution **OPT** of our problem will add k edges such that the s - t reliability after edge addition is: $1 - (1 - p)^r$. This is because $|U| = r$. In contrast, if no solution with k subsets exists for SET COVER, then **OPT** will produce s - t reliability at most: $1 - (1 - p)^{r-1}$ (because at least one of u_j would not be covered).

Let there be a polynomial-time β -approximation algorithm, **Approx** for Problem 3.1, such that $0 < \beta < 1$. According to the definition of approximation ratio, **Approx** will produce s - t reliability at least β times to that produced by **OPT**. Now, let us consider the inequality: $1 - (1 - p)^{r-1} < \beta[1 - (1 - p)^r]$. If this inequality has a solution for some values of β and p , then by simply running **Approx** on our instance of Problem 3.1, and checking the s - t reliability of the solution returned by **Approx**, one can answer SET COVER in polynomial time: a solution to SET COVER exists iff the solution given by **Approx** has s - t reliability $\geq \beta[1 - (1 - p)^r]$. Thus, to prove the theorem, we require to show that a solution to that inequality exists.

Figure 3.2: Example for non-submodularity and non-supermodularity

Our inequality has a solution iff $\beta > \frac{1-(1-p)^{r-1}}{1-(1-p)^r}$. One can verify that $\frac{1-(1-p)^{r-1}}{1-(1-p)^r} < 1$, for all $r \geq 1$ and $p > 0$. This implies that there will always be a value of $\beta \in (0, 1)$ and p for which $\beta > \frac{1-(1-p)^{r-1}}{1-(1-p)^r}$ is satisfied, regardless of r . Hence, there exists at least one value of β such that the inequality $[1 - (1 - p)^{r-1}] < \beta[1 - (1 - p)^r]$ has a solution, and, based on the above argument, such that no β -approximation algorithm for Problem 3.1 can exist. The theorem follows. $\square$

We further show that neither submodularity nor supermodularity holds for the objective function of Problem 3.1, and demonstrate with the following counter example. Therefore, standard greedy hill-climbing algorithms do not directly come with approximation guarantees for Problem 3.1.

Lemma 3.1. *The objective function of Problem 3.1 is neither submodular, nor supermodular w.r.t inclusion of edges.*

For any set $X \subseteq Y$ and all elements $x \notin Y$, a set function f is submodular if $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$. For supermodularity, the inequality is reversed.

Let us consider the example in Figure 3.2: s is the source node and t is the target nodes. Assume the node set $V = \{s, A, t\}$. Let $X = \{st\}$, $Y = \{st, sA\}$ be two edge sets. We have $R(s, t, (V, X, p)) = R(s, t, (V, Y, p)) = 0.5$. We find that $R(s, t, (V, X \cup \{At\}, p)) = 0.5$, $R(s, t, (V, Y \cup \{At\}, p)) = 1 - (1 - 0.5)[1 - 0.5^2] = 0.625$. Clearly, submodularity does not hold in this example.

Next, considering $X' = \{sA\}$ and $Y' = \{sA, st\}$, we have $R(s, t, (V, X', p)) = 0$ and $R(s, t, (V, Y', p)) = 0.5$. Then, $R(s, t, (V, X' \cup \{At\}, p)) = 0.25$, $R(s, t, (V, Y' \cup \{At\}, p)) = 0.625$. Therefore, supermodularity also does not hold.

Table 3.1: Reliability gains of three possible solutions for the example in Figure 3.3 under different probability setting.

α	ζ	Reliability		
		$\{sA, sB\}$	$\{sA, Bt\}$	$\{sB, Bt\}$
0.5	0.7	0.403	0.473	0.543
0.5	0.3	0.203	0.173	0.143
0.9	0.7	0.800	0.674	0.660

Figure 3.3: Example for the problem characterization

3.1.3 Characterization of the Problem

We next show in Observations 1~3 that the optimal solution to our problem varies based on most input parameters, even if the other set of input parameters remains the same, thereby making it non-trivial to utilize pre-existing solutions of past queries, as well as indexing-based or incremental methods.

Observation 1. *The optimal solution for Problem 3.1 may vary with different input probability threshold ζ .*

Observation 2. *The optimal solution for Problem 3.1 may vary when the edge probabilities in the original graph change.*

Observation 3. *When $k_1 < k_2$, the optimal solution for Problem 3.1 with k_1 may not be a subset of that with k_2 .*

All these three observation can be demonstrated with the example given in Figure 3.3, as follows.

Example 3.1. *In Figure 3.3, there are edges AB and At , both with probability α ($0 < \alpha < 1$), in this graph. And the edge directly connecting s and t can not exist, e.g., no direct flight can be established between two airports if they are too far away. Clearly, original reliability between s and t is 0. $\{sA, sB, Bt\}$ is the candidate set of edges to add for improving the s - t reliability.*

If budget $k = 1$, $\{sA\}$ is always the optimal solution. Its reliability is $\alpha\zeta$, which is larger than both $\alpha^2\zeta$ for solution $\{sB\}$ and 0 for solution $\{Bt\}$.

If budget $k = 2$, the s - t reliability of 3 possible solutions, $\{sA, sB\}$, $\{sA, Bt\}$, and $\{sB, Bt\}$. The reliability between s and t after adding them can be calculated as follows:

$$\begin{aligned} R(s, t, (V, E \cup \{sA, sB\}, p)) &= [1 - (1 - \zeta)(1 - \alpha \cdot \zeta)] \cdot \alpha \\ R(s, t, (V, E \cup \{sA, Bt\}, p)) &= \zeta \cdot [1 - (1 - \alpha)(1 - \alpha \cdot \zeta)] \\ R(s, t, (V, E \cup \{sB, Bt\}, p)) &= \zeta \cdot [1 - (1 - \zeta)(1 - \alpha^2)] \end{aligned}$$

Table 3.1 presents the reliability of these solutions with different α and ζ . Clearly, rows 1 and 2 have same α and different ζ , and their optimal solutions are different: $\{sA, sB\}$ and $\{sB, Bt\}$, respectively. This confirms our Observation 1. Similarly, we have same ζ but different α in rows 1 and 3, and obtain different optimal solutions. Therefore, we draw Observation 2. Moreover, $\{sA\}$, the optimal solution when $k = 1$, is not a subset of the optimal solution $\{sB, Bt\}$ when $k = 2$ if $\alpha = 0.5$, $\zeta = 0.7$, which implies our Observation 3.

Finally, we conclude this section with an interesting observation below: The direct edge st , if missing in the input graph, will always be in the top- k optimal solution. In other words, when the direct st edge is missing and if it can be added, for the top-1 solution, adding the direct st edge is the best solution.

Observation 4. *If the direct edge from s to t , st , is missing in the input graph, and is allowed to be added, st will always be included in the top- k optimal solution.*

Proof. Let G be a possible world (i.e., deterministic graph) of the original uncertain graph $\mathcal{G}$. Following Equation 3.2, the s - t reliability is calculated as: $\sum_{G \in \mathcal{G}} [I_G(s, t) \times Pr(G)]$. After adding k missing edges, G will partition into 2^k new possible worlds: $\{G_1, G_2, \dots, G_{2^k}\}$. $Pr(G) = \sum_{i=0}^{2^k} Pr(G_i)$. Clearly, when t is reachable from s in G , it will still be reachable from s in each of $\{G_1, G_2, \dots, G_{2^k}\}$, thus $I_{G_i}(s, t)$ will continue to be 1. Therefore, we only investigate those G containing no path from s to t , and thus reachability in a deterministic graph can be improved by adding new edges.

Suppose $\{e_1, e_2, \dots, e_k\}$ is an optimal solution without having st . For any G_i in $\{G_1, G_2, \dots, G_{2^k}\}$ obtained from some G (such that t was originally not reachable from s in G), we consider another solution by replacing e_j ($1 \leq j \leq k$) with st . e_j can have two possible status in G_i : **(1)** e_j exists in G_i . Here, I_{G_i} may or may not be 1. However, when

replacing e_j with st , I_{G_i} will always return 1, and induces reachability improvement; **(2)** e_j is absent in G_i . Then, the value of I_{G_i} depends only on other edges in the solution set. Replacing e_j with st will not impact the reachability. Therefore, replacing e_j with st will result in a new solution which has reliability gain at least as large as the earlier one. This implies that st , if allowed, can always be added in the optimal solution. $\square$

3.2 Related Work

Network design problems. Network design, optimization, and modification are widely studied research topics, where one modifies the network structure or attributes, targeting at some objective metrics or functions.

There exist many different metrics to characterize the “goodness” of the network, including average shortest paths [7, 18, 132], ratio of connected nodes [160], relative size of the largest connected component and average size of other components [7], network flow and delay [129, 130], the diameter of a network [54], centrality [87, 138] or eccentricity [54], average path length [136], and spectral measures [31, 36, 52]. The last group are derived from the adjacency and the Laplacian matrices of a graph. For example, [36] optimized the leading eigenvalue of a network by edge addition/deletion, due to the finding that the leading eigenvalue of the underlying graph is the key metric in determining the so-called “epidemic threshold” for a variety of dissemination models [178]. However, such global metric is not query-specific. In real-world, users may tend to optimize the network in a way that is relevant *only* to themselves, e.g., a campaigner would like to improve the influence of her product to her target customers, but not that of all similar products (from other competitors), and neither to other users who are not her targets. Moreover, many network metrics studied in the past cannot be easily generalized to probabilistic scenarios (e.g., connected component size). Our objective, *reliability*, is a fundamental metric to capture the probability that a given target node is reachable from a specific source node in an uncertain graph. Furthermore, we show that it is possible to generalize our objective to multiple-source-target cases in order to characterize a larger region in the network.

The major network manipulation operations include node addition/deletion [7, 18], edge addition/deletion [36, 54, 132], edge rewiring [18, 31], and updating edge weights [130].

Our goal is to improve the reliability between s - t pairs in a network: In our application scenarios, adding new edges is usually more practical. For example, it is often not realistic to set up a new airport only to improve the reliability of connections between two existing airports, rather establishing some new flights is much easier. In this paper, we study the problem of maximizing the reliability between a given pair of nodes by adding a small number of new edges. Altering the existing edge probabilities is not investigated here, and can be an interesting future research direction on this problem.

Reliability in uncertain networks. Due to the $\#P$ -hardness of s - t reliability estimation problem, various efficient sampling approaches have been proposed in the literature. Monte Carlo (MC) sampling [63] is a fundamental approach, which samples Z possible worlds from the input uncertain graph, and approximates the s - t reliability with the ratio of possible world in which t is reachable from s . One may combine MC sampling with BFS from the source node to further improve its efficiency [84, 97]. [116] proposed Lazy Propagation, which utilizes geometric distribution to avoid frequent probing of edges. BFSSharing improves the efficiency with offline indexes. Recursive sampling [84] and recursive stratified sampling [114] reduces the estimator variance by recursively partitioning the search space. Less samples are required for them to achieve the same variance as previous methods, thereby improving the efficiency. More recently, ProbTree index [123] was designed to support faster s - t reliability queries over uncertain graphs. Our problem and the proposed solution are orthogonal to the specific sampling method used, we have demonstrated in § 3.5 and in our paper [92] that its efficiency can be improved by employing recursive stratified sampling.

Some orthogonal directions to our problem include finding one “good” possible world [137, 158], adaptive edge testing [64–66] and crowdsourcing [119] for reducing uncertainty. In this work, we focus on improving the reliability of a source-target pair by adding a limited number of new edges.

3.3 Baseline Methods

In this section, we first present several baseline methods, that are straightforward, and demonstrate how they suffer from both effectiveness and efficiency issues. The discussions will be instrumental in developing a more accurate and scalable solution in § 3.4 and 3.5.

3.3.1 Individual Top- k Method

In the most straightforward approach, we consider every candidate edge one by one, check the reliability gain due to its addition in the input graph with edge probability ζ , and select the top- k edges with highest individual reliability gains.

Shortcomings. (1) To achieve reasonable accuracy, MC sampling requires around thousands of samples [84, 94]. Performing this for $\mathcal{O}(n^2)$ times is not scalable for large graphs. (2) Once an edge is added into the input graph, the reliability gain of adding other candidate edges may change. Hence, selecting the top- k edges based on individual reliability gains results in low-quality solution.

3.3.2 Hill Climbing Method

A better-quality solution would be the hill climbing algorithm: It greedily adds the edge that provides the maximum marginal gain to the s - t reliability at the current round, until total k new edges have been selected. In particular, consider that a set $E_1 \subseteq V \times V \setminus E$ of new edges have been already included, in the next iteration the hill climbing baseline selects a new edge $e \in V \times V \setminus (E \cup E_1)$, with $p(e) = \zeta$, such that:

$$e^* = \arg \max_{e \in V \times V \setminus (E \cup E_1)} [R(s, t, (V, E \cup E_1 \cup \{e\}, p)) - R(s, t, (V, E \cup E_1, p))] \quad (3.4)$$

Since Problem 3.1 is neither submodular nor supermodular, this approach does not provide approximation guarantees.

Shortcomings. Hill climbing also suffers from efficiency and accuracy issues. (1) This is more inefficient compared to individual top- k baseline. (2) In terms of accuracy, hill climbing still suffers from the *cold start* problem: At initial rounds, there would be several new edges with marginal reliability gain zero (or, quite small), resulting in random selections, which in turn produces sub-optimal solutions at later stages.

3.3.3 Centrality-based Method

Another intuitive approach is to find highly central nodes in the input graph, and connect them by new edges if they are not already connected, until the budget k on new edges is exhausted. In particular, we consider (1) degree centrality, that is, nodes having

higher aggregated edge probabilities considering all incoming and outgoing edges; (2) betweenness centrality, that is, nodes having larger number of shortest paths passing through. Such nodes are also known as the *hub nodes*: Connecting these hub nodes help in reducing network distances (as well as improving reliability over uncertain graphs).

Shortcomings. Although the method is efficient, and in general improves the s - t reliability, it is not customized for a specific s - t pair. This often results in low-quality solution.

3.3.4 Eigenvalue-based Method

Wang et al. [178] studied the importance of the largest eigenvalue of graph topology in the dissemination process over real networks. To model the virus propagation in a network, they assumed a fixed infection rate β for an infected node to pass the virus to its neighbor, and another fixed curing rate δ for an infected node. Then, they proved that if $\frac{\beta}{\delta} < \frac{1}{\lambda}$, where λ is the largest eigenvalue of the adjacency matrix of this network, the virus will die out in this network. Therefore, one can optimize the leading eigenvalue to control the virus dissemination in a network, e.g., with smaller λ , smaller curing rate δ is required for the same infecting rate β . Recently, Chen et al. studied the problem of maximizing the largest eigenvalue of a network by edge-addition [36], which is discussed in [92].

Shortcomings. (1) This method is not customized for a specific s - t pair, and may report low-quality solutions. (2) To the best of our knowledge, there is no equivalent transformation from virus propagation threshold $\frac{\beta}{\delta}$ to the s - t reliability. Therefore, maximizing the leading eigenvalue (to improve virus propagation) may not be equivalent to maximizing the s - t reliability.

The complexity analyses of all baseline methods can be found in our paper [92].

3.4 A Simplified Problem: Improve The Most Reliable Path

Due to the limitations of baseline approaches as discussed in § 3.3, we now explore an orthogonal direction following the notion of the *most reliable path*. The idea that we shall develop in this section will be the basis of our ultimate solution (to be introduced in

Table 3.2: Reliability gain and running time comparison without search space elimination. $k = 10$, $\zeta = 0.5$, *lastFM*

Method	Reliability Gain	Running Time (sec)
Individual Top- k	0.27	39184
Hill Climbing	0.32	406512
Centrality-based (degree)	0.03	19
Centrality-based (betweenness)	0.11	2998
Eigenvalue-based	0.09	213
Most Reliable Path	0.26	467
Individual Path Inclusion (proposed method)	0.29	332
Batch-edge Selection (proposed method)	0.31	421

§ 3.5) for the budgeted reliability maximization problem. A path between a source s and a target node t in an uncertain graph $\mathcal{G}$ is called the most reliable path $MRP(s, t, \mathcal{G})$ if the probability of that path (i.e., product of edge probabilities on that path) is maximum in comparison with all other paths between these two nodes.

$$MRP(s, t, \mathcal{G}) = \arg \max_{P \in \mathcal{P}(s, t, \mathcal{G})} \prod_{e \in P} p(e) \quad (3.5)$$

$\mathcal{P}(s, t, \mathcal{G})$ denotes the set of all paths from s to t in $\mathcal{G}$. The problem that we investigate here is a simplified version of our original problem (i.e., Problem 3.1) as stated next.

Problem 3.2. [Single-source-target most reliable path improvement] *Given an uncertain graph $\mathcal{G} = (V, E, p)$, a source $s \in V$, a target $t \in V$, a probability threshold $\zeta \in (0, 1]$, and a small positive integer k , find the top- k edges to add in $\mathcal{G}$, each new edge e having probability $p(e) = \zeta$, such that the probability of the most reliable path from s to t in the updated graph is maximized.*

$$\begin{aligned} E^* = \arg \max_{E_1 \subseteq V \times V \setminus E} & \prod_{e \in MRP(s, t, (V, E \cup E_1, p))} p(e) \\ s. \quad t. \quad & |E_1| = k; \quad \text{and} \quad p(e) = \zeta \quad \forall e \in E_1 \end{aligned} \quad (3.6)$$

Notice that the probability of the most reliable path from s to t cannot be larger than the s - t reliability. Thus, Problem 3.2 might be considered as a *simplified* version of the budgeted reliability maximization problem. Nevertheless, as reported in earlier

Table 3.3: Reliability gain and running time comparison after search space elimination. $k = 10$, $\zeta = 0.5$, $l = 30$, $r = 100$, *lastFM*. Additional time for search space elimination: 16 sec.

Method	Reliability Gain	Running Time (sec)
Individual Top- k	0.27	136
Hill Climbing	0.31	1256
Centrality-based (degree)	0.13	5
Centrality-based (betweenness)	0.21	21
Eigenvalue-based	0.20	19
Most Reliable Path	0.25	20
Individual Path Inclusion (proposed method)	0.30	16
Batch-edge Selection (proposed method)	0.33	22

studies [41, 102, 104], the most reliable path often provides a good approximation to the reliability between a pair of nodes. Thus, our intuition is simple: If the solution of Problem 3.2 is more efficient and results in higher-quality top- k edges (compared to baselines for the original budgeted reliability maximization problem), then we can augment this idea (e.g., instead of the most reliable path, one may consider multiple highly-reliable paths from s to t) to develop even better-quality solution for the budgeted reliability maximization problem.

Fortunately, Problem 3.2 can be solved exactly in polynomial time. We shall provide a constructive proof, which can also be used as an algorithm for Problem 3.2.

Theorem 3.3. *Problem 3.2 can be solved exactly in polynomial time.*

Proof. First, we color all existing edges in the input graph $\mathcal{G}$ as blue. Then, we add all missing edges to the graph, each having edge probability ζ (thus, resulting in a complete graph), and color these new edges as red. Name this new graph as $\overline{\mathcal{G}}$. The goal of Problem 3.2 is to find the most reliable path from s to t containing at most k red edges (can have zero or more blue edges), if any. Notice that we can convert the uncertain graph $\overline{\mathcal{G}}$ into an weighted graph G_0 , which has same set of edges and nodes as $\overline{\mathcal{G}}$, and the weight of each edge e in G_0 is: $w(e) = -\log p(e)$. Equivalently, we aim at finding the shortest path from s to t in G_0 containing at most k red edges (can have zero or more blue edges), if any.

To find such paths, we make k identical copies of G_0 . So, we have graphs $G_0, G_1, G_2, \dots, G_k$; and we update them as follows.

1. Remove all red edges from G_k .
2. For every $0 \leq i \leq k - 1$
 - For every red edge $e_j = (v_a, v_b)$ in G_i
 - Remove e_j from G_i
 - Draw a new edge from v_a in G_i to v_b in G_{i+1}

Now, we employ the Dijkstra's algorithm to find the shortest paths from s in G_0 to every t in G_i ($0 \leq i \leq k$). Each shortest path from s in G_0 to t in G_i corresponds to a path in the original graph $\bar{\mathcal{G}}$ with at most i red edges. We refer to these paths (if they exist) as $P_0, P_1, \dots, P_k$, respectively.

Consider a function W that gets as input a path, and returns the aggregate weight of edges on that path. If for every $1 \leq i \leq k$, we have $W(P_0) \leq W(P_i)$, then this means that adding no $k' \leq k$ edges to $\mathcal{G}$ can improve the probability of the most reliable path from s to t . Otherwise, we find $P = \arg \min_{1 \leq i \leq k} W(P_i)$, and consider all red edges in P . Adding these edges to $\mathcal{G}$ will result in the maximum probability of the most reliable path from s to t .

The time required for the above method is due to running the Dijkstra's algorithm for $k + 1$ times over a graph with $(k + 1)n$ nodes and $(k + 1)n^2$ edges. Hence, the overall time complexity of our method is $\mathcal{O}(k^2n^2 + k^2n \log(kn))$, which is polynomial in input size. The theorem follows. $\square$

Comparison with baselines. As shown in Table 3.2, solving the simplified most reliable path problem (Problem 3.2) is much faster than both baselines: Individual Top- k and Hill Climbing for the original problem (Problem 3.1). As expected, the improvement in s - t reliability via most reliable path-based solution is 0.26, which is lower but comparable to that of Hill Climbing: 0.32. However, the most reliable path approach terminates in 467 seconds, while Hill Climbing consumes about 4.7 days. For other two baselines, Centrality-based and Eigenvalue-based, most reliable path technique significantly outperforms them in reliability gain.

Algorithm 2 Search Space Elimination.

Input: source node s , target node t in uncertain graph $\mathcal{G} = (V, E, p)$, a number threshold r
Output: A set of edges E^+ as the search space

- 1: Find the top- r nodes $C(s)$ with highest reliability from the source node s
 - 2: Find the top- r nodes $C(t)$ with highest reliability to the target node t
 - 3: $E^+ \leftarrow \{(u, v) | u \neq v, u \in C(s), v \in C(t), (u, v) \notin E\}$
 - 4: **return** E^+
-

3.5 Proposed Solution: Budgeted Reliability Maximization

In this section, we present the method that we ultimately design for an effective and efficient solution to the single-source-target budgeted reliability maximization (Problem 3.1). Due to the success of the most reliable path technique as detailed in § 3.4, our final solution is developed based on a similar notion by employing multiple reliable paths, and further improved in two ways: **(1)** Reduction of search space by identifying only the most relevant candidate edges for a given s - t pair, and **(2)** improving solution quality by considering *multiple* highly reliable paths from s to t . In the following, we discuss various steps of our framework, and demonstrate accuracy and efficiency improvements compared to previous baselines.

3.5.1 Search Space Elimination

3.5.1.1 Reliability-based Search Space Elimination

In a sparse input graph $\mathcal{G}$, one can have as many as $\mathcal{O}(n^2)$ candidate edges. However, given a specific s - t query, all candidate edges may not be equally relevant. In particular, let us consider two nodes u and v : Both have low reliability either from source s , or to target t ; then adding an edge between u and v will not improve the s - t reliability significantly. Therefore, we select “relevant” candidate edges as follows (Algorithm 2). **(1)** We find the top- r nodes with the highest reliability from s . Similarly, we compute the top- r nodes having the highest reliability to t . Let us refer to these sets as $C(s)$ and $C(t)$, respectively. Notice that $s \in C(s)$ and $t \in C(t)$. **(2)** For two distinct nodes u, v , such that $u \in C(s)$, $v \in C(t)$, and u, v are not connected in the input graph $\mathcal{G}$, then we consider the new edge (u, v) , with edge probability $p(u, v) = \zeta$, as a candidate edge.

Figure 3.4: Run-through example for the proposed algorithm

We denote by E^+ the set of relevant candidate edges. Thus, we reduce the number of candidate edges from $\mathcal{O}(n^2)$ to only $\mathcal{O}(r^2)$.

The time complexity of this step is $\mathcal{O}(Z(n+m) + n \log r + r^2)$. The first term is due to MC sampling to compute the reliability of all nodes from s and to t , and the second term is due to sorting all nodes based on these reliability values.

3.5.1.2 Top- l Most-Reliable Paths Selection

Given the success of the most reliable path-based approach (§ 3.4), we further improve it with multiple highly reliable paths. Recent research has shown that what really matters in computing the reliability between two nodes is the set of highly reliable paths between them [41, 98, 102].

On adding the relevant candidate edges E^+ , we refer to the updated graph as $\mathcal{G}^+ = (V, E \cup E^+, p)$. Next, we find the top- l most reliable paths from s to t with the Eppstein's algorithm [60, 98] within $\mathcal{O}(m + n \log n + l)$. If a new edge does not appear in any of these top- l paths, it can be removed from E^+ . This further reduces the search space.

Example 3.2. Let us demonstrate “search space elimination” with Figure 3.4. Suppose we set $r = 3$, $l = 3$, and $\zeta = 0.5$. First, we select the top-3 nodes with highest reliability from source s . Clearly, $\{s, A, B\}$ will be selected. Similarly, $\{B, C, t\}$ are the top-3 nodes with highest reliability to target t . Node D , E , F , and G will be eliminated, and we obtain a graph presented in Figure 3.4(b). Then, we select top-3 most reliable paths between s and t after adding all missing edges (dotted lines) with given probability $\zeta = 0.5$ in Figure 3.4(b). They will be $\{sBt, sCBt, sCt\}$ (in decreasing order). Node A does not appear in any of these paths, and will be eliminated. Finally, we have a simplified graph shown in Figure 3.4(c).

Benefits of search space elimination. As shown in Table 3.3, our search space elimination methods can save about 99% of running time for the baselines: Individual

Top- k and Hill Climbing without accuracy loss. For Centrality-based and Eigenvalue-based baselines, both efficiency and accuracy get improved, because these baselines are now applied over a smaller and more relevant (to a specific s - t pair) subgraph. After including the time cost for conducting search space elimination: 16 seconds, the overall running time for most reliable path method and our proposed algorithms can be reduced by over 70% without accuracy loss.

3.5.2 Top- k Edges Selection

Our next objective is to find the top- k edges from the reduced set E^+ of candidate edges, so to maximize the s - t reliability. We formulate the problem as follows.

Problem 3.3 (Budgeted Path Selection). *Given the set $\mathcal{P}$ of the top- l most reliable paths from s to t in the updated graph $\mathcal{G}^+$, find a path set $\mathcal{P}^* \subseteq \mathcal{P}$ such that:*

$$\begin{aligned} \mathcal{P}^* &= \arg \max_{\mathcal{P}_1 \subseteq \mathcal{P}} R(s, t, \mathcal{P}_1) \\ \text{s. t.} \quad & |\{e : e \in E^+ \cap \mathcal{P}_1\}| \leq k \end{aligned} \tag{3.7}$$

For Problem 3.3, $R(s, t, \mathcal{P}_1)$ denotes the s - t reliability on the subgraph induced by the path set $\mathcal{P}_1$. In other words, we find a path set $\mathcal{P}^* \subseteq \mathcal{P}$ that maximize the s - t reliability, while also satisfying the constraint on k , the number of newly-added edges. Unfortunately, this problem is **NP**-hard as well, which can be proved by a reduction from the MAX k -COVER. Since the proof is analogous to the one in Theorem 5.2, we omit this for brevity. Instead, we design two practical and effective solutions as given below.

3.5.2.1 Individual Path-based Edge Selection

The algorithm is shown as Algorithm 3. First, we combine all paths from $\mathcal{P}$ that do not have any candidate edges from E^+ (line 5). We refer to these paths as $\mathcal{P}_1$, and the subgraph induced by $\mathcal{P}_1$ as G^* . Then, in each successive round, we iteratively include a remaining path P^* from $\mathcal{P} \setminus \mathcal{P}_1$ into G^* which maximally increases the reliability (estimated via MC-sampling) from s to t in G^* (line 7), while still maintaining the budget k on the number of included candidate edges in G^* . It can be formulated as:

$$P^* = \arg \max_{P \in \mathcal{P} \setminus \mathcal{P}_1} R(s, t, \mathcal{P}_1 \cup \{P\}) \tag{3.8}$$

Algorithm 3 Individual Path-based Edge Selection

Input: source node s , target node t in uncertain graph $\mathcal{G} = (V, E, p)$, a budget k for new edges, a probability threshold ζ , a candidate number threshold r , a most reliable path number threshold l

Output: A set of k edges E_1 (each with probability ζ) to add in $\mathcal{G}$ for maximizing the s - t reliability

```

1: Invoke Algorithm 4 to obtain the candidate edge set  $E^+$ 
2:  $\mathcal{G}^+ = (V, E \cup E^+, p)$ , each edge in  $E^+$  is assigned a probability of  $\zeta$ 
3: Find the top- $l$  most reliable paths  $\mathcal{P}$  from  $s$  to  $t$  in  $\mathcal{G}^+$ 
4:  $E_1 \leftarrow \emptyset, \mathcal{P}_1 \leftarrow \emptyset$ 
5: Move those the paths which do not contain any edge in  $E^+$  from  $\mathcal{P}$  into  $\mathcal{P}_1$ 
6: while  $|E_1| < k$  do
7:    $P^* = \arg \max_{P \in \mathcal{P} \setminus \mathcal{P}_1} R(s, t, \mathcal{P}_1 \cup \{P\})$ 
8:    $\mathcal{P}_1 \leftarrow \mathcal{P}_1 \cup \{P^*\}$ 
9:   Extract the set of edges  $E_{P^*}$  on path  $P^*$ 
10:   $E_1 \leftarrow E_1 \cup (E_{P^*} \cap E^+)$ 
11:  for  $P$  in  $\mathcal{P}$  do
12:    Extract the set of edges  $E_P$  on path  $P$ 
13:    if  $|E_1 \cup (E_P \cap E^+)| > k$  then
14:      Remove  $P$  from  $\mathcal{P}$ 
15:    end if
16:  end for
17: end while
18: return  $E_1$ 

```

We ensure that the number of included candidate edges from E^+ in $\mathcal{P}_1$ does not exceed k during the process (line 11-16). The included candidate edges in G^* are reported as our solution.

Let us denote by n' and m' the number of nodes and edges, respectively, in the subgraph induced by the top- l most-reliable path set $\mathcal{P}$, and T the number of MC samples required in each iteration. We need at most k iterations, thus the overall time complexity is $\mathcal{O}(kZ|P|(n' + m'))$.

3.5.2.2 Path Batches-based Edge Selection

The effectiveness of individual path selection can be improved by considering the relationships between paths in $\mathcal{P}$. The intuitions are: (1) different paths can share same set of candidate edges; (2) the candidate edge set of a path can be a subset of that for

Algorithm 4 Path Batch Construction

Input: A set of most reliable paths $\mathcal{P}$, a set of candidate edges E^+
Output: A set of path batches $\mathcal{PB}$

```

1:  $\mathcal{PB} \leftarrow \emptyset$ 
2: for Path  $P \in \mathcal{P}$  do
3:   Extract the set of edges  $E_P$  on path  $P$ 
4:   Compute the label  $L = E_P \cap E^+$  for path  $P$ 
5:   if  $PB_L \notin \mathcal{PB}$  then
6:      $PB_L \leftarrow \emptyset$ 
7:      $\mathcal{PB} \leftarrow \mathcal{PB} \cup \{PB_L\}$ 
8:   end if
9:    $PB_L \leftarrow PB_L \cup \{P\}$ 
10: end for
11: return  $\mathcal{PB}$ 
    
```

another path; and (3) different paths may have different number of candidate edges to be included.

Therefore, we design a path batch-based (instead of individual path-based) edge selection algorithm. First, we go through all paths in $\mathcal{P}$. If two paths share same set of candidate edges, they will be put into the same “path batch” (Algorithm 4). Each path batch is labeled by its candidate edge set (line 4, Algorithm 4), and in our algorithm we include a path batch in every round. In general, it follows the same procedure of Algorithm 3, but invokes Algorithm 4 after line 5. In all of follow-up steps of Algorithm 5, we shall select path batch PB instead of path P . When evaluating the marginal gain of a path batch, all other path batches whose candidate edge set is a subset of it, shall also be included in G^* in the current round. The marginal gain of this batch is normalized by the size of its candidate edge set. The detailed procedure is shown in the following example.

Example 3.3. Consider Example 3.2 and Figure 3.4, we are now selecting top-2 edges from 3 candidate edges $\{sB, sC, Bt\}$. If selecting paths individually, path sBt has the highest marginal gain 0.25 and will be selected in the first round. As budget $k = 2$ is exhausted, the solution set is $\{sB, Bt\}$. However, path sCt has reliability gain 0.15, and only adds 1 new edge. Its marginal gain per new edge is higher than that of sBt . Further, by considering it in batch path selection manner, including path $sCBt$ will also

activate path sCt . The reliability gain of adding them in batch is 0.3075, and the marginal gain per new edge is 0.1538, which is the winner of this round, and we find the optimal solution $\{sC, Bt\}$ in this example. The reliability gains for the other 2 possible solution are 0.28 for $\{sB, Bt\}$, 0.18 for $\{sB, sC\}$. This demonstrates the effectiveness of path batch selection procedure.

Benefits of path batches-based edge selection. As shown in Table 3.3, path batch selection and Hill Climbing have similar reliability gain, while path batch selection consumes significantly less running time. Comparing with individual path inclusion, path batch selection has some improvement in reliability gain, with comparable running time.

3.6 Multiple-Source-Target Reliability Maximization

In practice, queries may consist of multiple source and/or target nodes, rather than a single s - t pair. For example, in targeted marketing [91, 102, 111, 117] via social networks, the campaigner wants to maximize the information diffusion from a group of early adopters to a set of target customers. For such real-world applications, we extend our problem to adapt to multiple source/target nodes. In particular, we focus on maximizing an aggregate function (e.g., average, maximum, minimum) over reliability of *all* source-target pairs.

Problem 3.4. [Multiple-source-target budgeted reliability maximization] *Given an uncertain graph $\mathcal{G} = (V, E, p)$, a set of source nodes $S \subset V$, a set of target nodes $T \subset V$, a probability threshold $\zeta \in (0, 1]$, and a small positive integer k , find the top- k edges to add in $\mathcal{G}$, with each new edge having probability $p(e) = \zeta$, such that an aggregate function F over reliability of all s - t pairs ($s \in S, t \in T$) is maximized.*

$$\begin{aligned}
 E^* = & \arg \max_{E_1 \subseteq V \times V \setminus E} \mathbb{F}_{s, t \in S \times T} (R(s, t, (V, E \cup E_1, p))) \\
 s. t. & \quad |E_1| = k; \quad \text{and} \quad p(e) = \zeta \quad \forall e \in E_1
 \end{aligned} \tag{3.9}$$

Due to **NP**-hardness of Problem 3.1, its generalization, Problem 3.4 is also **NP**-hard. In the following sections, we consider three widely-used aggregate functions: average, minimum, maximum; and design efficient solutions.

3.6.1 Maximizing the Average Reliability

Our objective is:

$$\arg \max_{E_1 \subseteq V \times V \setminus E} \frac{1}{|S||T|} \sum_{s,t \in S \times T} R(s, t, (V, E \cup E_1, p)) \quad (3.10)$$

Note that this is equivalent to maximizing the sum of reliability of all s - t pairs. From the perspective of targeted marketing in social networks, a campaigner would like to maximize the spread of information to the entire target group; and therefore, she would prefer to maximize the average reliability.

Similar to the single-source-target budgeted reliability maximization problem, we first compute the reliable sets from source and target nodes, that is, $C(s)$ for all $s \in S$, and $C(t)$ for all $t \in T$. Next, for each pair of distinct nodes u, v , such that $u \in C(s), \forall s \in S, v \in C(t), \forall t \in T$, and u, v are not connected in the input graph $\mathcal{G}$, we consider a new edge (u, v) , having edge probability $p(u, v) = \zeta$, as a relevant candidate edge. We denote by E^+ the set of relevant candidate edges, and after adding them to $\mathcal{G}$, we refer to the updated graph as $\mathcal{G}^+ = (V, E \cup E^+, p)$.

Now, for each s - t pair, we identify the top- l most reliable paths in $\mathcal{G}^+$. Then we have total $|S||T|l$ paths in this set, and the path set might contain more than k new edges. Therefore, we employ the path batches-based edge selection method (§ 3.5.2.2): The algorithm iteratively includes path batches that maximize the marginal gain considering our current objective function (Equation 3.10), while maintaining the budget k on the number of newly inserted edges.

Time complexity. For simplicity, let $\mathcal{O}(P_1)$ denote the time complexity of reliability-based search space elimination, $\mathcal{O}(P_2)$ denote that of top- l most-reliable paths selection, and $\mathcal{O}(P_3)$ denote that of path batches-based edge selection, for the single-source-target problem. The time complexity of the proposed algorithm for average multiple-source-target budgeted reliability maximization problem is $\mathcal{O}((|S| + |T|)P_1 + |S||T|(P_2 + P_3))$. We need to evaluate all nodes' reliability from/to each source/target, which results in the first term. The second term is due to applying top- l path selection algorithm $|S||T|$ times for each s - t pair, and the path set size will be $|S||T|$ times of that for single-source-target problem.

3.6.2 Maximizing the Minimum Reliability

Our objective is:

$$\arg \max_{E_1 \subseteq V \times V \setminus E} \min_{s, t \in S \times T} R(s, t, (V, E \cup E_1, p)) \quad (3.11)$$

In other words, we aim at including k new edges such that the reliability of the s - t pair having the lowest reliability (after the addition of k edges) is maximized. In the targeted marketing setting, this can happen during complementary influence maximization [122], where multiple products are being campaigned simultaneously, and they are complementary in nature: Buying a product could boost the probability of buying another. Now, consider that each source node (e.g., an early adopter) is campaigning a different, but complementary product. In this situation, the campaigner would prefer to maximize the minimum spread of her campaign from any of the early adopters to any of her target users, because only a small percentage of the users who have heard about a campaign will buy the corresponding product.

To solve this problem, we first estimate the s - t reliability for each pair in $S \times T$ over the input graph $\mathcal{G}$. We sort these s - t pairs in ascending order in a priority queue based on their current reliability. Next, in successive rounds, we keep improving the reliability of the pair having the smallest current reliability, until the budget on k new edges can be exhausted. In particular, at any point in our algorithm, we know which source-target pair has the minimum reliability. We extract this pair from the top of the priority queue, and improve its reliability with the addition of a batch of suitable, new edges. For this purpose, we employ our algorithm for the single-source-target pair (discussed in § 3.5). The batch size can be set as $k_1 \ll k$. Note that the addition of new edges not only updates the reliability of the current pair, instead this will also increase the reliability of other s - t pairs.

Thus, after adding a batch of k_1 new edges, we re-compute the reliability of all s - t pairs and re-organize them in the priority queue. Once again, we extract the pair from the top of the priority queue, and improve its reliability with the addition of k_1 suitable, new edges. We repeat the above steps. Ultimately, we terminate the algorithm when we exhaust our budget of adding total k new edges.

Time complexity. The time complexity of our algorithm is $\mathcal{O}((|S| + |T|)P_1 + \frac{k}{k_1}AP_1 + \frac{k}{k_1}P_2 + P_3)$, $A = \min(|S|, |T|)$. Similar to maximizing the average reliability, we need

evaluating all nodes' reliability from/to each source/target. All s - t pair's original reliability can also be known through this process. However, after improving the reliability of the currently selected s - t pair by adding k_1 edges, we need to update the reliability of all s - t pairs. This will happen $\frac{k}{k_1}$ times, and the cost is $\frac{k}{k_1}AP_1$, $A = \min(|S|, |T|)$. The top- l paths selection will be operated $\frac{k}{k_1}$ times. The complexity of executing top- k_1 edge selection is $\mathcal{O}(\frac{k_1}{k}P_3)$, and it will happen $\frac{k}{k_1}$ times. Thus, the total time cost for edge selection remains $\mathcal{O}(P_3)$.

3.6.3 Maximizing the Maximum Reliability

Our objective function is:

$$\arg \max_{E_1 \subseteq V \times V \setminus E} \max_{s, t \in S \times T} R(s, t, (V, E \cup E_1, p)) \quad (3.12)$$

In the targeted marketing scenario, let us again consider complementary influence maximization [122], where each source user (e.g., an early adopter) is campaigning a different, but complementary product. However, each target user is now a celebrity in Twitter. Hence, the campaigner wants at least one target user to be influenced by one of her products. In other words, the campaigner would be willing to maximize the spread of information from at least one early adopter to at least one target customer.

Note that if $S \cap T \neq \emptyset$, the problem is trivial, as the maximum reliability is already one. Therefore, below we consider the case when $S \cap T = \emptyset$. A straightforward solution to our problem would be to separately consider each s - t pair from $S \times T$, improve its reliability by adding k new, suitable edges. Then, we pick the pair which achieves the maximum final reliability, and report those k new edges that were selected for this s - t pair. However, the time complexity of this approach is $\mathcal{O}(|S||T|)$ times to that of a single $s - t$ pair.

Next, we develop a more efficient algorithm without significantly affecting the quality. Our approach is similar to that of maximizing the minimum reliability (discussed in § 3.6.2). In each round, we maximize the reliability (by adding $k_1 \ll k$ new edges) of the pair having the current maximum reliability. After this, we re-compute the reliability of all pairs, and again pick the one which has the current maximum reliability. We terminate the algorithm when we exhaust our budget of adding total k new edges.

Time complexity. The time complexity will be the same as that of maximizing the minimal reliability, which is $\mathcal{O}((|S| + |T|)P_1 + \frac{k}{k_1}AP_1 + \frac{k}{k_1}P_2 + P_3)$, $A = \min(|S|, |T|)$.

Table 3.4: Properties of datasets, SPL denotes shortest path length, and C. Coe. denotes clustering coefficient

Dataset	#Nodes	#Edges	Edge Prob: Mean, SD, Quartiles	Type	Avg. SPL	C. Coe.
<i>Intel Lab Data</i>	54	969	0.33 ± 0.19 , {0.16, 0.27, 0.44}	Device	2.0	0.71
<i>LastFM</i>	6 899	23 696	0.29 ± 0.25 , {0.13, 0.20, 0.33}	Social	7.1	0.13
<i>AS_Topology</i>	45 535	172 294	0.23 ± 0.20 , {0.08, 0.21, 0.31}	Device	3.2	0.36
<i>DBLP</i>	1 291 298	7 123 632	0.11 ± 0.09 , {0.05, 0.10, 0.14}	Social	6.7	0.63
<i>Twitter</i>	6 294 565	11 063 034	0.14 ± 0.10 , {0.10, 0.10, 0.19}	Social	4.5	0.57
<i>Random_1</i>	1 000 000	2 500 000	0.30 ± 0.04 , {0.15, 0.30, 0.45}	Synthetic	4.1	0.11
<i>Random_2</i>	1 000 000	5 000 000	0.30 ± 0.04 , {0.15, 0.30, 0.44}	Synthetic	4.0	0.12
<i>Regular_1</i>	1 000 000	2 500 000	0.30 ± 0.04 , {0.15, 0.30, 0.45}	Synthetic	11.1	0.56
<i>Regular_2</i>	1 000 000	5 000 000	0.30 ± 0.04 , {0.16, 0.31, 0.45}	Synthetic	10.8	0.56
<i>SmallWorld_1</i>	1 000 000	2 500 000	0.29 ± 0.04 , {0.14, 0.30, 0.44}	Synthetic	4.6	0.55
<i>SmallWorld_2</i>	1 000 000	5 000 000	0.30 ± 0.04 , {0.15, 0.30, 0.45}	Synthetic	4.5	0.59
<i>ScaleFree_1</i>	1 000 000	2 500 000	0.30 ± 0.04 , {0.15, 0.30, 0.45}	Synthetic	5.6	0.46
<i>ScaleFree_2</i>	1 000 000	5 000 000	0.29 ± 0.04 , {0.15, 0.29, 0.44}	Synthetic	4.8	0.48

3.7 Experimental Results

We perform experiments to demonstrate effectiveness (i.e., reliability improvement), efficiency, scalability, and memory usage of our algorithms. We report sensitivity analysis by varying all input parameters in this section, and provide more sensitivity tests in the Appendix. The code is implemented in C++, executed on a single core, 40GB, 2.40GHz Xeon server.

3.7.1 Experimental Setup

Real-world Datasets. We use 5 real-world graphs, consisting of 3 social and 2 device networks (Table 5.2). (1) *Intel Lab Data* (<http://db.csail.mit.edu/labdata/labdata.html>). It is a collection of sensor communication data with 54 sensors deployed in the Intel Berkeley Research Lab between February 28th and April 5th, 2004. (2) *LastFM* (www.last.fm). It is a musical social network, where users listen to musics, and share them with friends. An edge between two users exists if they communicate at least once. (3) *AS_Topology* (<http://data.caida.org/datasets/topology/ark/ipv4/>). An autonomous system (AS) is a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators on behalf of a single administrative entity, e.g., a university. The AS connections are established with BGP protocol. It may fail due to various reasons, e.g., failure when one AS updates its connection configuration to ensure stricter security setting, while some of its peers can no longer satisfy it, or some connections are cancelled manually by the AS administrator. We downloaded one network snapshot per month, from January 2008 to December

2017. (4) *DBLP* (<https://dblp.uni-trier.de/xml/>). It is a well-known collaboration network. We downloaded it on March 31, 2017. Each node is an author and edges denote their co-author relations. (5) *Twitter* (<http://snap.stanford.edu/data/>). This is a widely used social network: Nodes are users and edges are re-tweets. All the social network datasets are undirected, and the other device networks are directed

Synthetic Datasets. In order to study the effects of network properties to the performance of our algorithms, we generate 8 synthetic datasets, with the help of NetworkX package (<https://networkx.github.io>). They can be categorized into following 4 kinds of networks, each having 2 instances with different number of edges. (1) *Random*. We generate the random networks with the well-known Erdős-Rényi model, that is, the edge between every pair of nodes exists with a fixed probability p . Here, the p value for *Random_1* is 5×10^{-6} , and the one for *Random_2* is 1×10^{-5} . The random network tends to have a small average shortest path length, and a small clustering coefficient. The degrees of nodes in general follow Poisson distribution. (2) *Regular*. We adopt the k -regular networks model here. In a k -regular network, every node has same degree of k . In particular, the k value for *Regular_1* is 5, and the one for *Regular_2* is 10. In a regular network, the average shortest path length is usually high, and the clustering coefficient is also high. (3) *SmallWorld*. In a small-world network, most nodes are not neighbors of one another, but the neighbors of any given node are likely to be neighbors of each other, and most nodes can be reached from every other node by a small number of hops or steps. Thus, it always has a small shortest path length, and a high clustering coefficient. The widely-used Watts-Strogatz model for generating small-world networks starts with a k -regular lattice, and re-writes its edge connections with probability p to obtain a small-world graph. We adopt $k = 5$ and $k = 10$ for our two instances, respectively. The p is set to 0.3. (4) *ScaleFree*. A scale-free network is a network whose degree distribution follows a power law. It usually has relatively smaller shortest path length, and higher clustering coefficient. Our scale-free networks are generated with Barabási-Albert preferential attachment model, where a graph of n nodes is grown by attaching new nodes each with m edges that are preferentially attached to existing nodes with high degree. For *ScaleFree_2*, the m is set to be 5. In order to maintain a consistent number of edges as all previous networks, we make slight change in the source code to allow alternating

$m = 2$ and $m = 3$ during the generation. Real-world social networks are mostly both small-world and scale-free. All the synthetic networks are undirected.

Edge probability models. Our problems and solutions are *orthogonal to the specific way of assigning edge probabilities*. We adopt some widely-used models for generating edge probabilities in our evaluation. **(1) Intel Lab Data** and **(3) AS_Topology**. The edge probabilities in these two datasets are real probabilities. For *Intel Lab Data*, the probabilities on edges denote the percentages of messages from a sender successfully reached to a receiver. For *AS_Topology*, once an AS connection (i.e., an edge) is observed for the first time, we calculate the ratio of snapshots containing this connection within all follow-up snapshots as the probability of existence for this edge. **(2) LastFM**. The probability on any edge corresponds to the inverse of the out-degree of the node from which that edge is outgoing. **(4) DBLP** and **(5) Twitter**. We assign the edge probability following $1 - e^{-t/\mu}$, which is an exponential cdf of mean μ to a count t [84]. In *DBLP*, t denotes the count of the collaborations between two authors. In *Twitter*, t is the count of re-tweet actions. We set $\mu = 20$.

Queries. For each dataset and single-source-target queries, we select 100 different s - t pairs. In practice, if two nodes are too close to each other, their original reliability will be naturally high; thus, it might be unnecessary to improve their reliability further. Therefore, we first select a source node uniformly at random, and find all its neighbors within 3-5 hops. A target node is chosen from those neighbors uniformly at random. We also demonstrate experimental results by varying the distance between s - t pairs in [92].

For multiple-source-target queries, we first generate a single-source-target query s - t . Then, for that s , we find all its neighbors that are within 5-hops away, and randomly select q of them into the source set S (i.e., source set size= q). Similarly, we pick q of the within 5-hop neighbors of t as the target set T , uniformly at random. We ensure that the source set and the target set do not overlap. Finally, 100 different source-target sets are generated.

Parameters setup. **(1) Budget on #new edges (k).** For single s - t queries, we vary k from 5 to 50, and use 10 as default. For multiple-source-target queries, we vary k from 10 to 500, use 100 as default. **(2) Probability on new edges (ζ).** We vary ζ from 0.3 to 0.7, and use 0.5 as default. **(3) Number of candidate nodes (r).** We vary r from 20 to 300,

and use 100 as default. **(4) Number of most-reliable paths (l).** We vary l from 10 to 50, and use 30 as default. **(5) #Sources and #targets.** For multiple-source-target queries, we vary source and target set sizes from 3 to 500. **(6) The ratio of $\frac{k_1}{k}$.** For multiple-source-target case with Max and Min aggregate functions, we further have a parameter k_1 as the budget for the current selected pair. We vary k_1 from 5% to 30% of k , and use 10% as default. **(7) Distance constraint for new edges.** In practice, some missing edges cannot be candidate edges due to physical constraints. As an example, in our case study in *Intel Lab* sensor network, only short distance connections (≤ 15 meters) are allowed to be established. In social networks, if two users have no common friends, it is unlikely that they will start communicating. In current experiments, we assume that a missing edge can be added only if its two endpoints are $\leq h$ -hops in the input graph. We vary h from 2 to 5, and use 3 as default. For real-world applications, one can easily set this constraint based on her requirements.

Competing methods. For single-source-target query, we compare our ultimate method: path batches-based edge selection (BE) with individual path-based edge selection (IP), most reliable path (MRP), and our best baseline: Hill Climbing (HC). For multiple-source-target case, we employ Hill Climbing (HC), Eigenvalue-based Optimization (EO) [36], and two most recent algorithms, ESSSP [138] and IMA [49], as competitors. Moreover, on the smallest dataset, *Intel Lab Data*, we have the exact solution (ES) as a competitor, which enumerates all possible combinations of k missing edges, and find the best one with highest reliability gain. All the methods are coupled with our search space elimination strategy (§ 3.5.1.1) and an advanced sampling method: RSS ([92]).

Performance Metrics. **(1) Reliability gain.** We compute reliability gain due to k new edges for each pair of source and target nodes, and report the average reliability gain over 100 distinct s - t pairs. **(2) Running time.** We report the end-to-end running time, averaged over 100 queries. **(3) Memory usage.** We report the average memory usage of running each query.

3.7.2 Single-source-target results

Comparison with the exact solution. The exact solution (ES) enumerates all possible combinations of k missing edges, and finds the best one with the highest reliability

Table 3.5: Comparison with the exact solution (ES), $k = 3$, $\zeta = 0.33$, $r = 54$, $l = 30$, *Intel Lab Data*.

Method	Reliability Gain	Running Time (sec)
ES	0.252	19189
IP	0.222	8
BE	0.237	12

Table 3.6: Single-source single-target budgeted reliability maximization on different datasets. $k = 10$, $\zeta = 0.5$, $r = 100$, $l = 30$.

Dataset	Reliability Gain				Running Time (sec)				Memory Usage (GB)			
	HC	MRP	IP	BE	HC	MRP	IP	BE	HC	MRP	IP	BE
<i>lastFM</i>	0.31	0.25	0.30	0.33	717	24	14	25	0.06	0.04	0.04	0.04
<i>AS_Topology</i>	0.42	0.40	0.41	0.42	785	30	26	32	0.30	0.28	0.28	0.29
<i>DBLP</i>	0.24	0.19	0.22	0.24	1105	125	118	129	6.9	6.2	6.5	6.5
<i>Twitter</i>	0.13	0.11	0.15	0.19	1053	140	127	141	11.0	9.4	9.8	9.8

gain. The number of missing edges can reach $\mathcal{O}(n^2)$ in sparse graph, and results in $\binom{n^2}{k}$ possible choices. Thus, the exact solution (ES) is extremely inefficient in larger graphs. However, due to the small size of *Intel Lab Data*, we can apply such exhaustive search, and empirically compare the performance of our proposed solution, BE.

Here, we follow the setting used in our case study in §3.7.4.1, that is, only 3 new short distance (≤ 15 meters) links are allowed to be established, each have the average probability 0.33. 30 distinct pairs of sensors, which are remote and with lower original reliabilities, are selected as queries.

As shown in Table 3.5, our proposed solution, BE, exhibits very close performance against the exact solution (ES) in reliability gain. Particularly, BE returns same set of edges as ES, in 25 out of 30 queries. However, the running time of BE are at least three orders of magnitude faster than ES. This demonstrates the both the effectiveness and the efficiency of our methods.

Comparison of all competing methods on different datasets with default parameters. In Table 3.6, we present the reliability gain obtained by four methods, and the corresponding running time and memory usage, on various datasets with default parameters. Clearly, our ultimate method, path batches-based edge selection (BE) outperforms others. For reliability gain, it wins on all datasets. On *Twitter*, the advantage of BE is more prominent. The reason is that *Twitter* is a sparser graph compared to other datasets, and the highly reliable paths connecting source to target are more likely

Table 3.7: Single-source-target reliability maximization on different synthetic datasets. $k = 10$, $\zeta = 0.5$, $r = 100$, $l = 30$.

Dataset	Reliability Gain				Running Time (sec)				Memory Usage (GB)			
	HC	MRP	IP	BE	HC	MRP	IP	BE	HC	MRP	IP	BE
<i>Random_1</i>	0.17	0.13	0.15	0.17	1142	142	120	145	5.8	5.2	5.3	5.4
<i>Random_2</i>	0.16	0.13	0.16	0.16	1240	160	131	171	7.1	6.7	6.9	6.9
<i>Regular_1</i>	0.23	0.15	0.16	0.24	609	54	38	69	5.1	4.7	4.8	4.9
<i>Regular_2</i>	0.21	0.16	0.16	0.22	697	84	54	91	6.8	6.4	6.6	6.6
<i>SmallWorld_1</i>	0.20	0.15	0.16	0.19	877	83	60	80	5.4	5.0	5.1	5.2
<i>SmallWorld_2</i>	0.18	0.14	0.17	0.19	967	104	79	101	7.0	6.4	6.5	6.5
<i>ScaleFree_1</i>	0.18	0.14	0.16	0.19	926	94	81	101	5.5	5.0	5.2	5.2
<i>ScaleFree_2</i>	0.17	0.14	0.15	0.17	972	116	93	120	6.9	6.4	6.6	6.6

to contain more than one missing edges — this fact enhances the impact of path batches. Individual path-based edge selection (IP) always has lower reliability gain compared to BE. The polynomial-time solution, MRP for the restricted version of our problem has the lowest reliability gain among these methods, as expected.

Considering the running time, IP is the best one. However, BE is only about 10-20 seconds slower than IP across all the datasets. Both of them are about an order of magnitude faster than the baseline HC. The memory usages of IP and BE are similar, while MRP costs slightly less memory.

Comparison of all competing methods on different synthetic datasets with default parameters. Similar to previous part, we present the results on synthetic datasets in Table 3.7. Our ultimate method, BE, still outperforms others on all datasets. For reliability gain, it first confirms our finding on the real datasets that the reliability gain tends to be higher on sparser graphs, regardless of the kind of network. Further, it can be observed that we can achieve higher reliability gain on regular networks. It is well-known that establishing a few short cut edges can sharply reduce the average shortest path length in a network, and transforming it gradually into a small-world graph. The original path length is higher in a regular graph, which allows more for improvement.

The running time on random graphs is the highest, while that of regular graphs is lowest. The top- r candidate nodes, $C(s)$ and $C(t)$, tend to be farther from s or to t on regular graphs. Since our s - t query pairs are 3-5 hops away, $C(s)$ and $C(t)$ will have more overlap on regular graphs, which can reduce the number of candidate edges. Furthermore, clustering coefficients are high on regular graphs, thus more edges have existed from $C(s)$ to $C(t)$. This again reduces the number of candidate edges. These are the reasons for

Table 3.8: Reliability gain and running time comparison with varying budget on $\#$ new edges k . $\zeta = 0.5$, $r = 100$, $l = 30$, *DBLP*.

k	Reliability Gain				Running Time (sec)			
	HC	MRP	IP	BE	HC	MRP	IP	BE
3	0.19	0.19	0.19	0.20	373	96	95	100
5	0.21	0.19	0.20	0.21	576	103	97	106
8	0.22	0.19	0.21	0.23	923	111	107	110
10	0.24	0.19	0.22	0.24	1097	117	112	121
15	0.24	0.19	0.23	0.25	1504	127	119	126
20	0.26	0.19	0.25	0.26	1974	136	125	131
30	0.26	0.19	0.26	0.26	3091	148	131	136
50	0.27	0.19	0.28	0.28	5102	162	139	142

the smaller running time of our methods on regular graphs. The random graphs have the contrary properties, therefore it is slower to excute our algorithms there.

Varying the budget k on $\#$ new edges. We present the results on *LastFM* and *DBLP* datasets in [92] and in Tables 3.8, respectively. Similar trends can be observed that the reliability gain increases with larger k . Such growth is more significant when k is small, for example the reliability gain increases from 0.27 to 0.33 when k increases from 5 to 10, while only 0.02 increase can be obtained when permitting k from 20 to 30, on *LastFM*. The reliability gain nearly saturates at $k=20$ on *DBLP*. On both datasets, BE outperforms other methods in reliability gain, no matter how large is k . The reliability gain of MRP converges at the beginning. This is because in the restricted version of our problem, we only consider the most reliable path. A path containing larger number of new edges will have longer length, and tends to have lower probability. Such paths are unlikely to be the most reliable path.

For running time, HC is $\approx 100\times$ slower than others. MRP, IP, and BE are comparable in running time. All of them finish within 200 seconds with the largest $k = 50$. The running time of MRP increases faster than IP and BE with larger k , since it requires k copies of the original graph to find the most reliable paths with exactly 0 to k missing edges, although this does not help improve the solution quality in practice.

Varying probability ζ on new edges. The experimental results on *AS_Topology* and *Twitter* are provided in Tables 3.9 and in [92], respectively. The reliability grows almost linearly with the probability threshold ζ . Sometimes, the growth rate may be even higher (e.g., on *Twitter*). The reason is that the optimal solution set of edges may change with

Table 3.9: Reliability gain, running time comparison with varying probability ζ on new edges. $k = 10, r = 100, l = 30, AS_Topology$.

ζ	Reliability Gain				Running Time (sec)			
	HC	MRP	IP	BE	HC	MRP	IP	BE
0.3	0.26	0.23	0.25	0.27	780	28	25	30
0.4	0.34	0.31	0.32	0.33	774	27	28	29
0.5	0.42	0.40	0.41	0.42	785	30	26	32
0.6	0.51	0.48	0.50	0.51	801	32	32	37
0.7	0.59	0.57	0.58	0.60	810	37	35	40

 Table 3.10: Analysis with different probabilities on new edges. $k = 10, r = 100, l = 30, Twitter$.

New edge probabilities	Reliability Gain				Running Time (sec)				Memory (GB)			
	HC	MRP	IP	BE	HC	MRP	IP	BE	HC	MRP	IP	BE
rand(0, 1)	0.17	0.13	0.16	0.18	1049	139	134	141	11.0	9.4	9.8	9.8
rand(0.2, 0.6)	0.13	0.10	0.11	0.14	1019	132	127	134	10.9	9.4	9.8	9.8
rand(0.4, 0.8)	0.21	0.17	0.20	0.21	1052	140	134	143	11.0	9.4	9.8	9.8
$N(0.5, 0.038)$	0.19	0.14	0.16	0.20	1036	135	133	136	11.0	9.4	9.8	9.8

different ζ (Observation 1), and a sharp increase may happen when shifting from a set of edges to another (Example 3.1). The running times of all the methods are not sensitive to different ζ . However, with larger ζ , the running time slightly increases.

Table 3.7.2 provides additional analysis about the probabilities on new edges. Here, instead of a fixed threshold ζ , we allow different probabilities on different new edges. Probabilities on new edges are generated uniformly at random in different range, or generated following normal distribution $N(0.5, 0.038)$ (99% of value generated are in range $(0, 1)$). It can be viewed that the results are very similar to all our previous study with fixed threshold ζ . *This confirms that our proposed algorithm, BE works well even when different probabilities for the missing edges are provided as input.*

More sensitivity and scalability tests can be found in our full paper [92]. In general, the running time and the memory usage of our algorithms are both about linear to the graph size.

3.7.3 Multiple-source-target results

Varying #source-target nodes. We present the results on the largest dataset, *Twitter*, in Tables 3.11, 3.12, and 3.13, for the aggregate functions: *Minimum*, *Maximum*, and

Table 3.11: Reliability gain and running time comparison for multiple-source-target pairs. $k = 100$, $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, *Twitter (Min.)*.

#Source:#Target	Reliability Gain					Running Time (sec)				
	HC	EO	ESSSP	IMA	BE	HC	EO	ESSSP	IMA	BE
3:3	0.34	0.29	0.35	0.26	0.38	1662	384	726	658	358
10:10	0.27	0.19	0.33	0.20	0.36	2140	979	1289	1058	1007
50:50	0.19	0.12	0.24	0.19	0.28	4051	2910	2946	2310	3049
100:100	0.15	0.06	0.18	0.11	0.19	7144	5766	6690	6302	5708
200:200	0.12	0.02	0.13	0.05	0.16	13615	8711	9899	8898	8981
500:500	0.12	0.04	0.12	0.05	0.15	30111	17198	22166	19454	18082

Table 3.12: Reliability gain and running time comparison for multiple-source-target pairs. $k = 100$, $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, *Twitter (Max.)*.

#Source:#Target	Reliability Gain					Running Time (sec)				
	HC	EO	ESSSP	IMA	BE	HC	EO	ESSSP	IMA	BE
3:3	0.27	0.20	0.19	0.21	0.27	1682	404	772	662	377
10:10	0.24	0.17	0.16	0.21	0.25	2381	982	1222	1091	1071
50:50	0.24	0.16	0.17	0.20	0.28	4051	2955	3042	2510	3366
100:100	0.23	0.16	0.14	0.19	0.26	7144	5822	6315	6335	6101
200:200	0.19	0.14	0.15	0.19	0.23	13615	8801	9538	9044	9114
500:500	0.20	0.12	0.11	0.18	0.22	30111	17699	20994	19049	18988

Table 3.13: Reliability gain and running time comparison for multiple-source-target pairs. $k = 100$, $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, *Twitter (Avg.)*.

#Source:#Target	Reliability Gain					Running Time (sec)				
	HC	EO	ESSSP	IMA	BE	HC	EO	ESSSP	IMA	BE
1:1	0.22	0.23	0.21	0.25	0.25	2003	246	398	334	182
3:3	0.21	0.24	0.23	0.24	0.24	4221	369	701	641	239
10:10	0.17	0.19	0.17	0.20	0.21	28948	944	1196	1115	787
50:50	0.14	0.17	0.15	0.17	0.17	131487	2856	2899	2401	2321
100:100	0.12	0.13	0.12	0.14	0.15	194449	2978	6449	6100	4662
200:200	-	0.09	0.09	0.11	0.12	-	8795	9647	8810	7812
500:500	-	0.06	0.06	0.08	0.10	-	17189	20154	18994	13908

Average, respectively. The baselines ESSSP and IMA can also run in single-source-target scenario, and we keep their results in the first row of Table 3.13. Our purposed method, BE, significantly outperforms the baselines, HC, EO, ESSSP, and IMA in reliability gain, and runs at least $40\times$ and $2\times$ faster than HC, with *Average* and *Minimum/Maximum* aggregate functions, respectively. The running times of EO, ESSSP, IMA, and BE are comparable. In general, the running time of our method, BE is almost linear to the number of source/target nodes.

Furthermore, our method results in higher reliability gain with *all* 3 aggregate functions when comparing with EO, especially for *Minimum* and *Maximum*. This is because EO is not query-specific. EO optimizes the leading eigenvalue of a graph, which is a global metric and may have little to do with the query pair having *Minimum* or *Maxi-*

Figure 3.5: Reliability gain and running time comparison with varying budget on #new edges k . $\zeta = 0.5$, $r = 100$, $l = 30$, #Sources=#Targets=100, *Twitter*.

mum reliability. The performance of IMA algorithm is closed to our method with *Average* aggregate function, since its objective, influence spread, can be considered as a variant of average aggregated reliability (see § 3.7.4.2). When we have only one source-target pair, the objective of IMA becomes exactly the same as ours, which results in the same reliability gain, as shown in Table 3.13. The performance of ESSSP is always worse than our method.

Varying the budget k on #new edges. Similar to single-source-target case, we vary k , now in a larger scale: 10 to 500, and present the result in Figure 3.5. The reliability gains for all three aggregate functions increase with larger k . The running time of BE with *Minimum/Maximum* aggregate function is less sensitive to a larger k , since the complexity of their top- k edge selection part remains the same as single-source-target case, while the search space elimination part scales up. On the contrary, the running time of BE with *Average* is almost linear to k . However, *Average* is still less time consuming than *Minimum/Maximum* with large k .

3.7.4 Case Studies

3.7.4.1 Intel Lab Sensor Network

To demonstrate the effectiveness of our problem, we conduct a case study on the *Intel Lab Data* (<http://db.csail.mit.edu/labdata/labdata.html>). This dataset contains the sensor network information with 54 sensors deployed in the Intel Berkeley Research Lab (map given in Figures 3.6 and 3.7) between February 28th and April 5th, 2004. The probabilities on links denote the percentages of messages from a sender successfully

Figure 3.6: Improving the reliability from sensor 21 (right) to 46 (left) with 3 new links (marked by dotted lines).

reached to a receiver. The average link probability is 0.33 (ignoring edge probabilities which are lower than 0.1).

Assume that our goal is to maximize the reliability from: (1) a sensor on the right hand side of the lab to a sensor on the left hand side (e.g., from sensor 21 to 46); (2) between two sensors on the diagonal of the lab (e.g., from sensor 15 to 40). The original reliability from sensor 21 to 46 is 0.40 , and that from sensor 15 to 40 is 0.28 . Due to budget constraints, only 3 new links are allowed for each case. We further assume that the probability of each new link would be the same as the average edge probability of the original dataset, that is, 0.33. We notice that if two sensors are more than 20 meters away, the original link probability between them is usually close to 0. Thus, we only allow establishing new links between a pair of sensors that are at most 15 meters away.

Figure 3.6 demonstrates the solution obtained by our algorithm for case (1). Only those links with probabilities higher than the average value (0.33) are shown in the figure, and the thickness represents link probabilities. Clearly, sensor 46 has very weak connections from outside, while sensor 21 is connected with the bottom part of the lab. A very dense network exists in the bottom part of the lab. Therefore, our solution for this case is to connect sensor 46 with sensors in the bottom part of the lab. By establishing three links 2 to 46, 35 to 46, and 37 to 46, we improve the reliability between 21 to 46 from 0.40 to 0.88 (i.e., the new reliability is more than twice the original reliability from 21 to 46).

For case (2), notice in Figure 3.7 that the sensors in the center part of lab are well-connected, and the links in this region are thicker than those in the bottom part. Sensor

Figure 3.7: Improving the reliability from sensor 15 to 40 (on the diagonal) with 3 new links (marked by dotted lines).

Figure 3.8: Influence spread comparison in *Databases* area. $\zeta = 0.5$, $r = 100$, $l = 30$, $\frac{k_1}{k} = 10\%$, *DBLP*.

15 (the source node) has a few connections with sensors in the bottom part, but no connection with sensors in the center part. Sensor 40 (the destination node) has limited connections beyond its physical neighbors. Existing configuration offers a poor reliability of 0.28 for the connection between source and destination which we like to improve. The smart decision made by our algorithm is as follows: First, connect sensor 35 to 40, thus making sensor 35 a bridge between the center and the bottom region of the network; Second, enable connection from sensor 15 to the center part (by establishing link from 15 to 10, and from 15 to 11). This results in 0.58 overall reliability from sensor 15 to 40, which is more than double of the original reliability value. These results illustrate how our proposed solution for the budgeted reliability maximization problem can be useful in solving real-life problems.

3.7.4.2 Influence Maximization

We present an application of budgeted reliability maximization in the classic influence maximization problem [94] using *DBLP*.

In social influence maximization following the widely-used independent cascade model [94], when some node u first becomes active at step t , it gets a single chance to activate each of its currently inactive out-neighbors v at step $t + 1$, with probability $p(u, v)$. Initially, only the source nodes are active, and the activation continues in discrete steps. When no more nodes can be activated, the number of active nodes in target set is referred to as the influence spread. With possible world notation, the influence spread from source set S to target set T can be formulated as:

$$Inf(S, T) = \sum_{G \subseteq \mathcal{G}} \left[Pr(G) \sum_{t \in T} I_G(S, t) \right] \quad (3.13)$$

As discussed in § 3.6.1, the *average reliability* from S to T is:

$$R_{avg}(S, T) = \frac{1}{|S||T|} \sum_{G \subseteq \mathcal{G}} \left[Pr(G) \sum_{s, t \in S \times T} I_G(s, t) \right] \quad (3.14)$$

Clearly, in each possible world, if we only check whether there is *at least one* path to t from any $s \in S$, instead of counting the *exact number* of $s \in S$ which has a path to t , our problem becomes equivalent to the (targeted) influence maximization problem. Adding a new edge in this network implies recommending and/or establishing collaboration with an author in the real-world [32].

We select a set of junior researchers in *Databases* area, containing 1000 authors randomly selected from all the authors with 1-3 papers in top-tier venues [SIGMOD, VLDB, ICDE]. In a similar way, we choose 50 senior researchers with more than 10 papers there, uniformly at random. The expected influence spread from the senior to the junior group is around 462, using the IC model. Next, we aim at maximally improving the influence spread from the senior group to the junior group, by adding up to 100 new edges. As shown in Figure 3.8, our method outperforms Eigenvalue-based optimization (EO) [36], and results in about 326 more influenced authors within the junior set.

3.8 Summary

In this chapter, we introduced and investigated the novel and fundamental problem of maximizing the reliability between a given pair of nodes in an uncertain graph by adding a small number of edges. We proved that this problem is **NP**-hard and also hard to approximate. Several interesting observations are presented to characterize it. Our proposed solution first eliminates the search space based on original reliability, and then selects the top- k edges following an iterative most-reliable path-batches inclusion algorithm. We further studied one restricted and several extended versions of the problem, to support a wider family of queries. The experimental results validated the effectiveness, efficiency, and scalability of our method, and rich real-world case studies demonstrated the usefulness of our budgeted reliability maximization problem. In future, a total reliability budget on new edges, instead of a fixed/ individual budget on each new edge, can be considered. This will add more complexity on selecting proper candidate edges and allocating reliability budget to them.

Chapter 4

Finding Seeds and Tags Jointly for Social Advertising

In this chapter, we study the novel problem of jointly finding influential seed users and topic tags in a social network for enhancing the viral marketing¹. Most of the current works on *influence maximization* assume fixed edge probabilities, that is, an user has can influence another with certain probability. However, the information propagated in social medias is always associated with some topics, e.g., hashtags on Twitter, and users may have different preference for them (Figure 1.1). Setting a small budget on the number of tags is critical, for example, (a) to avoid information overload against campaign effectiveness (e.g., if one wants to write an advertisement or a blog, she may want to focus on a small number of relevant topics), and (b) for cost-effective planning and due to other physical constraints (e.g., an album can accommodate a *limited* number of genres/ songs). Motivated by these timely demands, we revisit the classical influence maximization problem, and investigate a novel problem as follows. *Given a set of target customers, a small budget k on the number of seed users, and a small budget r on the number of relevant tags, what are the top- k seed nodes and the top- r most relevant tags that maximize the expected spread of the campaign within the target set?*

¹This work is published as [SIGMOD'18] Xiangyu Ke, Arijit Khan, and Gao Cong, *Finding Seeds and Relevant Tags Jointly: For Targeted Influence Maximization in Social Networks*.

4.1 Preliminaries

4.1.1 Problem Formulation

In the literature of influence maximization [40, 94], whenever a social network user gets influenced by a campaign (e.g., buys a product, shares a photo, or tweets an information), she is viewed as being activated. A social influence network is modeled as an uncertain graph $\mathcal{G} = (V, E, P)$, where V is a set of n nodes (users), $E \subseteq V \times V$ is a set of m directed edges (friendship links, follower/followee relations, etc.). We denote by C the set of all tags present in the social network. As stated earlier, these could be explicit hashtags, together with other representative keywords. Appearance of specific tags in an online campaign affects the corresponding influence diffusion probabilities between two users. Specifically, $P : E \times C \rightarrow (0, 1]$ is a function that assigns a conditional probability to every edge $e \in E$ given a specific tag $c \in C$, such that, $P((u, v)|c)$ denotes the probability that an active user u will influence her neighbor v , given the tag c in the campaign (see Figure 1.1). Such tag-dependent influence probabilities can be learnt from past propagation datasets, e.g., [14, 75, 116]. In this work, we employ the widely used Independent Cascade (IC) model [94] for information diffusion as following.

IC Model. In the classical IC model — that is, assuming fixed edge probabilities $P(u, v)$, and without considering different influences due to various tags — the campaign starts with an initially active set of seed nodes, and then unfolds in discrete steps. When some node u first becomes active at step t , it gets a single chance to activate each of its currently inactive out-neighbors v ; it succeeds with probability $P(u, v)$. If u succeeds, then v will become active at step $t + 1$. Whether or not u succeeds at step t , it cannot make any further attempts in the subsequent rounds. If a node v has incoming edges from multiple newly activated nodes, their attempts are sequenced in an arbitrary order. Also, each node can be activated only once and it stays active until the end. The campaigning process runs until no more activations are possible. Therefore, the IC model assumes the following influence cascading scenario: people get influenced by a campaign when they come in direct contact with their friends who *very recently* adopted that campaign.

It was shown in [94] that the IC model is equivalent to the *possible world* semantics. A possible world $G = (V, E_G)$ is one certain instance of the uncertain graph $\mathcal{G}$, where

$E_G \subseteq E$, and is obtained by independent sampling of the edges. Each possible world G is associated with a probability of existence $Pr(G)$ as follows.

$$Pr(G) = \prod_{e \in E_G} P(e) \prod_{e \in E \setminus E_G} (1 - P(e)) \quad (4.1)$$

Given a set $T \subseteq V$ of customers, the (targeted) influence spread in the possible world G is defined as the number $\sigma_G(S, T)$ of target nodes that are reachable from the seed set S in G , i.e.,

$$\sigma_G(S, T) = \sum_{t \in T} I_G(S, t) \quad (4.2)$$

In the above equation, $I_G(S, t)$ is an indicator function that takes value 1 if at least one node in S is reachable to t in G , and 0 otherwise. The expected influence spread $\sigma(S, T)$ by the seed set S within the target set T is computed as the expectation of the number of reachable target nodes from S , i.e.,

$$\sigma(S, T) = \sum_{G \in \mathcal{G}} [\sigma_G(S, T) \times Pr(G)] \quad (4.3)$$

Tag-aware IC Model. In reality, edge probabilities depend on various tags related to the campaign. The information diffusion probability via an edge e , given a set of tags $C_1 \subseteq C$ in the campaign, can be derived as $P(e|C_1) = \mathbb{F}_{c \in C_1} (P(e|c))$. Here, $\mathbb{F}$ is an aggregate function, which can be defined in several ways as follows.

- **Independent Tag Aggregation.** In this case, we assume that the existence of an edge is determined by an independent process (coin flipping), one per tag c , and the ultimate existence of an edge is decided based on the success of at least one of such processes. This assumption is consistent with the IC model [94], and naturally holds if the edge probabilities were learned assuming independence of tags [150]. Formally, the information diffusion probability via an edge e , given a set of tags $C_1 \subseteq C$ in the campaign, can be derived as $P(e|C_1) = 1 - \prod_{c \in C_1} (1 - P(e|c))$.
- **Topic-Based Tag Aggregation.** This model is recently introduced in [116], which follows the earlier topic-aware influence cascade model in [14]. In particular, the model learns a set of hidden topics $Z = \{z_1, z_2, \dots, z_{|Z|}\}$ from past contents

propagated in the social network. For an edge $e = (u, v)$, the topic-aware influence probability $p(e|z)$ denotes how likely u influences v given the topic $z \in Z$ in the campaign. Moreover, the method in [116] learns the probability $p(c|z)$, which is the probability of sampling a tag $c \in C$, given the topic z .

In this work, we shall consider the independent tag aggregation approach, which is also consistent with the IC model. Moreover, in real-world scenarios, it is generally hard to achieve a specific (optimal) topic distribution for a campaign. Instead, we provide the top- k tags directly that a campaigner can associate to maximize the spread of her influence among the target customers. Nevertheless, our hardness results also hold for the topic-based tag aggregation method.

Following the IC model, we assume that edge probabilities in the uncertain graph are independent of one another. Hence, given a set C_1 of tags in a campaign, the uncertain graph $\mathcal{G}$ yields 2^m deterministic graphs $G \subseteq \mathcal{G}|C_1$, where each G is a pair (V, E_G) , with $E_G \subseteq E$, and its probability of being observed is:

$$Pr(G|C_1) = \prod_{e \in E_G} P(e|C_1) \prod_{e \in E \setminus E_G} (1 - P(e|C_1)) \quad (4.4)$$

Analogously, the expected influence spread $\sigma(S, T, C_1)$ by the seed set S in $\mathcal{G}$, given the target set T and the tag set C_1 , is computed as:

$$\sigma(S, T, C_1) = \sum_{G \subseteq \mathcal{G}|C_1} [\sigma_G(S, T) \times Pr(G|C_1)] \quad (4.5)$$

Problem Statement. Given the target set T , a budget k on the maximum number of seed nodes, and a budget r on the maximum number of tags, jointly find the top- k seed nodes and the top- r relevant tags such that the influence spread within the target users is maximized.

$$\begin{aligned} \langle S^*, C_1^* \rangle &= \arg \max_{S, C_1} \sigma(S, T, C_1) \\ \text{s. t.} \quad &|S| = k, |C_1| = r \end{aligned}$$

4.1.2 Hardness of the Problem

Our problem relies on the classical influence maximization problem, which is **NP**-hard [94] and on the influence spread computation, which is **#P**-hard [41]. As a result, our problem is also hard.

Theorem 4.1. *Given the target set T and the set C_1 of top- r tags, finding the top- k seed nodes that maximize the influence spread within the target set is **NP**-hard.*

Theorem 4.2. *Given the target set T , seed set S , and the set C_1 of top- r tags, finding the influence spread from the seed nodes to the target set is **#P**-hard.*

To prove Theorems 4.1 and 4.2, one can construct an updated uncertain graph by only considering the tags in C_1 , and thereby assigning fixed edge probabilities.

On the other hand, the classical influence maximization problem (under the IC model) is submodular with respect to inclusion of seed nodes, whereas the influence spread can be estimated in polynomial time via Monte Carlo (MC) sampling (or, a more efficient reverse sketching method [23, 163, 164]), thus a polynomial-time greedy hill-climbing algorithm [94] can solve the classical influence maximization problem with $(1 - 1/e - \epsilon)$ approximation guarantee. Hence, the key question is whether our problem remains hard if a polynomial-time oracle for the classical influence maximization is employed. We show that the answer to this question is positive.

Theorem 4.3. *Given the target set T and the seed set S , finding the set C_1^* of top- r tags that maximizes the influence spread from the seed nodes to the target nodes is **NP**-hard, even assuming a polynomial-time oracle for estimating the influence spread.*

Proof. We prove **NP**-hardness by performing a reduction from the **NP**-hard Max r -Cover problem, defined by a collection of subsets $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$ of a ground set $U = \{u_1, u_2, \dots, u_n\}$; and the objective is to find a subset $\mathcal{S}^* \subset \mathcal{S}$ of size r such that maximum number of elements in U can be covered by $\mathcal{S}^*$. Now, we construct an instance of our top- r tags finding problem as follows (Figure 4.1). We create a graph $\mathcal{G}$ and add a set of nodes $t_1, t_2, \dots, t_n$, one for each element in U . We use them as target nodes. Moreover, we put in $\mathcal{G}$ another set of nodes $s_1, s_2, \dots, s_n$ (also, one for each element in

 Figure 4.1: Hardness of top- r tags selection

U), and use them as seed nodes. Finally, if some element $u_i \in U$ is covered by at least one of the subsets in $\mathcal{S}$, we add a directed edge (s_i, t_i) in $\mathcal{G}$. For each set $S_j \in \mathcal{S}$ that covers item u_i , we assign the tag-dependent edge probability $P((s_i, t_i)|c_j) = 1$. Now, we ask if there are r tags by which all target nodes $t_1, t_2, \dots, t_n$ can be influenced from the seed nodes $s_1, s_2, \dots, s_n$. Clearly, this corresponds to selecting r subsets from $\mathcal{S}$ that maximize the number of elements covered. Hence, the theorem follows. $\square$

Theorem 4.4. *Given the target set T and the seed set S , the problem of finding the top- r relevant tags that maximize the influence spread from the seed nodes to the target nodes, does not admit any **PTAS**, unless $\mathbf{P} = \mathbf{NP}$.*

Proof. A problem has a *Polynomial Time Approximation Scheme* (**PTAS**) if the problem admits a polynomial-time constant-factor approximation algorithm for *every* constant $\beta \in (0, 1)$. We prove Theorem 4.4 by contradiction, that is, if there exists at least one value of β such that, if a β -approximation algorithm for the top- r tags selection problem exists, then we can solve the **Set Cover** problem in polynomial time. Since **Set Cover** is **NP**-hard, clearly this can happen only if $\mathbf{P} = \mathbf{NP}$. The **Set Cover** problem is defined by a collection of subsets $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$ of a ground set $U = \{u_1, u_2, \dots, u_n\}$. The decision version of **Set Cover** asks the following question: given r , is there any solution with at most r subsets from $\mathcal{S}$ that cover all elements in U ?

Starting with an instance of **Set Cover**, we construct in polynomial time an instance of our top- r tags selection problem in the same way as in the proof of Theorem 4.3. If r

Figure 4.2: Example of non-submodularity. $P(e|c) = 0$ for all other tag and edge combinations that are not specified.

subsets suffice to cover all elements in U in the original instance of **Set Cover**, the optimal solution C_1^* to our problem would have influence spread $\sigma(S, T, C_1^*) = n$. Otherwise, if no r sets cover all elements in U , C_1^* would have influence spread at most $\sigma(S, T, C_1^*) = n - 1$. Assume that a polynomial-time β -approximation algorithm for the top- r tags finding problem exists, for some $\beta \in (0, 1)$. We call it “**Approx**”. **Approx** would yield a solution C_1 such that $\sigma(S, T, C_1) \geq \beta \sigma(S, T, C_1^*)$. Now, consider the inequality $n - 1 < \beta n$. If this inequality has a solution for some values of β , by running **Approx** on the instance and checking the influence spread of this solution, one can answer **Set Cover** in polynomial time: a solution to **Set Cover** exists iff the solution given by **Approx** has influence spread $\geq \beta n$. Thus, to prove the theorem, we need to show that a solution to that inequality exists. However, it is easy to see that there will always be a value of $\beta \in (\frac{n-1}{n} + \epsilon, 1)$ for which $n - 1 < \beta n$ is satisfied ($\epsilon > 0$ is a very small fraction). This is a contradiction and completes the proof. $\square$

Lemma 4.1. *Given the target set T and the seed set S , the problem of finding the top- r tags that maximize the influence spread from the seed nodes to the target nodes, is non-submodular with respect to inclusion of tags.*

A set function f is submodular if $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$, for all sets $A \subseteq B$ and all elements $x \notin B$. Let us consider the example in Figure 4.2. The seed set = $\{s_1, s_2\}$, and target set = $\{t_1, t_2\}$. Let $C_1 = \{c_1\}$, $C_2 = \{c_1, c_2\}$. We find that $\sigma(S, T, C_1) = 0.5 \times 0.6 = 0.3$, $\sigma(S, T, C_1 \cup \{c_3\}) = 0.3$, $\sigma(S, T, C_2) = 0.3$, and $\sigma(S, T, C_2 \cup \{c_3\}) = (0.5 \times 0.6) + (0.8 \times 0.9) = 1.02$. Therefore, submodularity does not hold in this example.

4.1.3 Overview of Our Solution

Since our problem of jointly finding the top- k seed nodes and the top- r tags is **NP**-hard, as well as hard to approximate, we design practical solutions aiming for both efficiency and accuracy.

Baseline Greedy Algorithm. This is a baseline approach (given in Section 4.5): We first select the best seed and the best tag, followed by the second-best seed and the second-best tag, and so on, in a greedy manner. We find that the accuracy of this Greedy method can be improved with an iterative algorithm as follows.

Proposed Iterative Algorithm. We develop an iterative algorithm that alternatively optimizes the top- k seeds (given in Section 4.3) and the top- r tags (described in Section 4.4) until it converges to a local optimum. The complete algorithm combining these two steps will be discussed in Section 4.5. We further consider various initialization criteria for faster convergence. For both top- k seeds and top- r tags selection, we employ *reverse sketching*-based influence maximization techniques [23, 163, 164], which is the state-of-the-art method for achieving good scalability and efficiency.

4.2 Related Work

We categorize related work as follows.

Influence Maximization. Kempe et al. [94] addressed the problem of influence maximization in a social network as a discrete optimization problem, which is to identify the set of seed nodes, having cardinality k , that maximizes the expected influence spread in the graph. Due to **NP**-Hardness of the problem, they proposed a hill climbing greedy algorithm, with an accuracy guarantee of $(1 - 1/e)$. They also used MC-simulation for estimating the influence spread from a seed set. However, later it was proved in [41] that the exact computation of influence spread is **#P**-Hard.

After that, many algorithms (see [40] for details) have been developed, both heuristic and approximated, to improve the efficiency of the original greedy method. Leskovec et al. [113] and Goyal et al. [76] exploited the sub-modularity property of the greedy algorithm, and proposed more efficient CELF and CELF++ algorithms, respectively. Chen et al. [41] avoided MC simulations, and developed the maximum influence arborescence (MIA)

model using maximum probable paths for the influence spread computation. Aiming at higher efficiency, Borgs et al. [23] introduced a reverse reachable sketching technique (RRS) without sacrificing the accuracy guarantee. Tang et al. [163, 164] proposed the TIM/TIM+ and IMM algorithms, and Li et al. [117] designed indexing methods, all based on the RRS technique, to further improve its efficiency. However, all these works assume that the influence cascade probability between a pair of users is fixed; and unlike ours, they do not consider the impacts of selecting the top- r tags in order to maximize the influence spread.

Topic Dependent Influence Maximization. The classic influence maximization problem has been recently considered in a topic-aware fashion [9, 14, 39, 116]. In both [9, 39], topic-aware influence maximization solves a different problem, i.e., finding a set of seed nodes that maximize the spread of information for a *given* topic set. In contrast, the top- r relevant tags are not known apriori in our setting. In [15], Barbieri et al. introduced the problem of identifying the top- k seed nodes, together with finding the optimal topic distribution. The key difference with our work is that finding the top- r relevant tags is different than finding the optimal topic distribution. As we reasoned earlier, in real-world scenarios, it is generally hard to achieve a specific (optimal) topic distribution for a campaign. Instead, we provide the top- k tags directly that a campaigner can associate to maximize the spread of her influence among the target customers. In addition, we have novel technical contributions via indexing for targeted influence maximization, selecting good initial conditions, and proving the local optimality of our iterative algorithm, aiming towards scalability and a high-quality solution. Very recently, Li et al. [116] explored a user’s most influential topic set in a social network. Our work is different as we aim at finding not only the top- r relevant tags, but also the top- k seed nodes for targeted influence maximization in a social network.

Conditional Reliability. Conditional reliability has been widely studied in systems and device networks [4], and is recently introduced over uncertain graphs in [98, 100]. They define reliability (i.e., the probability that a given set of source nodes can reach a given set of target nodes) when edge-existence probabilities are conditioned on external factors (i.e., catalysts), and study the problem of finding top- k catalysts that maximize the reliability between a limited number of source-target pairs. On the other hand, our

objective is different: Jointly find the top- r tags and the top- k seed nodes for targeted influence maximization in a social network. For our top- r tags selection problem, we develop Batch-paths selection algorithm, following the idea of Individual-paths selection method in [98, 100]. We, however, show that Batch-paths selection method can achieve up to 30% more influence spread compared to Individual-paths selection, while the running time remains similar.

4.3 Algorithms for Finding Seeds

In this section, we design efficient algorithms for finding the top- k seed nodes, assuming that the set C_1 of top- r tags are provided. Therefore, we derive an updated uncertain graph by only considering the tags in C_1 , and thereby assigning fixed edge probabilities. This is identical to the classical (targeted) influence maximization problem; therefore it is **NP**-hard (Theorem 4.1), but the objective function is monotonic and submodular with respect to the seed set.

Lemma 4.2. *Given the target set T and the set C_1 of top- r tags, the targeted influence spread (i.e., spread within T) is monotone and submodular with respect to the seed set [102, 157].*

Therefore, one can apply a greedy algorithm (also known as an iterative hill-climbing algorithm) that finds the seed set with an approximation guarantee $(1 - 1/e)$ to the optimal solution [94, 135]. At each iteration, the algorithm selects a non-seed node u as a seed node that maximizes the marginal gain. Formally,

$$u = \arg \max_{v \in V \setminus S} [\sigma(S \cup \{v\}, T, C_1) - \sigma(S, T, C_1)] \quad (4.6)$$

Moreover, we can employ CELF [113] and CELF++ [76] optimizations to further improve the efficiency of the top- k seeds finding.

4.3.1 Targeted Reverse Sketching

Reverse sketching-based influence maximization techniques, first proposed in [23] and later improved in [163, 164], are the state-of-the-art methods with both good scalability and an $(1 - 1/e - \epsilon)$ approximation guarantee. The algorithms in [23, 163, 164] were

designed without any notion of target customers. We first refine it to adjust to targeted influence maximization with the same approximation guarantee $(1 - 1/e - \epsilon)$, and later implement a scalable solution with smart indexing techniques (in our full paper [91]).

Let G denote a (deterministic) sub-graph of the input uncertain graph $\mathcal{G}$, generated by removing each edge $e \in E$ with probability $1 - p(e)$ independently. The *Reverse Reachable (RR) Set* for a random target node v consists of all nodes that can reach v in G . The workflow of targeted reverse sketching, which is similar to the original one in [23, 163, 164], is as follows:

1. Generate θ random RR-sets from $\mathcal{G}$.
2. Next, a greedy algorithm repeatedly identifies the node presenting in the majority of RR-sets, adds it to the seed set, and the RR-sets containing it are removed. This process continues until k seed nodes are identified.

In [164], it was proved that when θ is sufficiently large, reverse sketching returns near-optimal results with high probability.

Our major update is that the target node v in each $\text{RR-set}(G, v)$ is sampled uniformly at random *only* from the target set T (and not from all nodes). We refer to it as targeted reverse sketching (TRS).

Theorem 4.5. *If θ satisfies:*

$$\theta \geq (8 + 2\epsilon) \cdot |T| \cdot \frac{\ln n + \ln \binom{n}{k} + \ln 2}{OPT_T \cdot \epsilon^2} \quad (4.7)$$

then TRS returns a $(1 - 1/e - \epsilon)$ -approximate solution, with at least $1 - n^{-1} \binom{n}{k}^{-1}$ probability.

In the above Inequality, OPT_T is the maximum influence spread in the target set T from any node set of cardinality k . Due to the similarity of our Inequality 4.7 with that of the original Inequality from [164], we omit the proof here (see the Appendix in our original paper [91] for details, which are similar to the proofs in [164]). However, *our primary contribution for the targeted influence maximization is designing several smart indexing schemes*, whose improvement is demonstrated in Figure 4.3 and details are introduced next.

Figure 4.3: Running time of TRS (state-of-the-art) vs. LL-TRS (our method), *Twitter*, $\#tags=5$, $\#targets=3K$, $\epsilon=0.1$, $\alpha=1$, $\delta=0.01$, $h=3$.

4.3.2 Indexing for Targeted Reverse Sketching

Sampling RR-sets online is still time-consuming for larger social networks. Moreover, our method works in several rounds — every time finding the seed nodes over a different graph (due to different tag sets), and this requires generating of different RR-sets at each round. Aiming at higher efficiency, we slightly loosen the constraint on RR-set independency (that is required to provide an approximation guarantee for TRS), and instead design heuristic indexing methods that appropriately generate random samples beforehand, thereby significantly reducing the online querying time (up to 30% as shown in Figure 4.3). With a series of improvements, we develop three indexing schemes, I-TRS, L-TRS, and LL-TRS, and later show that *our ultimate approach LL-TRS supersedes the original RR-set based scheme in online query time, while having a small overhead due to index building and storage*.

We refer to our first indexing scheme as I-TRS (i.e., Indexing for Targeted Reverse Sketching). Note that in successive iterations of our framework, the optimal tag set may be different. The working graph for an iteration is a combination of each chosen tag’s individual uncertain graph. This motivates us to sample and build indexes on uncertain graphs by considering each tag separately. We define our random samples, denoted by *Possible World Index*, as following.

Possible World Index. A possible world index (I, c) , for a tag c , is the sub-graph of $\mathcal{G}$, generated by: (a) first keeping only the edges associated with tag c , (b) then removing

Figure 4.4: I-TRS indexing and querying: The original uncertain graph contains three tags $\{c_1, c_2, c_3\}$. $P(e|c) = 0$ for all other tags that are not specified. The query considers the target set $T = \{D, E\}$. The selected tag set $C_1 = \{c_1, c_2\}$. We set $\theta = 4$, $\theta_c = 3$.

any remaining edge $e \in E$ with probability $1 - p(e|c)$, and finally (c) retaining all nodes of $\mathcal{G}$ inside G , even if there is no edge connected to some node in G .

Example 4.1. Figure 4.4 demonstrates the workflow of I-TRS on an uncertain graph $\mathcal{G}$ (shown in sub-figure(a)). $\mathcal{G}$ originally contains 3 tags $\{c_1, c_2, c_3\}$. When generating a possible world index for a specific tag c_i , we consider only those edges e in $\mathcal{G}$ such that $p(e|c_i) > 0$; and then remove it with probability $1 - p(e|c_i)$. We record all the edges remained as one possible world index for the tag c_i . Suppose, in this example, we are building $\theta_c = 3$ possible world indexes for every tag c_i . Then, we repeat the aforementioned procedure 3 times for each tag. Thus, we obtain all possible world indexes

as shown in sub-figure(b). Finally, sub-figure(c) presents how we generate RR-Sets when processing online queries. Assume that one requires $\theta = 4$ RR-Sets and the selected (top-2) tag set is $\{c_1, c_2\}$. Then, we repeat the following steps 4 times: First, randomly select one possible world index from each index set of tag c_1 and c_2 , and combine them together as the working graph G . For example, the first working graph in sub-figure(c) is constructed by combination (i.e., graph union) of two possible world index, I_{11} and I_{22} . Next, we choose a random node t from target set T , and apply reverse BFS in G starting from t , recording all visited nodes as an RR-Set. This ultimately produces 4 RR-Sets, $\{\{\}, \{A, B, C\}, \{B, C\}, \{C\}\}$.

Index Size Estimation. Two working graphs will be correlated if they share some common indexes. As an example, in Figure 4.4(c), the working graphs for RR-sets 1 and 4 are correlated, because they use the same index I_{11} for tag c_1 . We, therefore, aim at reducing the average number of common indexes between any pair of working graphs from the set G of all θ working graphs. Note that the following theorem does not ensure approximation guarantee, but only provides a bound on the required number of possible world indexes per tag. It ensures that the average number of common indexes between any pair of working graphs is practically small.

Theorem 4.6. Let $C(G)$ denote the random variable corresponding to the average number of common indexes between any two different working graphs selected from the set G of all θ working graphs, built by I-TRS. If θ_c satisfies the following, for each tag c ,

$$\theta_c \geq \frac{r\theta}{\alpha\delta(\theta - 1) + r} \quad (4.8)$$

then $C(G) \leq \alpha$ holds with at least $(1 - \delta)$ probability.

Proof. $C(G)$ can be calculated as follows.

$$C(G) = \frac{1}{\theta(\theta - 1)} \sum_{g_i \in G} \sum_{I_j \in I(g_i)} \sum_{g_t \in G \setminus g_i} \phi(I_j, g_t) \quad (4.9)$$

where $I(g_i)$ is the set of indexes used by the working graph g_i . The indicator function $\phi(I_j, g_t)$ returns 1 if $I_j \in I(g_t)$ and 0 otherwise.

Let $X(I_j)$ be the random variable denoting the total count of index I_j appearing in all θ working graphs in G . We have:

$$X(I_j) = \sum_{g_i \in G} \phi(I_j, g_i) \quad (4.10)$$

For each tag, we have θ_c candidate indexes, and they share the same probability to be chosen by a working graph. Therefore, $X(I_j)$ follows the Binomial distribution with probability $\frac{1}{\theta_c}$, that is, $X(I_j) \sim B(\theta, \frac{1}{\theta_c})$ and $\mathbb{E}[X(I_j)] = \frac{\theta}{\theta_c}$. Then, we can rewrite Equation 4.9 as:

$$\begin{aligned} C(G) &= \frac{1}{\theta(\theta-1)} \sum_{g_i \in G} \sum_{I_j \in I(g_i)} \left(\sum_{g_t \in G} \phi(I_j, g_t) - \phi(I_j, g_i) \right) \\ &= \frac{1}{\theta(\theta-1)} \sum_{g_i \in G} \sum_{I_j \in I(g_i)} (X(I_j) - 1) \\ &= \frac{r}{\theta-1} (X(I_j) - 1) \end{aligned} \quad (4.11)$$

Taking expectation at both side, we get:

$$\mathbb{E}[C(G)] = \frac{r}{\theta-1} (\mathbb{E}[X(I_j)] - 1) = \frac{(\theta - \theta_c)r}{(\theta-1)\theta_c} \quad (4.12)$$

By Markov's Inequality, we have:

$$\Pr[C(G) \leq \alpha] \geq 1 - \frac{\mathbb{E}[C(G)]}{\alpha} \quad (4.13)$$

Let $\delta = \frac{\mathbb{E}[C(G)]}{\alpha}$, we have:

$$\begin{aligned} \alpha\delta &= \mathbb{E}[C(G)] = \frac{(\theta - \theta_c)r}{(\theta-1)\theta_c} \\ \implies \theta_c &= \frac{r\theta}{\alpha\delta(\theta-1) + r} \end{aligned} \quad (4.14)$$

This completes the proof. $\square$

Empirical Study. Table 4.1 and Figure 4.5 provide an experimental study for Theorem 4.6. We show in Table 4.1 that the difference of influence estimated by TRS (approximation method) and I-TRS (our indexing method) is within 0.2% in all cases. Figure 4.5(a) provides justification for our good performance. Empirical results confirm

Method	Inf. Spread in Targets (%), #Seeds(k) = 20				Inf. Spread in Targets (%), #Tags(r) = 20			
	$r=5$	$r=10$	$r=20$	$r=30$	$k=5$	$k=10$	$k=20$	$k=30$
<i>TRS</i>	24.70	48.49	71.26	75.11	42.38	61.69	71.48	73.36
<i>I-TRS</i>	24.58	48.6	71.33	74.92	42.51	61.66	71.33	73.29
Deviation	-0.12	+0.11	+0.07	-0.19	+0.13	-0.03	-0.15	-0.07

Table 4.1: Accuracy comparison between TRS and I-TRS with optimal tags, *Yelp*. #targets=3K, $\delta = 0.01$, $\alpha = 1$.

Figure 4.5: Implication of Theorem 4.6, *Yelp*. #targets=3K, #seeds=10, $\delta = 0.01$, $\alpha = 1$. Optimal tags are used.

that, when we set the probabilistic upper bound $\alpha = 1$ (and $\delta = 0.01$), the average number of pairwise common index is only 0.01, which is similar to its expectation (Equation 4.12). This means that the average number of common indexes between any pair of working graphs is practically small, which leads to the similar influence estimation as the approximation method, TRS. Moreover, Figure 4.5(b) shows that our θ_c is much smaller than θ (about 3 ~ 4 orders of magnitude), leading to smaller index size and building time.

Time Complexity. Following [164], the time complexity of TRS algorithm can be expressed as: $\mathcal{O}(k(m+n)(c_1 + c_2 + pc_3) \log n/\epsilon^2)$, where k is the number of seeds, m and n the number of edges and nodes, respectively. The term $\mathcal{O}(k(m+n) \log n/\epsilon^2)$ represents the number of edges examined to ensure $(1 - 1/e - \epsilon)$ approximation guarantee, whereas $(c_1 + c_2 + pc_3)$ denotes the cost of processing each edge. In particular, c_1 is the cost to read an edge information (i.e., its endpoints, probability), c_2 the cost of a coin toss (i.e., random number generation) for checking its existence, and c_3 the cost to check whether its source node is visited or not during the reverse BFS, if the coin toss succeeded (with edge probability p). In contrast, for our indexing algorithm, the time complexity is: $\mathcal{O}(k(m+n)pt(c_1 + c_3) \log n/\epsilon^2)$, where t is the average number of tags per edge. This

is because our indexes remove the coin toss cost, instead an edge may exist in multiple tag's indexes. Specifically, for smaller t , we find that $pt(c_1 + c_3) < (c_1 + c_2 + pc_3)$, and this difference is magnified by a large number $\mathcal{O}(k(m + n) \log n/\epsilon^2)$ of edges examined. This explains the efficiency improvement due to our indexing method over the state-of-the art TRS algorithm by a margin of 30% (Figure 4.3).

The further improved version of our proposed index scheme, call *Lazy and Local TRS Index (LL-TRS)*, can be found in our full paper [91].

4.4 Algorithms for Finding Tags

Given a seed set S , a target set T , and a budget r on the number of tags, we develop algorithms to find the top- r tags that maximize the expected influence spread from S to T . The problem is **NP**-hard, hard to approximate, and neither submodular, nor supermodular for inclusion of tags (see our full paper [91]). Thus, unlike our top- k seeds finding algorithm in the earlier section, a similar hill-climbing approach does not come with a provable approximation guarantee. Therefore, *we adopt a greedy heuristic analogous to [98], and improve its quality (by a margin of 30%) with a novel batch-paths selection strategy.*

We briefly introduce the individual paths selection method [98] in Section 4.4.1, point out its shortcomings in Section 4.4.2, and finally describe our novel batch-paths selection algorithm in Section 4.4.3.

4.4.1 Individual Paths Selection

Intuitively, what matters in computing the influence spread from the seed set S to the target set T is the set of highly probable paths connecting them [41, 102, 104], where the probability of a path is determined by multiplying the edge probabilities on that path. Motivated by this, a two-step approach could be developed as follows.

- First, we select the top- l most probable paths between every seed-target pair (in a tag agnostic manner). It could be implemented with the Eppstein's algorithm [60, 98]. Let us denote by $\mathcal{P}$ the set of all paths selected as above.

Figure 4.6: Tags selection. Seed set = $\{A, B, C\}$, target set = $\{G, H, I\}$.

- Next, we iteratively include these paths from $\mathcal{P}$ into our solution $\mathcal{P}_1$ that maximally increase the influence spread (estimated via MC-sampling) from S to T , while still maintaining the budget on total r tags over all included paths.

Since the algorithm includes only one path at a time in the second step, we refer to this method as *individual paths selection*.

4.4.2 Shortcomings of Individual Paths Selection

We demonstrate with the following example that the individual paths selection approach has several critical shortcomings.

Example 4.2. In Figure 4.6, there are total 14 paths between the seed set $\{A, B, C\}$ and the target set $\{G, H, I\}$. They are as follows. $A \rightarrow G : \{e_3e_8, e_1e_7\}$, $A \rightarrow H : \{e_1e_4e_{10}, e_9\}$, $A \rightarrow I : \{e_1e_4e_{11}\}$, $B \rightarrow G : \{e_7\}$, $B \rightarrow H : \{e_4e_{10}\}$, $B \rightarrow I : \{e_4e_{11}\}$, $C \rightarrow G : \{e_2e_7\}$, $C \rightarrow H : \{e_2e_4e_{10}, e_5e_{10}\}$, $C \rightarrow I : \{e_2e_4e_{11}, e_5e_{11}, e_6e_{12}\}$. Let us apply the individual paths selection algorithm to find the top-3 tags in this example. In the first iteration, path e_3e_8 will be included in the solution, since it has the highest individual expected spread $\sigma(S, T, \{e_3e_8\}) = 0.81$. In the second round, because of budget on tags: $r = 3$, there are only three choices: $\{e_7, e_9, e_6e_{12}\}$. Considering the marginal gain, e_6e_{12} will be included. Therefore, we obtain the top-3 tag set $\{c_2, c_3, c_5\}$. The expected spread achieved by this top-3 tag set is: $\sigma(S, T, \{e_3e_8, e_6e_{11}\}) = 0.81 + 0.63 = 1.44$.

Unfortunately, the best 3 tags in this example are c_4, c_5, c_6 , which result in an expected influence spread $\sigma(S, T, \mathcal{P}_1) \approx 2.61$, where $\mathcal{P}_1 = \{e_7, e_9, e_2e_7, e_4e_{10}, e_5e_{10}, e_2e_4e_{10}, e_5e_{11}, e_2e_4e_{11}, e_6e_{12}\}$. It is obvious that in Example 4.2, the individual paths inclusion algorithm makes a bad choice in the first round, and can never recover from it. Below we investigate the key reasons behind this problem.

(1) Although the path e_3e_8 has the highest probability (0.81) among all paths, the tags on this path, c_2 and c_3 , do not appear elsewhere in the graph. Our final goal is to select optimal tags, but not the paths themselves. As an example, the path e_4e_{10} has only 0.56 probability; however, selecting e_4e_{10} will also activate paths e_7 , e_5e_{10} , and e_6e_{12} , since they all share same tags c_4 and c_5 , leading to a total expected spread $\sigma(S, T, \{e_4e_{10}, e_5e_{10}, e_6e_{12}, e_7\}) = 0.8[1 - (1 - 0.7)(1 - 0.9)] + 0.9 \times 0.7 + 0.8 \approx 2.21$. This implies that paths sharing same tags must be evaluated together.

(2) Though the path e_7 has a slightly lower probability (0.8) compared to that of e_3e_8 (0.81), including it only costs 1 tag, while e_3e_8 consumes 2 tags. Therefore, it is more appropriate to evaluate the marginal gain per new tag included, rather than that of a path.

(3) Some paths, e.g., e_7 , e_1e_7 , and e_2e_7 , are actually equivalent in the original graph. All the source nodes shall be activated in the beginning, therefore the existence of edges between them has no impact on influence diffusion. By removing edges between source nodes beforehand, we can avoid including redundant paths, which can significantly reduce the computation cost (only 8 out of 14 paths remaining for Example 4.2).

Inspired by above, we develop a batch-paths selection algorithm that achieves higher accuracy with the included tags.

4.4.3 Batch-Paths Selection

We start with defining a *path-batch* $\mathcal{P}(C)$ for a tag set C , which is a set of paths such that for every path $P \in \mathcal{P}(C)$, its tag set is equal to C . For example, $\mathcal{P}(c_4, c_5) = \{e_4e_{10}, e_5e_{10}\}$, whereas $\mathcal{P}(c_5) = \{e_6e_{12}\}$. Clearly, path-batch is our measuring unit in the novel algorithm. All paths located in the same path-batch contain same tags. Hence, activating one of them will activate other paths in the batch. Furthermore, since the tag sets of path-batches can have subset relationships, we should also find those batches

Figure 4.7: Batch-paths lattice for Figure 4.6. Dashed circles present redundant search space due to edges between seeds (removed).

Figure 4.8: Updates of batch-paths lattice on inclusion of tags $\{4, 5\}$

which are “dominated” by the current batch (based on subset relationships over tag sets), and activate them. For example, activating $\mathcal{P}(c_4, c_5)$ will activate all paths in the path-batch $\mathcal{P}(c_5)$. We define the *descendent* of a path-batch $\mathcal{P}$ as:

$$Des\mathcal{P}(C) = \{\mathcal{P}(C_1) : C_1 \subseteq C\} \quad (4.15)$$

We are now ready to define the objective of our greedy approach, that iteratively includes a path-batch $\mathcal{P}^*$ having the maximum marginal gain ratio (i.e., marginal gain/new tag), while still maintaining the budget r on the number of included tags.

$$\mathcal{P}^* = \arg \max_{\mathcal{P} \in \mathcal{PB} \setminus \mathcal{PB}'} \left[\frac{\sigma(S, T, Des\mathcal{P} \cup \mathcal{PB}') - \sigma(S, T, \mathcal{PB}')}{C(\mathcal{P})} \right] \quad (4.16)$$

In the above equation, $\mathcal{PB}$ is the collection of all path-batches, whereas $\mathcal{PB}'$ is the set of already included path-batches in our solution. $Des\mathcal{P}$ denotes the descendent of the path-batch $\mathcal{P}$, and $C(\mathcal{P})$ is the number of new tags in the path-batch $\mathcal{P}$.

Algorithm 5 Batch-Paths Selection Algorithm

Input: path set $\mathcal{P}$ from source set S to target set T , a positive integer r

Output: A subset of paths $\mathcal{P}_1 \subseteq \mathcal{P}$ and its tag set C_1 whose size is not larger than r

- 1: $\mathcal{P}_1 = \phi, C_1 = \phi$
 - 2: Construct path-batches $\mathcal{PB}$ and the lattice with $\mathcal{P}$
 - 3: **while** $|\mathcal{PB}| > 0$ and $C_1 < r$ **do**
 - 4: Find optimal path-batch $\mathcal{P}^*$ to include next (Equation 4.16)
 - 5: Add all the paths in $\mathcal{P}^*$ and its descendants into $\mathcal{P}_1$
 - 6: Add tags of $\mathcal{P}^*$ to C_1
 - 7: Update the lattice to remove tags of $\mathcal{P}^*$ from it (Figure 4.8)
 - 8: **end while**
 - 9: **return** Selected tag set C_1
-

Efficient Algorithm. To find the optimal path-batch at every iteration (by following Equation 4.16), we build a *path-batch lattice* as shown in Figure 4.7. (1) All the edges between source nodes are removed beforehand. (2) Each node in the lattice represents a path-batch, labeled by its tag set. (3) In each level of the lattice, the nodes (i.e., path-batches) share same number of tags. (4) Each node in the lattice points to all the nodes in the successive lower level whose tag set is a subset of its own.

To construct the lattice, we consider the paths in $\mathcal{P}$, and filter out those paths which violate the budget r on the number of tags. The lattice is initially empty. For each path in $\mathcal{P}$, if its tag set is the same as some existing batch in the lattice, we add the path into that batch. Otherwise, we create a new batch containing this path. When a new path-batch is generated, we locate its appropriate level within the lattice according to its tag set size. Finally, we add links from each node to all other nodes in the successive lower level whose tag set is a subset of its own. Below we demonstrate it with an example which iteratively includes path-batches from the lattice into our solution.

Example 4.3. Let us apply the batch-paths inclusion technique to find the top-3 tags in Figure 4.6. The corresponding lattice is presented in Figure 4.7. We select the path-batch (with its descendent) $Des\mathcal{P}(c_4, c_5) = \{e_4e_{10}, e_5e_{10}, e_7, e_6e_{12}\}$ as the best batch in the first round (by following Equation 4.16), which generates an expected spread $\sigma(S, T, Des\mathcal{P}(c_4, c_5)) = 0.8[1 - (1 - 0.7)(1 - 0.9)] + 0.9 \times 0.7 + 0.8 \approx 2.21$. After the inclusion of $Des\mathcal{P}(c_4, c_5)$ in our solution, we update the lattice with the remaining tags and get new one as shown in Figure 4.8. Since only one more tag can be selected (as $r=3$), the batch $\{c_6\}$ is selected by following Equation 4.16). The final tag set selected

by our batch-paths inclusion method is $\{c_4, c_5, c_6\}$, and they can bring 2.61 total expected spread, which is higher than that of the individual-paths inclusion (1.44).

Time Complexity. We denote the set of paths as $\mathcal{P}$ and the collection of path-batches as $\mathcal{PB}$. The time complexity to construct the lattice is $\mathcal{O}(|\mathcal{P}| \cdot |\mathcal{PB}| + |\mathcal{PB}|^2)$, the first term is due to grouping the paths into batches and forming the nodes of the lattice, and the second term is due to adding links in the lattice.

The procedure to find the top- r tags requires $\mathcal{O}(r|\mathcal{PB}|w(n' + m'))$ time when using MC-sampling, where w is the number of MC samples, n' and m' are the number of nodes and edges, respectively, of the subgraph induced by the paths in $\mathcal{P}$. The term $w(n' + m')$ is due to estimating the influence spreads of different path-batches via MC-sampling. If using indexed reverse sketching method, it requires $\mathcal{O}(\theta|\mathcal{P}| + r|\mathcal{PB}||\mathcal{P}|)$. The first term is for checking each path's existence in each RR index. The second term is due to computing each batch's coverage in the RR indexes. We note that the number of paths in each batch is at most $|\mathcal{P}|$. Therefore, the total complexity is $\mathcal{O}(\theta|\mathcal{P}| + a|\mathcal{PB}||\mathcal{P}| + b|\mathcal{PB}|w(n' + m'))$, where $a + b = r$, the budget on the number of tags.

Some further improvement about the index for influence spread computation can be found in our full paper [91].

4.5 Proposed Iterative Algorithm

We first initialize the seed set and the tag set (as discussed in Section 5.3 in our complete paper [91]). Next, the framework alternatively and in an iterative manner, optimizes the top- k seed users and the top- r relevant tags, assuming that the others are fixed. These are computed by the seeds finding algorithm and the tags finding algorithm, respectively. At the end of every round, we verify if a fixed-point is reached, i.e., the top- k seeds and the top- r tags achieve a similar influence spread in two successive rounds. Our proposed iterative algorithm, in essence, is similar to various classical data mining and machine learning algorithms, e.g., K-means and Expectation-maximization. We prove in [91] that our method converges to a local optimum, because at every round, the expected influence spread within the target set monotonically increases until convergence.

Theorem 4.7. *During the course of our iterative algorithm, the expected spread within the target set monotonically increases.*

Algorithm 6 Complete Algorithm for Jointly Finding k -Seeds and r -Tags

Input: Graph $\mathcal{G} = (V, E, P)$, target set T , budgets k and r

Output: Optimal seed set S^* and tag set C^*

- 1: Initialize S^* of size k , C^* of size r
 - 2: **while** not converge **do**
 - 3: Find optimal S^* assuming given C^*
 - 4: Find optimal C^* assuming given S^*
 - 5: **end while**
 - 6: **return** S^*, C^*
-

Proof. Let $S^{(t)}, C_1^{(t)}$ denote the sets of top- k seeds and the top- r tags, respectively, at the start of the t -th iteration. In the current iteration, we first optimize the seed set, given the tag set. Therefore,

$$\sigma(S^{(t+1)}, T, C_1^{(t)}) \geq \sigma(S^{(t)}, T, C_1^{(t)}) \quad (4.17)$$

Next, we optimize the tag set, given the seed set as obtained above. Hence, the following holds.

$$\sigma(S^{(t+1)}, T, C_1^{(t+1)}) \geq \sigma(S^{(t+1)}, T, C_1^{(t)}) \quad (4.18)$$

Combining Equations 4.17 and 4.18, the expected spread within target set monotonically increases in successive iterations. $\square$

Finally, we note that our algorithm iterates the objective function $\sigma(S, T, C_1)$ that is bounded above, and whose domain is a finite set. Coupled with Theorem 4.7, we guarantee that the algorithm converges to a local optimum after a finite number of iterations.

4.6 Initialization for Iterative Algorithm

Uniform at Random Initialization. A simple way to initialize the seed set and the tag set is uniform at random selection from the set of all nodes and all tags, respectively. Intuitively, this initializing method is cheap, but might not always be of good quality, leading to higher convergence time for Algorithm 6. Below, we propose the following initialization methods aiming for faster convergence.

Influence Maximization-based Seeds Initialization. We run a classical influence maximization algorithm (e.g., [164]) to find the top- k seed nodes that maximize the

Table 4.2: Characteristics of datasets.

Name	#Nodes	#Edges	#Tags	Edge Prob: Mean, SD, Quartiles
lastFM	1 322	13 842	78	0.26, 0.23, {0.06, 0.2, 0.41}
DBLP	704 266	4 727 290	230	0.26, 0.15, {0.18, 0.18, 0.33}
Yelp	125 368	808 909	195	0.33, 0.25, {0.18, 0.26, 0.5}
Twitter	6 294 565	11 063 034	500	0.27, 0.14, {0.18, 0.18, 0.33}

influence spread over the target set, while considering all tags. This leads to a good-quality initial seed set (and therefore, faster convergence); however, the initialization phase becomes more expensive.

Frequency-based Tags Initialization. Generally, those tags appearing frequently among target nodes’ incident edges are more relevant. Therefore, we aggregate the total probability for each tag to appear in all target nodes’ incident edges. Next, we select the top- r tags with the highest accumulative probability as initial tags.

In our experiments, we compare some combination of the aforementioned initializing methods, and analyze their trade-offs.

Frequency-based Search Space Elimination. We compute the aggregated probability for each tag. Tags with low aggregated probability shall be either in too few edges, or having too low edge probabilities. Such tags generally contribute less in influence diffusion, and we remove them beforehand from the search space.

4.7 Experimental Results

4.7.1 Experimental Setup

We perform experiments to demonstrate the accuracy (i.e., influence spread within target set), efficiency, and scalability of our algorithms (averaged over 10 runs). The code is implemented in C++ and executed on a single core of a 256GB, 2.40GHz Xeon server.

Dataset. We use four real-world social networks (Table 4.2). **(1) lastFM.** We obtain this dataset from [14], which is a music-listening history record. Nodes denote users and edges represent their friendships. **(2) DBLP.** *DBLP* (<http://dblp.uni-trier.de/xml>) is a well-known collaboration network. We downloaded it on March 31, 2017. Each node is an author and edges denote their co-author relations. **(3) Yelp.** *Yelp* (<https://www.yelp.com/>

dataset_challenge) is a user-business reviewing network. Every node is a user. Edges are generated based on their friendship, and directed according to the time-stamps of their reviews for same businesses. **(4) Twitter.** In *Twitter* (<http://snap.stanford.edu/data/>), nodes are users and edges are their re-tweet relationships. *lastFM* and *DBLP* graphs are undirected, whereas *Yelp* and *Twitter* are directed.

Tags and Edge Probabilities. **(1) lastFM.** The tags are given by music styles. **(2) DBLP.** The tags are extracted from all paper titles, e.g., database system, neural network, FPGA, etc, based on both their frequency and how well they can represent various sub-areas of computer science. **(3) Yelp.** Users may visit same businesses and each business belongs to some categories (e.g., travel, dance clubs, burger, etc.). We use these categories as tags. **(4) Twitter.** Tags are given by hashtags. For every edge (u, v) and tag c , we compute the frequency (i.e., number of occurrences) of c in (u, v) . Let us denote this frequency by t , then we assign probability $p((u, v)|c) = 1 - e^{-t/a}$ [141]. The intuition is that the more the times the tag c appears on the edge (u, v) , the higher the chance (i.e., the probability) that u influences v for that tag. As an example, for *lastFM* we calculate the frequency t for every music style c based on each node pair’s common listening history. We set $a = 5$ for *DBLP* and *Twitter*, $a = 10$ for *Yelp*, and $a = 1000$ for *lastFM* (since each node pair has a large common listening history, we set higher a in this dataset to avoid unusually high connectivity between node pairs).

Parameters Setup. **(1) Seed set size (k).** We vary k from 5 to 100. **(2) Tag set size (r).** We vary r from 5 to 30. **(3) Target set size ($|T|$).** By default, as the target set, we use 500 nodes for *lastFM*, and 3K nodes for other datasets. We, however, demonstrate scalability by considering up to 50K target nodes. **(4) RR-Sketches.** ϵ is set as 0.1 following [164]. Probability parameter δ and upper bound of common index parameter α in Theorem 4.6 are set as 0.01 and 1, respectively, and the local-region parameter h as 3. We also verify the sensitivity of our methods with respect to these parameters in our full paper [91]. **(5) Tags finding.** Number of paths per seed-target pair for our Batch-Paths selection algorithm is set as 10, the optimality of which was demonstrated in our complete paper [91].

Competing Methods and Initialization. We compare our baseline greedy method (see our full paper [91] for details) with the proposed iterative algorithm. For the iterative

 Figure 4.9: Accuracy and efficiency analysis to adjust seed set size k . $r = 10$, $|T| = 3K$.

 Figure 4.10: Accuracy and efficiency analysis to adjust tag set size r . $k = 20$, $|T| = 3K$.

algorithm, we consider four initialization techniques (see our full paper [91] for details). We denote random initialization on seeds and tags as RS and RT , respectively. Influence maximization-based seed initialization is denoted as IMS , and frequency-based tags initialization is denoted as FT . We find that the best initializing method is $RS + FT$, while other combinations will be evaluated in Section 4.7.2.

4.7.2 Accuracy and Efficiency Analysis

We present influence spread within target set (accuracy) and running time (efficiency) comparison between baseline greedy and our iterative algorithms, while varying two major parameters, seed set size k and tag set size r , under different initializing conditions.

Varying Seed Set Size k . As shown in Figure 4.9, the influence spread of our method significantly outperforms the baseline in accuracy, and executes in similar time. Influence spread rises with larger k . Such growth is more significant when k is small for larger datasets, for example on *Twitter* between $k = 5$ and $k = 10$ the influence increase is from 62% to 77%. On the contrary, the running time is more sensitive to larger k , whose increasing tendency is near linear to the growth of k . For seeds finding part, Theorem 4.5 requires bigger θ for bigger k (generally less than twice when k doubles). For tags finding part, $2k$ seed size induces, on average, twice the number of paths, which roughly requires double evaluating time. Therefore, our running time increases linearly with bigger k .

Varying Tag Set Size r . For accuracy, the increasing tendency of r is similar to that of k . With a few important tags, we can already influence most targets. As an example, only

Table 4.3: Accuracy and efficiency for different initialization methods, *Yelp*. $r = 20$, $|T| = 3K$.

Method	Inf. Spread in Targets (%)					Running Time (Sec)				
	# Seeds (k)					# Seeds (k)				
	5	10	20	50	100	5	10	20	50	100
<i>RS+RT</i>	42	64	75	82	84	6 400	11 100	17 100	27 200	39 400
<i>IMS+RT</i>	42	66	75	82	84	5 800	9 500	14 800	23 500	34 200
<i>RS+FT</i>	48	74	78	82	84	3 700	7 700	12 000	20 100	30 200
<i>IMS+FT</i>	49	74	78	82	84	5 000	9 200	14 000	22 300	33 100

Table 4.4: Case study: top-10 tags found for influence maximization in three target cities.

Target City	Top-10 Tags
Las Vegas	Arts & Entertainment, Travel, Hotels, Buffets, Dance Clubs, Desserts, Chinese, Mediterranean, Japanese, Burger
Toronto	Canadian, Chinese, Japanese, Pubs, Coffee & Tea, Italian, Arts & Entertainment, Comfort Food, Chiropractors, Physical Therapy
Pittsburgh	Burger, Mexican, Seafood, Grocery, Arts & Entertainment, Italian, Sport Bars, Coffee & Tea, Ice Cream & Frozen Yogurt, Specialty Food

top-20 tags can influence about 70% targets in the Yelp dataset. Moreover, the influence spread of our iterative algorithm significantly outperforms the greedy baseline in influence spread.

The running time increases at a higher rate when r is small, which is because the change in influence spread is significant. When r is large enough, adding more tags changes little in the influence spread, leading to a lower increase rate in the running time.

Varying Initialization. As shown in Table 4.3, different initializing methods eventually lead to similar influence spread. In some specific cases, e.g., with $k = 5$ or 10 seed nodes in *Yelp*, starting with *RT* gets stuck in a local optimum, which is about 8% less than initializing with *FT*. However, with large enough k (e.g., $k = 50$ or 100), high-quality seeds and tags are always found in our solution.

More analyses about scalability and sensitivity can be found in our full paper [91].

4.7.3 Case Study

To demonstrate the effectiveness of our problem, we conduct a case study on the real-world *Yelp* social network dataset, downloaded from yelp.com.sg/dataset/challenge. This

Figure 4.11: Influence spread comparison using cross-city tags. $k=10$ seeds, $r=10$ tags, Y-axis denotes the % of spread by 10 selected tags, over total spread by all 195 tags.

is a reviewing-based social network, where users can assign reviews to a business (e.g., a restaurant, a travel agency, etc.) after visiting it. Each business is characterized by a few categorical attributes (e.g., Pubs, Seafood, Hotels, Clubs, Arts, etc.), which we use as tags in our problem setting. There are total 195 distinct categories in this dataset. For a pair of friends u and v in the social network, and for their reviews on every common business category, we compute the influence direction and probability — based on their time-stamps of visits and their frequency of reviewing the same category. More details about the dataset can be found in Section 2.4.

As shown in Table 4.4, the most relevant tags found by our method, for maximizing the influence spread in each target city, are quite different, e.g., for *Las Vegas*, popular categories include *Dance Clubs*, *Arts & Entertainments*, and *Travel*, whereas for *Pittsburgh*, various food items appear in the top-10 list, including *Specialty Food*, *Seafood*, *Coffee & Tea*, *Ice Cream & Frozen Yogurt*. Hence, if someone wants to promote her hotels via tweets or blog posts in each one of those cities, the best choice of advertisement tags are not the same (entertainment facilities for *Las Vegas*, while foods for *Pittsburgh*). Furthermore, Figure 4.11 shows that an optimal set of advertisement tags for a specific city may not work in the same way for other cities. It is interesting to note that even only with 10 most relevant tags (found by our algorithm), we can obtain nearly 90% of the influence spread, as compared to that achieved by all 195 tags present in the dataset. This case study demonstrates the usefulness and relevance of our novel query, which we investigate in this work.

4.8 Summary

In this work, we introduced and investigated the novel problem of jointly finding the top- k seed nodes and the top- r relevant tags for targeted influence maximization in a social network. We deeply characterized the problem, proving that it is **NP**-hard and also hard to approximate. To solve this problem, we refined both state-of-the-art seeds finding and tags finding algorithms together with indexing and batch-path selection methods, and thus improved their efficiency by 30% and accuracy up to 30%, respectively. Finally, we developed an iterative framework, together with smart initialization techniques, that alternatively optimizes the top- k seeds and the top- r tags, and we proved that the influence spread monotonically increases until convergence. Our experimental results demonstrated that the proposed iterative algorithm achieves high influence spread compared to baselines, and converges fast in 3 iterations.

Chapter 5

PERC System: Probabilistic Entity Resolution with Crowd Errors

Entity Resolution (ER) is a critical step in data cleaning and analytics. Although many machine-based methods existed for ER task, crowdsourcing is becoming increasingly important since humans can provide more insightful information for complex tasks, e.g., clustering of images and natural language processing. However, human workers still make mistakes due to lack of domain expertise or seriousness, ambiguity, or even malicious intent. To this end, we present a system¹, called PERC (probabilistic entity resolution with crowd errors), which adopts an uncertain graph model to address the entity resolution problem with noisy crowd answers. Using our framework, the problem of ER becomes equivalent to finding the maximum-likelihood clustering. In particular, we propose a novel metric called “reliability” to measure the quality of a clustering, which takes into account both the connected-ness inside and across all clusters. PERC then automatically selects the next question to ask the crowd that maximally increases the “reliability” of the current clustering. This significantly reduces the overall crowdsourcing cost — by 50%, as well as improves its quality by 15%, which is demonstrated in our empirical results (Table 5.1 and [184]).

To the best of our knowledge, PERC is the first demo system on *crowdsourced entity resolution with human errors*, aiming to present an uncertain graph-based solution, which is efficient, effective, and user-friendly in practice. Other recent demos in this area include

¹This work is published as the following two papers: (1) [CIKM’17] Vijaya Krishna Yalavarthi, Xiangyu Ke, and Arijit Khan, *Select Your Questions Wisely: For Entity Resolution with Crowd Errors*. (2) [VLDB’18] (Demo) Xiangyu Ke, Michelle Teo, Arijit Khan, and Vijaya Krishna Yalavarthi *A Demonstration of PERC: Probabilistic Entity Resolution with Crowd Errors*.

Table 5.1: Crowdsourcing cost reduction by PERC over state-of-the-art approaches: We present the number of crowdsourcing questions required to achieve a certain accuracy (0.8 F1-measure for *Cora* dataset, and 0.9 for others) for various methods.

<i>datasets</i>	# crowdsourced questions			% crowdsourcing cost reduction by PERC over	
	MinMax [77]	bDense [172]	PERC [this work]	MinMax	bDense
Allsports	13.6K	16.0K	11.7K	13.97%	26.87%
Gymnastics	1.3K	1.5K	0.8K	38.46%	46.67%
Landmarks	11.0K	8.0K	5.9K	46.36%	26.25%
Cora	22.5K	14.0K	7.2K	68.00%	48.57%

NADEEF/ER [58], KATARA [45], and CrowdCleaner [169]. In particular, NADEEF/ER is a rule-based and interactive ER system without crowdsourcing, whereas both KATARA and CrowdCleaner are crowdsourced ER systems without explicitly considering crowd errors. Hence, these past demonstrations are different from ours.

5.1 Preliminaries

5.1.1 Background

Entity Resolution (ER). An ER algorithm receives an input set of records $R = \{r_1, r_2, \dots, r_n\}$ and a pairwise similarity function F , and it returns a set of matching pair of records: $\mathbb{C} = \{R_1, R_2, \dots, R_m\}$, such that, $R_i \cap R_j = \emptyset$ for all i, j , and $\cup_i R_i = R$. We call each R_i a *cluster* of R , and each cluster represents a distinct real-world entity. The partition of R into a set of clusters is called a *clustering* $\mathbb{C}$ of R . If r_1 and r_2 are matching (non-matching), they are denoted by $r_1 = r_2$ ($r_1 \neq r_2$).

An ER algorithm generally obeys the two following relations.

Transitivity. Given three records r_1, r_2 , and r_3 , if $r_1 = r_2$ and $r_2 = r_3$, then we have $r_1 = r_3$.

Anti-transitivity. Given three records r_1, r_2 , and r_3 , if $r_1 = r_2$ and $r_2 \neq r_3$, then we have $r_1 \neq r_3$.

Thus, a clustering $\mathbb{C}$ of the input set R of records is transitively closed. One can derive the following theorem combinatorially. We omit the proof due to limitation of space.

Figure 5.1: Possible worlds of an uncertain graph: Three possible worlds G_5 , G_6 , G_7 are not clusterings, as they are not transitively closed. For example, in G_5 , $A = C$ and $C = B$, but $A \neq B$, thus violating transitivity.

Theorem 5.1. *For n records, there can be $(2^n - n)$ different clusterings, where each cluster in some clustering must have between $(1, n)$ records.*

Crowdsourced ER. We use a crowdsourcing platform such as Amazon’s Mechanical Turk (AMT), which provides APIs for conveniently using a large number of human workers to complete micro-tasks (also known as Human Intelligent Tasks (HITs)). To identify whether two records belong to the same entity, we create an HIT for the pair, and publish it to AMT with possible binary answers: A worker needs to submit ‘YES’ if she thinks that the record pair is matching, and ‘NO’ otherwise.

For mitigating crowd errors, we allow multiple workers to perform the same HIT. We then assign an edge with probability $p(r_i, r_j)$ between two records r_i and r_j , where $p(r_i, r_j)$ is the ratio of crowd workers who voted YES on the question if r_i and r_j are same entity.

Uncertain Graph. Every HIT creates an uncertain, undirected edge between the respective record pair, thereby generating an uncertain, undirected graph $\mathcal{G} = (R, E, p)$. Each record $r_i \in R$ denotes a node in the graph, $E \subseteq R \times R$ represents the set of edges between the record pairs that were crowdsourced, and $p(e) \in (0, 1)$ is the probability of the edge $e \in E$ as derived earlier. In our context, it is important to note that $p(e) = 0$ (i.e., all crowd workers voted non-matching) is *not* equivalent to the edge e being absent in $\mathcal{G}$ (i.e., the pair is not crowdsourced yet).

To reflect independence across different crowdsourcing tasks (i.e., each HIT can be performed by a different set of workers), we employ the well-established notion of possible world, together with the assumption that each edge can be matching or non-matching, independent of other edges [84]. Hence, the uncertain graph $\mathcal{G}$ yields $2^{|E|}$ deterministic graphs (or, possible worlds) $G \subseteq \mathcal{G}$, where each G is a pair (R, E_G) , with $E_G \subseteq E$ are matching edges, and its probability of being observed is given in Equation 5.1.

$$P(G) = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)) \quad (5.1)$$

Next, we have the following observation.

Lemma 5.1. *Every clustering of the input record set R corresponds to some possible world of the uncertain graph $\mathcal{G} = (R, E, p)$. However, every possible world of $\mathcal{G}$ might not be a clustering of R .*

The first part of the lemma is trivial (i.e., follows from the definition of a possible world), whereas the second part holds since every possible world is not transitively closed. We demonstrate this fact with an example in Figure 5.1, where three possible worlds G_5, G_6 , and G_7 of the given uncertain graph are not clusterings.

Since every clustering corresponds to some possible world, we define the likelihood of a clustering as the probability of the respective possible world being observed. In Figure 5.1, the likelihood of the clustering $\{(A, B), (C)\}$ is same as $P(G_4)$, which is 0.288.

5.1.2 Entity Resolution Problem

Given $R, \mathcal{G}$, let us consider a clustering $\mathbb{C} = \{R_1, R_2, \dots, R_m\}$ of R . We define the likelihood of $\mathbb{C}$ as the probability that (1) all edges inside every cluster R_i exist, and (2) all edges across every pair of clusters R_j, R_k do not exist. Since an edge can exist independent of others, we compute the likelihood $L(\mathbb{C})$ as follows.

$$L(\mathbb{C}) = \prod_{R_i \in \mathbb{C}} \left[\prod_{e \in E \cap (R_i \times R_i)} p(e) \right] \times \prod_{\substack{R_j, R_k \in \mathbb{C} \\ j < k}} \left[\prod_{e \in E \cap (R_j \times R_k)} (1 - p(e)) \right] \quad (5.2)$$

We formally introduce the ER problem below.

Problem 5.1 (Entity Resolution). *Given the set R of records and an uncertain graph $\mathcal{G} = (R, E, p)$, find the (transitively closed) clustering $\mathbb{C}$ of R having the highest likelihood $L(\mathbb{C})$.*

The problem of finding the most-likely clustering (also referred to as the maximum-likelihood clustering), however, is **NP**-hard, which can be verified by a polynomial-time reduction from the **NP**-hard correlation clustering problem [172].

Theorem 5.2. *Given an uncertain graph $\mathcal{G} = (R, E, p)$ over records set R , finding the maximum-likelihood clustering of R is **NP**-hard.*

Correlation clustering is the most natural setting for clustering a set of records that are connected by both positive and negative edges [79]. Many approximate and heuristic algorithms were proposed for correlation clustering [59, 172]. Indeed, all prior works such as [77, 172, 176] in the domain of crowdsourced ER, that incorporated human error, also employed correlation clustering. Therefore, in our PERC framework, we apply correlation clustering for the ER problem. We shall first introduce our next crowdsourcing algorithm in the following, which is the key contribution of this work.

5.2 Related Work

Crowdsourcing in Data Management. Recently, crowdsourcing has been adopted in video and image annotations, search relevance, and natural language processing [8, 73].

Several systems have been developed to incorporate human work into a database/mobile system, e.g., CrowdDB, Deco, CrowdSearch, CDAS, CrowdForge, TurKit, and Qurk [120, 126]. There are also studies on leveraging crowd’s ability to improve data management tasks, e.g., selection, sort, skyline, join, mining, classification, and max/top-k [38, 125].

Crowdsourced Entity Resolution (ER). An important problem in crowdsourced ER is to reduce the number of questions asked to workers, e.g., a clustering-based method [174] where each question is a group of records and asks workers to classify the records into different clusters. Demartini et. al. [55] and Jeffrey et. al. [83] designed crowdsourcing systems based on a probabilistic framework, but does not employ transitivity to reduce the crowdsourcing cost. Wang et. al. [175] and Vesdapunt et. al. [173] utilized transitivity to reduce the number of questions. Various models to select high-quality questions were developed in [180, 182]. The most recent work [30] used a partial order approach, which additionally requires each entities having multiple attributes. More importantly, all these works assume no crowd error, or employ majority voting.

Recently, MinMax, PC-Pivot, and DENSE [77, 172, 176] directly incorporated crowd errors in ER tasks. However, as we stated earlier, these methods consider ad-hoc, local features to select next questions, such as individual paths, nodes, or the set of either positive or negative edges. Hence, they generally fail to capture the strength of the entire clustering, resulting in higher crowdsourcing cost in order to achieve a reasonable ER accuracy.

Dealing with Crowdsourcing Errors. Quality control is critical in crowdsourcing [120]. Machine learning techniques have been employed to determine the quality of the crowd, e.g., [53, 85, 89]. Orthogonal to these works, our proposed solution incorporates crowd errors while performing next crowdsourcing and ER tasks.

Entity Resolution Algorithms. Entity resolution (ER), also known as entity reconciliation, deduplication, or record linkage, is well studied in data cleaning and integration. Many ER algorithms have been proposed based on different input settings, e.g., single-pass clustering, star clustering, cut clustering, correlation clustering, and Markov clustering [70, 106]. We used correlation clustering because this is the most natural setting for clustering a set of records that are connected by both positive and negative edges [59]. Besides, our contribution — reliability-based next crowdsourcing question selection is orthogonal to the specific ER method employed.

Figure 5.2: The architecture of PERC system

5.3 Framework Overview

Figure 5.2 presents PERC’s architecture and the interaction between modules. The core modules of PERC — Entity Resolution (ER) and Next Crowdsourcing (NC) modules — are implemented in C++. The Displayer module, which provides the visualization of dynamic and interactive clustering results and next question selecting results, is developed with D3.js library (<http://d3js.org/>). We introduce the four modules of the PERC system in the following.

Interactor. The *Interactor* receives input datasets of various types (e.g., images, text, video), and pre-processes them to identify the records set R . It selects, uniformly at random, a small number record pairs. The *Record Pairs-HITs Transformer* component

then publishes these pairs as HITs (Human Intelligent Tasks) on AMT platform. The *Interpreter* generates or updates the uncertain graph once the crowd answers are obtained. In particular, to identify whether two records belong to the same entity, we create an HIT for the pair, and publish it to AMT with possible binary answers: A worker submits ‘Yes’ if she thinks that the record pair is matching, and ‘No’ otherwise. For mitigating crowd errors, we allow multiple workers to perform the same HIT. We then assign an edge with probability $p(r_i, r_j)$ between two records r_i and r_j , where $p(r_i, r_j) \in (0, 1)$ denotes the ratio of crowd workers who voted Yes on the question if r_i and r_j are same entity. We denote by E the set of edges as formed above.

Entity Resolution Processor. On receiving an uncertain graph $\mathcal{G} = (R, E, p)$ as formed above, our *Entity Resolution processor* applies a greedy correlation clustering algorithm (e.g., the spectral-connected-components method [172]) to generate the maximum likelihood clustering (MLC) of the uncertain graph. After every crowdsourcing task, we add an uncertain edge between the respective record pair, thereby updating $\mathcal{G}$. Then, the *MLC Update Detector* verifies if the previous MLC needs to be updated or not.

Next Crowdsourcing Questions Selector. After the initial MLC generation, the *Next Crowdsourcing Questions Selector* iteratively finds the best record pair $\langle r_i, r_j \rangle$ and crowdsources it, until our budget is exhausted, or we already find a complete (uncertain) graph over the records set R . If MLC remains the same in successive iterations, the priority queue consisting of next crowdsourcing questions remains the same, and the question to ask in the current round is easy to decide by simply selecting the best (e.g., top-most) pair from the queue. Otherwise, the *Reliability Evaluator* re-computes the priority queue based on the metric called “reliability” of the new MLC. We shall formally introduce this novel criteria in Section 5.4.2.

Displayer. Our system also contains a user-friendly *Displayer* module for interactive visualization. The *ER Result Displayer* shows the dynamic changes of the graph and the clustering result based on the crowd answers obtained so far, and the *NC Result Displayer* presents the next questions to be published in the crowdsourcing platform. Users are allowed to rollback for visualizing every batch of questions asked previously and their corresponding ER result. If the ground-truth answer is provided, PERC will color the record nodes and presents it together with the real-time ER result.

Figure 5.3: Workflow of PERC

5.4 Core Algorithms

Two primary modules of PERC rely on each other, as illustrated in Figure 5.3, and run their own algorithms. In this section, we briefly introduce the core metrics and algorithms. More technical details can be found in our full research paper [184].

5.4.1 Entity Resolution Processor

The task of *ER Processor* is to find the maximum likelihood (and transitively-closed) clustering for a given uncertain graph $\mathcal{G} = (R, E, p)$. The likelihood of a clustering $\mathbb{C} = \{R_1, R_2, \dots, R_m\}$ is defined as the probability that (1) all edges inside every cluster R_i exist, and (2) all edges across every pair of clusters R_j, R_k do not exist. Since an edge can exist independent of others (i.e., each HIT can be performed by a different set

of workers), we compute the likelihood $L(\mathbb{C})$ as:

$$L(\mathbb{C}) = \prod_{R_i \in \mathbb{C}} \left[\prod_{e \in E \cap (R_i \times R_i)} p(e) \right] \times \prod_{\substack{R_j, R_k \in \mathbb{C} \\ j < k}} \left[\prod_{e \in E \cap (R_j \times R_k)} (1 - p(e)) \right]$$

Therefore, the entity resolution problem is equivalent to finding the maximum likelihood clustering (MLC) over the uncertain graph constructed by the input records set. It can be performed by employing state-of-the-art *correlation clustering* algorithms [59, 172]. The users can select among the following algorithms: *Best*, *First*, *Pivot* and *Spectral-Connected-Components* (SCC). Usually we suggest SCC, since it performs the best in our empirical evaluation, compared to other approaches. It starts from the record pair having the highest probability of being the same entity, given the answers for these two records. If this probability is higher than 0.5, SCC merges the two records into one cluster. In each successive step, the algorithm finds the clusters with the highest probability of being the same entity, given the answers between them. If this probability is higher than 0.5, the two clusters are merged into one cluster. Otherwise, SCC stops and returns the current set of clusters as output. Given two clusters R_i and R_j , SCC computes the probability $Pr(R_i, R_j)$ of merging them as given below.

$$Pr(R_i, R_j) = \frac{\prod_{(r_k, r_l) \in (R_i \times R_j) \cap E} p(r_k, r_l)}{\prod_{(r_k, r_l) \in (R_i \times R_j) \cap E} p(r_k, r_l) + \prod_{(r_k, r_l) \in (R_i \times R_j) \cap \bar{E}} (1 - p(r_k, r_l))}$$

5.4.2 Next Crowdsourcing Processor

Once the MLC is constructed, the *Next Crowdsourcing processor* iteratively finds the the best entity pair $\langle r_i, r_j \rangle$ to crowdsource next.

5.4.2.1 Reliability of a Clustering

Intuitively, our objective is to identify a pair $\langle r_i, r_j \rangle \notin E$, that can improve the quality of the given MLC as much as possible. To this end, we identify the two following “connectedness”-based criteria that determine the quality of a clustering $\mathbb{C} = \{R_1, R_2, \dots, R_m\}$.

- (1) How well each cluster R_i is connected?
- (2) How well every pair of clusters R_j, R_k ($j < k$) is disconnected?

Figure 5.4: Reliability of a clustering

Given a clustering $\mathbb{C} = \{R_1, R_2, \dots, R_m\}$ and the uncertain graph $\mathcal{G} = (R, E, p)$, all edges inside a cluster are called YES edges, whereas the edges across two clusters are referred to as NO edges. If e is an YES edge, we define its existence probability $p_Y(e) = p(e)$. On the other hand, if e is a NO edge, we compute its existence probability as $p_N(e) = 1 - p(e)$. We derive an YES-NO graph $\mathcal{G}_{Y|N}$ from the uncertain graph $\mathcal{G}$ as follows. $\mathcal{G}_{Y|N}$ has the same set of nodes and edges as $\mathcal{G}$, but each edge e in $\mathcal{G}_{Y|N}$ has a binary label $L(e)$, which can be either YES or No, as defined above. For a YES edge e , its probability $p_{Y|N}(e) = p_Y(e)$. For a NO edge e , its probability $p_{Y|N}(e) = p_N(e)$.

Given a clustering $\mathbb{C} = \{R_1, R_2, \dots, R_m\}$ and the YES-NO graph $\mathcal{G}_{Y|N}$, the reliability of $\mathbb{C}$ is defined as the probability that every cluster R_i is connected and every pair of clusters R_j, R_k ($j < k$) is disconnected, i.e.,

$$\begin{aligned}
 Rel(\mathbb{C}) &= \sum_i \log(Connect(R_i)) + \sum_{j < k} \log(Discon(R_j, R_k)) \\
 Connect(R_i) &= \sum_{G \subseteq \mathcal{G}_{Y|N}} [I(R_i, G) \times P(G)] \\
 Discon(R_j, R_k) &= \begin{cases} 0 & ; \text{if } (R_j \times R_k) \cap E = \phi \\ 1 - \prod_{(r_i, r_l) \in (R_j \times R_k) \cap E} (1 - p_N(r_i, r_l)) & ; \text{otherwise} \end{cases}
 \end{aligned}$$

In the above equation, $I(R_i, G)$ is an indicator function over a possible deterministic graph $G \subseteq \mathcal{G}_{Y|N}$ taking value 1 if records in R_i are all connected (by YES edges) in G , and 0 otherwise.

Example 5.1. In Figure 5.4, we compute the reliability of the MLC $\mathbb{C}^* = \{(A, B, C), (D)\}$. We first construct the YES-NO graph on the right. Then, we have: $Connect(A, B, C) = 0.72$, $Connect(D) = 1.0$, and $Discon((A, B, C), (D)) = 1 - (1 - 0.8)(1 - 0.4) = 0.88$. Hence, $Rel(\mathbb{C}^*) = \log 0.72 + \log 1 + \log 0.88 \approx -0.20$.

Table 5.2: Properties of datasets

Dataset	# Records	# Entities	# Record-Pairs Crowdsourced	Crowd Error Rate
<i>All Sports</i>	267	86	35 511	5.67% (10 ques. / pair)
<i>Gymnastics</i>	94	12	4 371	10.65 % (5 ques. / pair)
<i>Landmarks</i>	529	15	30 070	4.82 % (5 ques. / pair)
<i>Cora</i>	949	165	29 281	27.77 % (5 ques. / pair)

5.4.2.2 Next Crowdsourcing Question Selection

The *Next Crowdsourcing Question Selector* computes, for every record pair $\langle r_i, r_j \rangle \notin E$, the improvement in reliability of the current MLC $\mathbb{C}^*$, if one crowdsources the pair, and thereby assigns the corresponding edge probability $p(r_i, r_j)$. Unfortunately, one does not know $p(r_i, r_j)$ apriori, therefore we consider an optimistic scenario. We study this in [184], and show that $Rel(\mathbb{C})$ increases monotonically as we have larger value of $p_{Y|N}(e)$, where e is the newly added edge. Thus, $p_{Y|N}(e) = 1$ leads to the optimal gain. In other words, we select the record pair that maximally increases $Rel(\mathbb{C}^*)$, under such optimistic assumption. Also, we found that the priority of the following record pairs $\langle r_k, r_l \rangle \in (R_i \times R_j) \setminus E$, for a certain R_i and R_j , are the same in a specific round, which helps us avoid redundant computation for pairs across the clusters. Finally, we develop efficient algorithms [184] based on the Monte-Carlo sampling, and omit the details here due to lack of space.

5.5 Experimental Results

We present empirical results with four real-world, benchmark datasets (three image datasets and one text dataset). We evaluate entity resolution (ER) accuracy, efficiency, and crowdsourcing cost of PERC under various initial conditions, and by asking the next crowdsourcing questions one at a time and also in batches, with different crowd errors. We compare PERC with four state-of-the-art crowdsourced ER approaches: transitive closure (TC) clustering [173, 175, 182], MinMax [77], PC-Pivot [176], DENSE and bDENSE [172].

5.5.1 Environment Setup

The detailed descriptions about the datasets used can be found in our full paper [184].

Evaluation Metrics.

• **Accuracy:** After a certain number of answers are collected by the next crowdsourcing method, we apply ER algorithm for clustering. To measure accuracy, we compare the output to the gold standard clustering. Specifically, we employ precision (p) and recall (r), defined as follows.

$$p = \frac{\text{\#record-pairs correctly reported as matching}}{\text{\#record-pairs reported as matching}} \quad (5.3)$$

$$r = \frac{\text{\#record-pairs correctly reported as matching}}{\text{\#matching record-pairs in gold clustering}} \quad (5.4)$$

Finally, we compute F1-measure, which is defined below.

$$F1\text{-measure} = 2pr/(p + r) \quad (5.5)$$

Following previous works [77, 172], we use F1-measure to demonstrate the accuracy of PERC and other competitors.

- **Crowdsourcing Cost:** The crowdsourcing cost denotes the total number of distinct record pairs being crowdsourced.
- **Efficiency:** We report the average computation time required to select the next batch of crowdsourcing questions. Clearly, this is the runtime of the algorithm to select the next batch of questions, and it excludes the crowdsourcing time (which would be similar across different algorithms, for a given batch size).

Compared Algorithms.

- **Transitive Closure (TC):** This method selects, uniformly at random, one of those record pairs for which the matching/ non-matching relationship cannot be inferred (via transitivity and anti-transitivity) from the existing edges. Following [173, 182], we consider majority voting while deciding on the next crowdsourcing results. TC-clustering never reaches $F1\text{-measure} \geq 0.75$ over our datasets, which is because this method does not consider conflicting evidences.
- **DENSE and bDENSE:** DENSE [172] considers only either the set of positive edges, or the set of negative edges between two disjoint record sets for calculating the strength of evidences, denoted as the ρ -ratio (for details, see Introduction). bDENSE is a batch version of DENSE, that selects multiple questions (having higher ρ -ratios) to ask next,

thereby allowing many crowd taskers to answer those questions in parallel. For ER, these methods apply SCC-clustering.

The authors in [172] considered majority voting to decide on the next crowdsourcing results. Moreover, they also assigned a fixed human accuracy of 0.9 (i.e., error rate = 0.1) on those answers.

- **MinMax:** For ER, [77] finds all positive and negative paths between a record pair. The weight of a path is determined by the smallest edge weight on that path. Finally, the algorithm selects the maximum-weight path to decide whether the records are matching or not. For next crowdsourcing, the authors proposed a hybrid strategy that prefers either a more certain matching pair, or a less certain non-matching pair. As (1) **MinMax** only considers the maximum-weight path (and ignores all other paths) between a record pair for both ER and next crowdsourcing, and (2) it does not consider the length of a path (intuitively, the error accumulated across a short-length path would be less than that through a longer path), the method can easily produce less effective results.

- **PC-Pivot:** For ER, [176] uses pivot-based correlation clustering. The clustering refinement phase consists of either splitting, where nodes are removed from clusters; or merging, where two clusters are combined. The problem is that every node is considered individually, and edges connecting to that node are used to calculate the respective benefit. Hence, the method may fail to capture the strength of the entire clustering, resulting in higher crowdsourcing cost in order to achieve a reasonable ER accuracy.

5.5.2 Next Crowdsourcing Results

We started with different numbers of initial edges and batch sizes based on the size of our datasets. In particular, we had about 2K, 0.2K, and 1.4K initial crowdsourced edges, respectively, for *AllSports*, *Gymnastics*, and *Landmarks* datasets. We set the batch size as 320, 40, and 120 questions, respectively, over these datasets.

5.5.2.1 Crowdsourcing Cost Improvement

In Figure 5.5, we show the number of next crowdsourcing questions required to reach a certain accuracy. We consider F1-measure of 0.75 and above, because higher accuracy results are more important in real-world applications. We do not show TC-clustering,

Figure 5.5: Cost improvement: # next crowdsourcing questions required to reach a certain accuracy (F1-measure)

Figure 5.6: Accuracy improvement (F1-measure) for next crowdsourcing

Figure 5.7: Efficiency improvement: Computation time required to select a batch of next crowdsourcing questions

because it did not achieve an accuracy over 0.75 in all our datasets. We find that the number of crowdsourcing questions required to obtain a higher accuracy is much less — often by a margin of 50% — for PERC, in comparison to bDENSE, PC-Pivot, and MinMax. For example, to achieve F1-measure of 0.95 in the *Gymnastics* dataset, PERC, bDENSE, PC-Pivot, and MinMax require 863, 1792, 3360, and 1866 next crowdsourcing questions, respectively. These results demonstrate the effectiveness of PERC in reducing crowdsourcing cost.

5.5.2.2 Accuracy Improvement

In Figure 5.6, we illustrate accuracy improvements of PERC over state-of-the-art approaches. We observed that the F1-measure of PERC increases at a higher rate and

quickly reaches around 0.95 with less number of next crowdsourcing questions, compared to other methods, in all our datasets. As an example, with about 12K next crowdsourcing questions over *AllSports*, the F1-measure of PERC is 0.95, whereas for bDENSE, MinMax, PC-Pivot, and TC-clustering, the F1-measures are 0.75, 0.79, 0.58, and 0.67, respectively. These results demonstrate the accuracy improvements of PERC next crowdsourcing algorithm.

5.5.2.3 Efficiency Improvement

We compare the average computation time required to select a batch of next crowdsourcing questions, which is computed as follows. We first measure the computation time to select all next crowdsourcing questions in order to reach a certain accuracy, e.g., F1-measure of 0.9 for PERC, bDENSE, PC-Pivot, and MinMax. One may recall that PERC next crowdsourcing might trigger an update of the previous maximum-likelihood clustering. We empirically found that these updates happen 20~25% of the times after next crowdsourcing, and the times consumed for such re-clusterings are also added in the total time required for PERC. Since TC-clustering does not achieve such a high accuracy, we instead consider the time required to obtain the highest possible accuracy via TC-clustering. Next, we divide this time by the total number of batches issued to crowd workers, and report this value as the average computation time to select one batch of next crowdsourcing questions for the respective methods.

Figure 5.7 shows that the average time for one batch selection is at least an order of magnitude faster in case of PERC, compared to that of bDENSE, PC-Pivot, and MinMax. We note that the Y-axis is logarithmic in these figures. For example, with the *Landmarks* dataset, the average time to select one batch (with 120 questions) using PERC is only 0.5 sec, whereas it requires about 15 sec, 12 sec, and 51 sec, respectively, to select a batch of same size using bDENSE, MinMax, and PC-Pivot. Thus, our empirical results illustrate that PERC is at least an order of magnitude faster compared to bDENSE, MinMax, and PC-Pivot, in terms of selecting the next crowdsourcing questions.

5.6 Demonstration

For demonstration, we will present a web app on a laptop (demonstration video: <https://www.youtube.com/watch?v=rQ7nu3b8zXY>). Figure 5.8 presents the user interface of

Figure 5.8: User interface of PERC

PERC. The top panel is designed for user to provide the input dataset, ground-truth answer file (which is optional, and used for the visualization of accuracy), correlation clustering algorithm (default as SCC), and the next crowdsourcing algorithm (default as PERC).

The monitor window in the middle shows the real-time uncertain graph generated by crowd answers (gray edges). Users can tick at the top right boxes to view the intermediate ER clustering results and the next questions selected (red edges). Each node here represents a record in the input dataset. Those nodes being grouped together are considered as the same entity by PERC. The red edges are the batch of questions to be asked next, while the grey ones represent the questions that were already asked (for clarity, we only draw the edges with probability over 0.5 here. The higher the probability on the edge, the darker and the thicker it will be). If the ground-truth is given, the nodes will be colored (each color stands for an entity based on the ground-truth), thereby providing a better visualization of accuracy. When the cursor is hovered for a second, the details for the corresponding item will be displayed in the left panel. For nodes, it would be the detailed content, e.g. an image. For edges, it would be the two records that it connects and the corresponding probability of being the same (as shown in Figure 5.8). Users are allowed to rollback or forward the process bar at the top of the monitor to dynamically view the crowdsourcing process in an interactive manner and to terminate the process

at any moment. Finally, the users can also employ our system to visualize the results of other existing crowdsourced ER techniques, e.g., MinMax [77] and bDense [172], for an effective comparison.

From the right panel, user can view the ER results, together with all the crowd answers, and the next crowdsourcing questions. Also, users may download them in files for further analysis.

5.7 Summary

We studied crowdsourced entity resolution together with erroneous crowd answers. Our solution PERC does not require any user-defined threshold values, and no apriori information about the error rate of crowd workers. We formulated the problem considering an uncertain graph model and using possible world semantics with edge independence. We employed the notion of reliability in uncertain graphs to identify the most effective next crowdsourcing questions. Based on detailed empirical results with four real-world datasets, PERC improves the accuracy by 15%, reduces the crowdsourcing cost by 50%, and also decreases the next question selection time by an order of magnitude compared to state-of-the-art approaches.

Chapter 6

Multi-Relation Graph Summarization

We study the novel problem of multi-relation graph summarization¹. Storing, managing, and mining very large graph data has become a hot topic in the database research community in recent years, fueled by the unprecedented growth rate of knowledge graphs, social networks, and Internet-of-Things². At the current rate of data volume increase, it is becoming highly impractical to store, manage, process, and visualize such big graphs. Graph summarization alleviate this problem, producing a concise graph representation (i.e., summary) that still can be meaningfully explored and queried. Graph summarization has shown to be beneficial in a wide range of applications, such as visualization, interactive and exploratory analysis, approximate query processing, reducing the on-disk storage footprint, and graph processing in modern hardware [96].

In real-world, entities are often correlated in multiple ways, either explicitly or implicitly. For instance, two persons could be identified as friends when they follow each other on social media, and they could be colleagues (or, even business rivals) working on similar projects, or potential collaborators in the future.

BioGRID (thebiogrid.org) describes seven different types of genetic interactions between genes in Homo Sapiens. STRING (string-db.org) models the complex protein-to-protein interactions with six types of correlations statistically learned from existing

¹This work is in submission to VLDB 2020 as Xiangyu Ke, Arijit Khan, and Francesco Bonchi, *Multi-relation Graph Summarization*.

²The Knowledge and Action Graph of Microsoft has 21 billion facts, 18 billion action links, and over five billion relationships between more than one billion people, places, and things [82]. Facebook has 800 million active users [110]. Graph-of-Things (GoT), which is a live knowledge graph system for Internet-of-Things, has been adding millions of records per hour, and roughly about 10 billion RDF triples per month [140].

protein databases, revealing that most protein interactions are associated with at least two types of correlations. Other applications where these multiple relations exist between entities include social and financial networks [67], urban and transportation systems [29], ecology research [159], and recommender systems [124]. Data in these domains are typically modelled as *multi-relation networks* (also known as *multi-layer*, *multiplex*, or *multi-dimensional networks*), that is, a graph where multiple edges of different types may exist between any pair of nodes [57]. This graph model has been attracting increasing research interest in graph analytics, such as in shortest path finding [190], core decomposition and densest subgraph discovery [67], node clustering [19], frequent subgraphs mining [185], and in social networks analysis [50].

In this paper we introduce and study, for the first time, the problem of multi-relation graph summarization. Before discussing the contributions of our work and how they collocate in the state of the art, we need to provide some background notions and formally define the problems we tackle.

6.1 Problem and Related Work

We consider graph summarization obtained by aggregating nodes into *supernodes*. In particular, we consider *lossless* summarization as introduced by Navlakha et al. [134], for simple undirected single-relation graphs. Given a graph $G = (V, E)$ a lossless summary $\mathcal{S} = \langle G_S, \mathcal{C} \rangle$ consists of a *graph summary* $G_S = (V_S, E_S)$ and a set of *edge corrections* $\mathcal{C} = \langle \mathcal{C}^+, \mathcal{C}^- \rangle$, where:

- $V_S = \{S_1, \dots, S_k\}$ is the set of *supernodes* inducing a partition of V , i.e., $\bigcup_{i=1}^k S_i = V$ and $\bigcap_{i=1}^k S_i = \emptyset$;
- $E_S \subseteq V_S \times V_S$ is a set of *superedges* between supernodes (possibly including self-loops);
- $\mathcal{C}^+ \subseteq E$ is the set of edges to be inserted to reconstruct G , while $\mathcal{C}^- \subseteq (V \times V) \setminus E$ is the set of edges to be deleted.

The summarization is lossless because given the summary $\mathcal{S}$ we can reconstruct the original graph *exactly*, by (1) “exploding” each superedge $(U, W) \in E_S$, i.e., creating

Figure 6.1: A simple graph and its summary.

an edge (u, w) for each pair of nodes $u, w \in U \times W$, (2) adding each edge in $\mathcal{C}^+$, and (3) removing the edges in $\mathcal{C}^-$. More formally: $E = \{(u, w) \mid u \in U, w \in W, (U, W) \in E_S\} \cup \mathcal{C}^+ \setminus \mathcal{C}^-$. An example of such lossless summary is provided in Figure 6.1.

The problem studied by Navlakha et al. [134] is to find the smallest possible summary of an input graph.

Problem 6.1 (Lossless-Sum [134]). *Given a simple, undirected, graph $G = (V, E)$, find its smallest lossless summary $\mathcal{S} = \langle G_S, \mathcal{C} \rangle$, i.e., the one that minimizes $|E_S| + |\mathcal{C}|$.*

The cost of storing the mapping from nodes in V to supernodes in V_S is disregarded in the objective function [134], since it remains constant across different summaries, i.e., $\mathcal{O}(|V|)$. The problem can be seen through the lenses of the *Minimum Description Length* (MDL) principle [147], which states that the best theory to infer from a set of data is the one which minimizes the sum of: (1) the size of the theory and (2) the size of the data when encoded through the theory. Here, the data is the input graph G , the theory is the summary graph represented by supernodes V_S and superedges E_S , and the correction list $\mathcal{C}$ is the encoding of the data with regards to the theory. For instance, in the example in Figure 6.1 the cost of the summary is 2 (1 superedge + 1 corrections), against a cost of 5 of the original graph (i.e, 5 edges).

As observed in [134], a summary is entirely defined by the partitioning of nodes into supernodes. In fact, once provided such partitioning, superedges can be deterministically decided by simply checking whether they induces advantages w.r.t. the summary cost or not. Finally, once decided the superedges, also the corrections are straightforwardly identified. Therefore Problem 6.1 is essentially a graph partitioning problem: given a single-relation graph G with m nodes and n edges, there are $\sum_{k=1}^n \binom{n}{k} = \mathcal{O}(2^n)$ possibilities to partition these n nodes into supernodes V_S . To solve it, Navlakha et al. [134] propose a simple greedy agglomerative heuristic with no quality guarantee. They also

study a lossy version of the problem. To the best of our knowledge, the computational complexity of Problem 6.1 has not been addressed in the literature [134, 156]. We keep the problem of investigating the computational complexity of Problem 6.1 open here, and consider it as an interesting future work.

A similar approach is followed by LeFevre and Terzi [112] who study summaries obtained by aggregating nodes into supernodes. However, they do not consider having a corrections table, instead they keep on each superedge the information about how many original edges it represents (but not which ones): this is clearly a lossy summarization. Their objective is then to find the summary minimizing the loss for a given number k of allowed supernodes (which implicitly controls the compression rate). The loss is represented by the *reconstruction error*, i.e., the difference between the original graph and the probabilistic graph that one can reconstruct from the summary. Similar to [134], LeFevre and Terzi [112] propose a simple greedy agglomerative heuristic with no quality guarantee.

Later Riondato et al. [146] studying the same problem of LeFevre and Terzi [112] expose a connection between graph summarization and geometric clustering problems (e.g., k -median), thanks to which they achieve the first polynomial-time approximation algorithm for the problem of [112]. In this paper we show that, following the same intuition, we can achieve the first polynomial-time approximation algorithm for the lossless summarization problem.

6.1.1 Multi-relation graph summarization

In this work we extend the notion of lossless summarization over multi-relation graphs. An undirected, multi-relation graph G is a triplet (V, E, R) , where V is a set of n nodes, R is a set of q relations, and $E \subseteq V \times V \times R$ is a set of m undirected edges. Therefore, in a multi-relation graph, each edge is a triplet: e.g., an edge between nodes u and v in relation $r \in R$ is represented by (u, v, r) . A summary $\mathcal{S} = \langle G_S, \mathcal{C} \rangle$ is defined as in the single-relation case, the only difference is that the correction edges in $\mathcal{C}$ are triplets and also superedges are now triplets, i.e., $E_S \subseteq V_S \times V_S \times R$.

Example 6.1. An example of a multi-relation graph and its summary is provided in Figure 6.2. The graph on the left hand-side is defined over 3 relations, contains 5 nodes and 16 edges.

Figure 6.2: A multi-relation graph and its summary.

Figure 6.3: The individual summary for each relation in Figure 6.2(a).

The summary on the right-hand side is obtained by grouping $\{a, c\}$ and $\{b, d\}$ as two supernodes and keeping $\{e\}$ as a supernode over all 3 relations. No correction is required here. The cost of such summary is thus 6 (given by 6 superedges + 0 correction), while the cost of the original graph was 16 (i.e., 16 edges).

Therefore, the problem studied in this paper is the following.

Problem 6.2 (Lossless-Sum-Multi). *Given an undirected, multi-relation graph $G = (V, E, R)$, find its smallest lossless summary $\mathcal{S} = \langle G_S, \mathcal{C} \rangle$, i.e., the one that minimizes $|E_s| + |\mathcal{C}|$.*

In some cases one might be interested in a summary with a predefined number k of supernodes. Moreover, one might solve Problem 6.2 above by using algorithms which takes in input the number of supernodes k , with a wrapper for selecting the optimal value of k . Therefore, we also study the following problem.

Problem 6.3 (k -Lossless-Sum-Multi). *Given an undirected, multi-relation graph $G = (V, E, R)$ and a positive integer k , find a summary graph $G_S = (V_S, E_S, R)$ having k supernodes (i.e., $|V_S| = k$), and a list of corrections $\mathcal{C}$, resulting in the smallest lossless summary $\mathcal{S} = \langle G_S, \mathcal{C} \rangle$, i.e., the one that minimizes $|E_s| + |\mathcal{C}|$.*

Table 6.1: Characterization of the most related papers.

Paper	Lossless	Multi-Relation	Approx. Guarantees
[134]	✓		
[112]			
[146]			✓
[16]			✓
[156]	✓		
this work	✓	✓	✓

6.2 Other related work

Graph summarization has been used for a wider range of problems related to static plain graphs [21, 22, 25, 43, 44, 86, 107, 145], static attributed graphs [37, 152, 166, 189], dynamic and stream graphs [5, 6, 12, 48, 62, 74, 95, 151, 153, 162, 170, 191]. We refer the reader to excellent surveys and tutorials [17, 35, 96, 108, 118, 121, 167] for a review on these problems. Regardless this wide literature, *no prior work has tackled graph summarization in the multi-relation graph setting*. Therefore, the rest of this section covers the literature about single-relation graph summarization.

In previous section, we already reviewed the related papers which constitute the background for our work. Table 6.1 collocates our contribution within the most important related work.

Aggregation-based graph summary. Notable techniques under this category are pattern mining and community based summarization [27, 148], supernode and edge-correction (thus lossless) [134, 156], supernode and reconstruction-error (thus lossy) [16, 112, 146]. Supernode based aggregation methods [16, 112, 134, 146, 156] are most similar to ours and are summarised in Table 6.1.

Web graph and social networks compression. Boldi and Vigna [22] showed that web graphs are compressible down to almost two bits per edge. Chierichetti et al. [43] used shingle ordering instead of lexicographical ordering of web pages, in order to tackle social networks. Finding an order of nodes, which captures the “regularity” of the network, is a challenging problem. Boldi et al. [21] introduced a layered label propagation algorithm for reordering very large graphs. Other interesting works include [25, 44, 86, 107, 145]. Most of them focus on reducing the number of bits needed to encode an edge, and none compute graph summaries: indeed, their compressed representation is not really a graph.

Attribute-based graph summary. Nodes and edges of many real-world graphs are annotated with attributes. Hence, there exist graph summarization works considering both topology and semantics of the node and edge attributes [103]. FUSE [152] is a functional summarization technique for protein interaction networks, and this helps comprehending high-level functional relationships in disease-related PPI networks such as Alzheimer’s disease network. SNAP [166] and OLAP [37] allow interactive summarization at various resolutions over heterogeneous networks. Topology and attribute-based summarization of a large collection of small graphs (e.g., chemical compounds) and its application in constructing data-driven visual graph query interfaces are discussed in [189]. These methods are not directly comparable to ours, since our summarization deals with the graph structure.

Application-oriented graph summary. These are graph summarization techniques for efficient query answering and pattern mining, such as reachability, shortest path, and pattern matching queries [61, 168, 192], keyword search [183], distributed graph computation [88], graph mining [34, 47, 110, 128], eigenvector centrality [112], neighborhood query [127], information cascade and influential node discovery [143, 144, 155].

6.3 Baseline Methods: Aggregate Summaries Across Relations

We first present several straightforward baseline methods for the Lossless-Sum-Multi problem, then demonstrate how they suffer from effectiveness issues, which will be instrumental in developing a more accurate and scalable solution in § 6.4.

6.3.1 Overview

Our baseline algorithms follow a two-step approach: **(1)** We explore the input graph for one relation at a time, and generate a lossless summary for each of them. **(2)** The summaries across relations are properly aggregated to obtain one uniform summary.

In the first step, our problem is same as the Lossless-Sum problem. Given a set of summaries $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_q\}$, each for a specific relation, our next target is to find a single summary, i.e., a partition into supernodes, that agrees as much as possible with the q individual summaries.

In § 6.3.2, we revisit some widely-used graph summarization techniques for the **Lossless-Sum** problem. The number of supernodes can be specified to be a user-defined number k . To the best of our knowledge, all existing algorithms [134, 156] for the **Lossless-Sum** problem are heuristic in nature, without any theoretical guarantee on the summary size. *Our contribution lies in designing a k -Median algorithm to formally prove an approximation guarantee for our objective.* In § 6.3.3, we formally define the summary aggregation problem, and provide several methods to aggregate the individual summaries across relations. Finally, in § 6.3.4 we discuss the potential problems for these two-step baseline algorithms.

6.3.2 Single-Relation Graph Summarization

We first design a k -Median algorithm for generating lossless summary, having provable guarantee for the **Lossless-Sum** problem with a user-specified number of supernodes k . Then, we revisit **Greedy** and **Randomized** algorithms from [134]. We finally discuss an advanced algorithm, **SWeG**, with similar idea in recent literature [156]. An example is provided in § 6.3.2.4 to demonstrate these algorithms. Note that in § 6.5, we discuss several empirical methods for finding a suitable k for the k -Median method, which helps it adapt to the general **Lossless-Sum** problem.

6.3.2.1 k -Median Algorithm

We perform k -Median clustering on the rows of the adjacency matrix A_G of the input graph G , and create k supernodes. In particular, the goal of k -median clustering would be to find a set of k centers $\mathbf{x} = \{x_1, x_2, \dots, x_k\}$ which minimize the k -median cost for the node set $V \subseteq \mathbb{R}^n$. The k -median cost is defined as $\sum_{v \in V} d(v, \mathbf{x})$, where $d(v, \mathbf{x}) = \min_{x \in \mathbf{x}} d(v, x)$. Here, $d(v, x)$ denotes the Euclidean distance between two points (i.e., nodes) $v, x \in \mathbb{R}^n$. The nodes are then grouped into k supernodes based on their nearest cluster center. Notice that the k -summary is a graph summary with exactly k supernodes. After that, we include superedges and correction list as discussed in § 6.1.

Performance guarantee on the summary size. We prove that the k -Median algorithm guarantees 16-approximation to the optimal solution for the **Lossless-Sum** problem with k supernodes. Here, we only consider the number of correction edges $|\mathcal{C}|$, since the

number of superedges $|E_s|$ can be bounded with $\mathcal{O}(k^2)$ for a k -summary. Our proof is built on top of a theoretical result by Riondato et al. [146], that establishes 8-approximation guarantee for a similar k -Median algorithm, with respect to the quality of a *lossy summarization*, known as the l_p reconstruction error as below.

For a lossy summarization [16, 112, 146], only the graph summary $G_S = (V_S, E_S)$ is created; no additional correction list is stored. The summary G_S is a complete graph in this case, including all self-loops, i.e., $E_S = V_S \times V_S$. Given a summary, the graph is approximately reconstructed by the expected adjacency matrix, $A_{G_S}^\uparrow$, which is an $(n \times n)$ matrix with $A_{G_S}^\uparrow(u, w) = \frac{|E_{UW}|}{|U||W|}$. Here, U, W are supernodes in V_S such that $u \in U$ and $w \in W$, and E_{UW} is the set of edges actually exist between U and W in the original graph G . The quality of the summary, called the l_p reconstruction error, is measured by a norm of difference between the input adjacency matrix A_G and the reconstructed adjacency matrix $A_{G_S}^\uparrow$.

The l_p reconstruction error (RE_p) of a summary G_S for a graph G is:

$$RE_p(G, G_S) = \sqrt[p]{\sum_{u=1}^{|V|} \sum_{w=1}^{|V|} (|A_G(u, w) - A_{G_S}^\uparrow(u, w)|)^p} \quad (6.1)$$

In this paper, we use $p = 1$, that is, the l_1 reconstruction error. By Riondato et al. [146], we have the following theorem.

Theorem 6.1. *Let $G_S^\#$ be the k -summary induced by the k -Median partitioning of the rows of A_G , and let G_S^+ be the optimal k -summary for G with respect to the l_1 -reconstruction error. The l_1 -reconstruction error of $G_S^\#$ is an 8-approximation to the best l_1 -reconstruction error. Formally,*

$$RE_1(G, G_S^\#) \leq 8 \cdot RE_1(G, G_S^+) \quad (6.2)$$

Next, to prove our ultimate result that the k -Median algorithm guarantees 16-approximation to the optimal solution for the Lossless-Sum problem with k supernodes, Lemma 6.1 will be useful.

Lemma 6.1. *Let G_S^+ be the optimal k -summary for G with respect to the l_1 -reconstruction error, and let G_S^* be the optimal k -summary for G with respect to the number of correction edges. The correction list size, $|\mathcal{C}_{G_S^+}|$ of G_S^+ is a 2-approximation to the best size of correction list, $|\mathcal{C}_{G_S^*}|$. Formally,*

$$|\mathcal{C}_{G_S^+}| \leq 2 \cdot |\mathcal{C}_{G_S^*}| \quad (6.3)$$

Proof. Let us denote by $\alpha_{UW} = \frac{|E_{UW}|}{|U||W|}$. From the definition of l_1 reconstruction error in Equation 6.1, we get:

$$\begin{aligned} RE_1(G, G_S) &= \sum_{u=1}^{|V|} \sum_{w=1}^{|V|} |A_G(u, w) - A_{G_S}^\dagger(u, w)| \\ &= 2 \cdot \sum_{(U, W) \in V_S \times V_S} |U||W| \alpha_{UW} (1 - \alpha_{UW}) \end{aligned} \quad (6.4)$$

The intuition behind this derivation is that there are $|U||W|$ cells corresponding to each supernode pair (U, W) in both the original adjacency matrix A_G and the reconstructed adjacency matrix $A_{G_S}^\dagger$. In $A_{G_S}^\dagger$, all such cells are filled with α_{UW} . In A_G , there are α_{UW} proportion of cells having value 1, and the rest $(1 - \alpha_{UW})$ proportion of cells having value 0. The subtraction results of $|A_G(u, w) - A_{G_S}^\dagger(u, w)|$ for the first group are all $(1 - \alpha_{UW})$, and those for the second group are all α_{UW} . Thus, we derive the second line in Equation 6.4.

Now in the context of lossless summary, we decide whether to keep a superedge between a pair of supernodes (U, W) by the edge density between them. If $\alpha_{UW} > 0.5$, maintaining a superedge can result in less storage overhead of $\mathcal{C}^-$, than that of $\mathcal{C}^+$ without this superedge. Suppose $\mathcal{C}$ is the correction list, then its size can be calculated as given below.

$$|\mathcal{C}| = \frac{1}{2} \sum_{(U, W) \in V_S \times V_S} |U||W| \begin{cases} (1 - \alpha_{UW}), & \text{if } \alpha_{UW} > 0.5 \\ \alpha_{UW}, & \text{otherwise} \end{cases} \quad (6.5)$$

From Equations 6.4 and 6.5, we have:

$$\frac{RE_1(G, G_S)}{|\mathcal{C}|} = \begin{cases} 4\alpha_{UV}, & \text{if } \alpha_{UV} > 0.5 \\ 4(1 - \alpha_{UV}), & \text{otherwise} \end{cases} \quad (6.6)$$

Clearly, $2 \leq \frac{RE_1(G, G_S)}{|\mathcal{C}|} \leq 4$, since $\alpha_{UV} \in [0, 1]$. In other words,

$$\frac{1}{4}RE_1(G, G_S) \leq |\mathcal{C}| \leq \frac{1}{2}RE_1(G, G_S) \quad (6.7)$$

Suppose G_S^+ be the optimal k -summary for G with respect to the l_1 -reconstruction error, and let G_S^* be the optimal k -summary for G with respect to the number of correction edges. Thus, we get:

$$\begin{aligned} |\mathcal{C}_{G_S^+}| &\leq \frac{1}{2}RE_1(G, G_S^+) \quad \triangleright \quad \text{by the r.h.s of Equation 6.7} \\ &\leq \frac{1}{2}RE_1(G, G_S^*) \quad \triangleright \quad \text{since } G_S^+ \text{ is optimal wrt } RE_1 \\ &\leq 2 \cdot |\mathcal{C}_{G_S^*}| \quad \triangleright \quad \text{by the l.h.s of Equation 6.7} \end{aligned} \quad (6.8)$$

This completes the proof. $\square$

Theorem 6.2. *Let $G_S^\#$ be the k -summary induced by the k -Median partitioning of the rows of A_G , and let G_S^* be the optimal k -summary for G with respect to the number of correction edges. The correction list size, $|\mathcal{C}_{G_S^\#}|$ of $G_S^\#$ is a 16-approximation to the best size of correction list, $|\mathcal{C}_{G_S^*}|$. I.e.,*

$$|\mathcal{C}_{G_S^\#}| \leq 16 \cdot |\mathcal{C}_{G_S^*}| \quad (6.9)$$

Proof. We denote by G_S^+ the optimal k -summary for G with respect to the l_1 -reconstruction error. Next, we derive the follows.

$$\begin{aligned} |\mathcal{C}_{G_S^\#}| &\leq \frac{1}{2}RE_1(G, G_S^\#) \quad \triangleright \quad \text{by the r.h.s of Equation 6.7} \\ &\leq 4 \cdot RE_1(G, G_S^+) \quad \triangleright \quad \text{by Theorem 6.1} \\ &\leq 4 \cdot RE_1(G, G_S^*) \quad \triangleright \quad \text{since } G_S^+ \text{ is optimal wrt } RE_1 \\ &\leq 16 \cdot |\mathcal{C}_{G_S^*}| \quad \triangleright \quad \text{by the l.h.s of Equation 6.7} \end{aligned} \quad (6.10)$$

Hence, the proof is completed. $\square$

6.3.2.2 Greedy Algorithm

The **Greedy** algorithm [134] is a heuristic, bottom-up approach. It first considers every node in the input graph as a supernode, and iteratively merges the best pair $\{u, w\}$ with the maximum cost reduction $\Delta C(u, w)$.

Between any pair of supernodes U and W , let Π_{UW} be the set of all possible pairs $\{a, b\}$, such that $a \in U$ and $b \in W$. $E_{UW} \subseteq \Pi_{UW}$ is defined as the set of edges actually present in the input graph G , i.e., $E_{UW} = \Pi_{UW} \cap E$. Therefore, the cost of the supernode pair (U, W) is calculated as:

$$C(U, W) = \min\{|\Pi_{UW}| - |E_{UW}| + 1, |E_{UW}|\} \quad (6.11)$$

The neighbor set $N(U)$ of U is defined to be the set of supernodes W that have such edge $\{a, b\} \in E_{UW}$. We define the cost of a supernode U as follows.

$$C(U) = \sum_{X \in N(U)} C(U, X) \quad (6.12)$$

The cost reduction due to merging supernodes U and W into a new supernode H can be calculated as:

$$\Delta C(U, W) = \frac{C(U) + C(W) - C(H)}{C(U) + C(W)} \quad (6.13)$$

Taking fraction instead of the absolute cost reduction in the above equation is to get rid of the bias towards nodes with higher degree.

Based on the above notion, the general workflow of **Greedy** algorithm is given below. **(1)** It computes the cost reduction $\Delta C(U, W)$ for all pairs of nodes in the input graph G which are 2-hops apart, and records those pairs which are positive. **(2)** The best pair of nodes $\{U, W\}$ with highest cost reduction is merged into a new node W . **(3)** Delete cost reduction records about U or W for all the nodes which is within 2-hops to U or W , and compute their cost reduction to the new supernode H . **(4)** Update the cost reduction between nodes in $N(H)$. **(5)** Repeat (2)-(4) until no positive cost reduction exist.

Notice that **Greedy** directly works with the **Lossless-Sum** problem. It can easily handle the additional input k for the number supernodes by **(a)** terminating earlier when the number of supernodes become k (even before satisfying condition (5)), or **(b)** force to merge the pairs with less “sacrifice” in summary quality (i.e., negative cost reduction) after (5), if a smaller k is required.

6.3.2.3 Randomized Algorithm and SWeG

Comparing with the basic **Greedy** algorithm, the **Randomized** approach reduces the high computation overhead, yet empirically achieving similar compression. In each step, it randomly selects an unexplored supernode U , and computes $\Delta C(U, W)$ with all its unexplored neighbors W in $N(U)$. If no positive reduction exists, U is marked explored, and the algorithm continues. Otherwise, U is merged with its best neighbor W (i.e., having the highest reduction) into a new node H . U and W are removed, and H is now unexplored. The algorithm stops when all nodes are explored.

The most recent algorithm, **SWeG** [156], is an advanced version of the **Randomized** algorithm with further efficiency improvement. It first divides the graph into smaller disjoint groups. Each node group contains supernodes with similar connectivity, based on the concept of the shingle of a node u as defined below:

$$f(u) = \min_{w \in N_u \text{ or } w=u} h(w) \quad (6.14)$$

where h is a bijective hash function $h : V \rightarrow \{1, \dots, |V|\}$, and N_v is the set of neighbors of node v in the input graph G . Two nodes have the same shingle with probability equal to the Jaccard similarity of their neighbor sets [26]. The shingle of a supernode U is extended to be:

$$F(U) = \min_{u \in U} (f(u)) \quad (6.15)$$

Two supernode $U \neq W \in V_S$ are more likely to have the same shingle if the nodes in U and those in W share similar connectivity. After the nodes are partitioned into groups, a similar procedure as **Randomized** is conducted within each group to merge supernodes. Moreover, this allows distributed implementation. The whole pipeline (grouping nodes, and merging nodes within each group) repeats T times, each time with a different, randomly generated hash function h . We note that h can be easily produced by shuffling $\{1, \dots, |V|\}$.

6.3.2.4 Example for Single-Relation Graph Summarization Methods

Figure 6.4 provides an example for single-relation graph summarization algorithms. Given the example graph on the left, **Greedy** produces the summary S_{GD} . At the beginning, the

Figure 6.4: Example for single-relation graph summary methods.

cost of each node equals to the number of edges incident to this node. Merging node a with c , or merging node b with d both lead to 50% cost reduction, regardless of order. Then we achieve an intermediate result, which is the same as S_{KM} . The cost of the superedge between supernodes ac and bd are counted twice from both side, which encourages **Greedy** to continue merging. *This demonstrates that Greedy tends to result in bigger-size supernodes.* Though, this will not cause any issue in this simple example, we later demonstrate in § 6.3.4 that this may create trouble in aggregated summary finding (i.e., the second step of baseline approaches). S_{KM} is the summary for k -Median when $k = 2$, which can be easily obtained with the provided adjacency matrix. If k is set to be 1, k -Median will return same result as S_{GD} . The result of the **Randomized** algorithm depends on the random order of node selection. In this example, however, the final result of **Randomized** would also be S_{GD} . The costs of these two summaries are same: S_{GD} has 1 superedge and 1 correction edge, while S_{KM} has 2 superedges.

6.3.2.5 Time Complexity Analysis

k -Median Algorithm. Using the techniques from [81], an $n \times \mathcal{O}(\log n)$ *sketched matrix* can be constructed in time $\mathcal{O}(n + m)$. Afterwards, the distance between any two rows of the original matrix A_G can be approximately computed (within a constant factor) in $\mathcal{O}(\log n)$ with high probability. Then, we use the result from [2] to select $\mathcal{O}(k)$ rows from the sketched matrix in time $\mathcal{O}(nk \log n)$. Finally, we use the techniques in [131] to obtain an $\mathcal{O}(1)$ -approximation to the clustering of the $\mathcal{O}(k)$ rows of the sketched matrix. The optimal clustering of the rows of the sketched matrix is also an $\mathcal{O}(1)$ -approximation to the optimal clustering of the rows of the original adjacency matrix. The total running time is $\mathcal{O}(m + nk \log n)$.

Greedy Algorithm. In the first step of cost reduction computation, there exist roughly $n \cdot \text{Hop}_2(W)$ node pairs. Here, $\text{Hop}_2(W)$ denotes the number of nodes that are at

most two-hops away from W , similarly we define $Hop_3(W)$, $Hop_4(W)$, etc. We store the pairs in a heap, the height of this heap is $\log(n \cdot Hop_2(W))$. When the merging in step (2) and (3) happens, we look at each neighbor X of the new supernode W , and recompute the costs of all pairs containing X . The number of such pairs is about $Hop_2(X)$. Then, summing for all the neighbors of W makes it become roughly $Hop_3(W)$. Further, recomputing the cost requires iterating through all the edges of both the nodes, which adds another hop and make it $Hop_4(W)$. Updating the heap takes $Hop_3(W) \cdot \log(n \cdot Hop_2(W))$. Thus, the total complexity is $\mathcal{O}(Hop_4(W) + Hop_3(X) \cdot \log(n \cdot Hop_2(W)))$. Let d_{av} ($d_{av} \leq n$) be the average degree for each node, then it becomes: $\mathcal{O}(d_{av}^3(d_{av} + \log n + \log d_{av}))$.

Randomized Algorithm. In each merging step, we compute the cost reduction of all the $Hop_2(V)$ pairs containing the selected node V , which takes roughly $Hop_3(V)$ time. Thus, the complexity of the **Randomized** algorithm becomes $\mathcal{O}(d_{av}^3)$.

SWeG Algorithm. The dividing step takes $\mathcal{O}(|E|)$ time, since the computational bottleneck is to compute the shingles of all supernodes, which accesses every edge twice. The cost of the merging step is same as that of the **Randomized** algorithm: $\mathcal{O}(d_{av}^3)$, but here d_{av} may be smaller since it operates on a smaller graph. If we only allow sequential implementation and repeat T times, it becomes $\mathcal{O}(pTd_{av}^3)$, where p is the number of disjoint groups.

The above complexity analysis is for summary construction of a single-relation graph. Clearly, m therein denotes the number of edges considering a single relation. In our baseline approaches, since we apply these algorithms relation-by-relation (and follow up with a summary aggregation algorithm), total complexity of baseline is: q (i.e., number of relations) times the complexity for summarizing one single relation, plus the complexity of summary aggregation algorithm.

6.3.2.6 Horizontal Comparison among k -Median, Greedy, Randomized, and SWeG

Among these four algorithms, only k -Median is an approximation algorithm, with provable guarantee for the **Lossless-Sum** problem with k supernodes. The others are heuristic ones, without any approximation guarantee on the summary size of **Lossless-Sum**.

The two randomized algorithms sacrifice quality (i.e., not picking the best pair to merge in each iteration), to achieve better efficiency. **SWeG** first divides the nodes into disjoint groups, and apply the **Randomized** algorithm in each group. Ideally, the nodes in each groups would have similar connectivity, and merging them results in higher cost reduction. Thus, the quality of **Greedy** and **SWeG** summaries tend to be better than that of the **Randomized** algorithm. However, we still do not know which one is better between **Greedy** and **SWeG** algorithms, unless one evaluates them experimentally. Similarly, it is hard to analytically compare the approximate solution of **k-Median** with three heuristic algorithms. Thus, we empirically evaluate them in § 6.6.

As derived in the time complexity analysis, the **k-Median** algorithm is more sensitive to the number of nodes in the input graph, while **Greedy**, **Randomized**, and **SWeG** are more sensitive to the average degree of each node. Although, **SWeG** tends to have smaller d_{av} (average degree for each node), and this reduces its complexity, this may not lead to efficiency improvement if we only allow sequential implementation, since additional terms of iterations T and number of groups p are included. We also experimentally compare the efficiency of these four algorithms in § 6.6.

6.3.3 Summary Aggregation

The second phase of our baseline approach is to aggregate summaries across individual relations to obtain one uniform summary for the entire multi-relation graph. We formally define this problem following the well-known **clustering aggregation** problem [72]. Both summary and clustering techniques partition the nodes (objects) into groups, and the aggregation operation further aims at finding a new uniform partition with minimum total disagreements to the current partitions.

A summary $\mathcal{S}$ corresponds to a partition of n nodes. The partition size k can either be given as an input (§ 6.3.2.1), or can be automatically decided by the algorithm (§ 6.3.2.2). For any pair of nodes (u, v) , the indicator function $I_{\mathcal{S}}(u, v)$ returns 1 if and only if u and v are located in the same supernode under summary $\mathcal{S}$ (0 otherwise). Then, we have $D_{u,v}(\mathcal{S}_i, \mathcal{S}_j)$ ($i, j \in [1, q]$) to characterize if two summaries $\mathcal{S}_i$ and $\mathcal{S}_j$ disagree to each other on the partitioning of u and v . Formally,

$$D_{u,v}(\mathcal{S}_i, \mathcal{S}_j) = \begin{cases} 1, & \text{if } I_{\mathcal{S}_i}(u, v) \neq I_{\mathcal{S}_j}(u, v) \\ 0, & \text{otherwise} \end{cases} \quad (6.16)$$

The total disagreement between summaries $\mathcal{S}_i$ and $\mathcal{S}_j$ is:

$$D_V(\mathcal{S}_i, \mathcal{S}_j) = \sum_{(u,v) \in V \times V} D_{u,v}(\mathcal{S}_i, \mathcal{S}_j) \quad (6.17)$$

In general, this metric counts the number of node pairs on which the two summaries disagree to each other. Therefore, the *summary aggregation* problem is given as follows.

Problem 6.4 (Sum-Agg). *Given a set of q summaries (i.e., node partitions) $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_q\}$ on a set of nodes V , compute a new summary $\mathcal{S}$ that minimizes the total disagreements with all the given summaries, that is, it minimizes $\sum_{i=1}^q D_V(\mathcal{S}, \mathcal{S}_i)$.*

We now show that the **Sum-Agg** problem can be reduced to the well-studied correlation clustering problem [13], thus standard procedures for solving correlation clustering can be employed.

For any pair of nodes $(u, v) \in V \times V$, we define the distance between them as $\mathcal{D}(u, v) = \frac{1}{q} \cdot |\{i : 1 \leq i \leq q \text{ and } I_{\mathcal{S}_i}(u, v) = 0\}|$, which means the fraction of summaries that assign the pair (u, v) into different supernodes. The correlation clustering objective is to find a node partitioning $\mathcal{P}$ that minimizes $\mathcal{D}_{\mathcal{P}} = \sum_{(u,v), I_{\mathcal{P}}(u,v)=1} \mathcal{D}(u, v) + \sum_{(u,v), I_{\mathcal{P}}(u,v)=0} (1 - \mathcal{D}(u, v))$. If the solution $\mathcal{P}$ places u, v in the same group, it will disagree with $q \cdot \mathcal{D}(u, v)$ of the original partitionings due to individual relations. In contrast, it will disagree with $q(1 - \mathcal{D}(u, v))$ remaining partitionings if the solution keeps u, v separate. Thus, for any partitioning $\mathcal{P}$, we have $q \cdot \mathcal{D}_{\mathcal{P}} = \sum_{i=1}^q D_V(\mathcal{P}, \mathcal{S}_i)$, which is our objective of **Sum-Agg** (i.e., Problem 6.4). Due to this reduction, the following algorithms for solving correlation clustering can be employed. Some of them also come with provable approximation guarantees.

The BEST Algorithm. The most simple algorithm, **BEST**, is to find one of the input summaries, $\mathcal{S}_i$ that minimizes the total number of disagreements to other inputs. It can be computed in time $\mathcal{O}(q^2n)$, where q is the number of input summaries (also the number of relations), and n the number of nodes in the input graph. Though simple, it yields a solution with an approximation ratio at most $2 \cdot (1 - 1/q)$ [72].

The Balls Algorithm. The **Balls** algorithm [33] first sorts the nodes in an increasing order of the total distance to all other nodes. Recall that for any pair of nodes $(u, v) \in V \times V$, we define the distance between them as $\mathcal{D}(u, v) = \frac{1}{q} \cdot |\{i : 1 \leq i \leq q \text{ and } I_{\mathcal{S}_i}(u, v) = 0\}|$,

Figure 6.5: Empirical comparison among five summary aggregation algorithms over *DBLP_10*. that is, the fraction of summaries that assign the pair (u, v) into different supernodes. The algorithm is defined with an input parameter α . The intuition of the algorithm is to find a set of nodes that are close to each other, and far from other nodes. In order to find a good cluster, we take all nodes that are close (within a ball) to a node u . The triangle inequality guarantees that if two nodes are close to u , then they are also relatively close to each other. Once such a cluster is found, we remove it from the graph, and proceed with the rest of the nodes. The **Balls** algorithm consumes $\mathcal{O}(n^2)$ running time, while ensuring $\max\{\frac{1-\alpha}{\alpha}, \frac{1+2\alpha}{1-2\alpha}, \frac{2-2\alpha}{1-2\alpha}\}$ approximation guarantee [72]. The additional time cost for pre-computing distances between all pairs of nodes is $\mathcal{O}(qn^2)$.

The Agglomerative Algorithm. It follows the standard bottom-up procedure. The distance $\mathcal{D}(u, v)$ between nodes u, v is the same as in the **Balls** algorithm. The **agglomerative** algorithm first creates a singleton group for each node. Then, in each step, it picks the pair of groups A and B with the smallest average distance, which is defined as the average distance of all pair of nodes (a, b) , such that $a \in A$ and $b \in B$. If it is less than $1/2$, the two selected groups are merged into a single one. The algorithm terminates when no merging is possible. The time complexity of this algorithm is $\mathcal{O}(n^2 \log n)$.

The Furthest Algorithm. The **Furthest** algorithm is a top-down method. At the beginning, all nodes are placed in one group. Then, the pair of nodes which are furthest apart are found, and kept in two different groups. All remaining nodes are assigned to the center that incurs the least cost. In the following steps, each time a new center is found to be furthest from the current centers, and the node assignment changes accordingly. The procedure continue until the new solution induces a worse cost. Suppose that at the end we have k centers, then total running time will be $\mathcal{O}(k^2 n)$.

The LocalSearch Algorithm. The LocalSearch algorithm starts with some partition of nodes. It can be a randomly generated one, or the result of any aforementioned algorithm. The algorithm goes through each node, and decides to keep it still, move it to other group, or make it a singleton, by comparing various costs. An efficient way to compute the cost of assigning a node v to the cluster C_i is as follows:

$$\text{cost}(v, C_i) = M(v, C_i) + \sum_{j \neq i} (|C_j| - M(v, C_j)) \quad (6.18)$$

Here, $M(v, C_i) = \sum_{u \in C_i} \mathcal{D}(u, v)$. The cost of assigning a node v as a singleton is $\sum_j (|C_j| - M(v, C_j))$. The running time of LocalSearch is $\mathcal{O}(Tn^2)$, where T is the number of iterations before no better move can be found.

Empirically comparing summary aggregation algorithms. Figure 6.5 presents the experimental comparison of the five aforementioned summary aggregation algorithms. All individual summaries are generated by the Greedy algorithm. The *compactness* is evaluated with the relative size of the summary to that of the original graph. For more details on the dataset and experimental setup, we refer to § 6.6. We observe that the compactness of Agglomerative, Furthest, and LocalSearch are comparable, and they all outperform BEST and Ball algorithms. Considering also the running time, the Furthest algorithm is slightly better than others. Thus, in our following experimental sections, the Furthest is selected as the default summary aggregation method.

6.3.4 Limitation of Baselines Methods

In this section, we discuss the shortcoming of the baseline two-step approaches. Figures 6.6 provides an example graph for the following demonstration.

Example 6.2. Figures 6.7 and 6.8 demonstrate the whole pipeline of baseline methods presented in this section. For the graph induced by considering each relation, we draw the optimal summary based on Greedy (Figure 6.7). Next, with the help of the distance values calculated and presented in Figure 6.8(a), all the summary aggregation algorithms return the Summary S_1 , which has 3 superedges and 6 correction edges. However, there exist another summary S_2 , which can represent the input graph with 6 superedges and without any correction edge. This example demonstrates that the two-step baselines may return a lower-quality solution.

Figure 6.6: The example graph with 3 relations.

Figure 6.7: The individual summary for each relation.

Figure 6.8: (a) The graph with distances between nodes for summary aggregation computation. We omit node E here since its distance to any other node is 1. (b) The aggregated summary of the individual summaries. All the summary aggregation algorithms produce the same result in this example. (c) The potentially best summary S_2 .

Based on this example, we now discuss the limitation of baseline two-steps methods: Minimizing the disagreement between the final summary and the individual summaries (obtained via summarizing on individual relations) does not directly optimize our ultimate objective in **Lossless-Sum-Multi** and **k-Lossless-Sum-Multi** problems. Recall that the summary aggregation operation tends to minimize the disagreement between the final summary and the input set of summaries. Therefore, if the input has a majority population of low-quality summaries, or inherently meaningless summaries for certain relations, the aggregation will be forced to agree to those useless summaries.

Notice that in Figure 6.7, each of the individually optimal summaries S_{r_1} and S_{r_2} has two supernodes, as opposed to three supernodes in the individually optimal summary S_{r_3} . This is because **Greedy** tends to produce larger-size supernodes. Next, when we aggregate S_{r_1} , S_{r_2} , and S_{r_3} as in Figure 6.8(a)-(b), the resultant summary S_1 also has two supernodes, exactly same as the two supernodes in S_{r_1} and S_{r_2} . Unfortunately, S_2 that has three supernodes (same as in S_{r_3}) is the optimal summary (Figure 6.8(c)). This

demonstrates that: (1) if the input has a majority population of low-quality summaries, the aggregation will be forced to agree to those useless summaries. (2) Individual summaries having larger-size supernodes, as produced by Greedy, may cause trouble in the aggregation procedure.

6.4 Proposed Algorithms: Holistic Summary of All Relations

In this section, we present the methods that we ultimately design for an effective and efficient solution to the **Lossless-Sum-Multi** and k -**Lossless-Sum-Multi** problems. Unlike the two-step baseline approaches, we develop holistic algorithms that summarize the graph in a lossless manner and considering all relations. We shall discuss holistic versions of k -**Median** (§ 6.4.1), **Greedy** (§ 6.4.2), and **Randomized** (§ 6.4.3) algorithms, referred to as k -**Median**⁺, **Greedy**⁺, and **Randomized**⁺, respectively. We further establish that k -**Median**⁺ guarantees 16-approximation to the optimal solution for the k -**Lossless-Sum-Multi** problem.

6.4.1 Holistic k -Median: k -Median⁺

For a multi-relation graph, an adjacency matrix exists for each relation. Our intuitive idea is whether it is possible to *aggregate the adjacency matrices due to different relations into one aggregated matrix, and then cluster the rows of this aggregated matrix*. Potential aggregation operations may include **Sum**, **Or**, and **Concatenate**. Let $\{A_{G_1}, A_{G_2}, \dots, A_{G_q}\}$ be a set of q ($n \times n$) matrices, the **Sum** operation produces an aggregated matrix by $A_G(i, j) = \sum_{1 \leq x \leq q} A_{G_x}(i, j)$, the **Or** operation produces an aggregated matrix by $A_G(i, j) = \bigcap_{1 \leq x \leq q} A_{G_x}(i, j)$, and the **Concatenate** operation produces an aggregated matrix $A_G = (A_{G_1} | A_{G_2} | \dots | A_{G_q})$ by concatenating the rows. Among them, we show below that concatenation permits k -**Median**⁺ in achieving 16-approximation guarantee to the optimal summary size.

Theorem 6.3. *Let A_G be the concatenated matrix of the adjacency matrices for individual relations, i.e., $A_G = (A_{G_1} | A_{G_2} | \dots | A_{G_q})$, where A_{G_i} is the adjacency matrix for relation i , $i \in (1, q)$. Let $G_S^\#$ be the k -summary induced by the k -Median partitioning of the*

rows of A_G , and let G_S^* be the optimal k -summary for G with respect to the number of correction edges. The correction list size, $|\mathcal{C}_{G_S^\#}|$ of $G_S^\#$ is a 16-approximation to the best size of correction list, $|\mathcal{C}_{G_S^*}|$. Formally,

$$|\mathcal{C}_{G_S^\#}| \leq 16 \cdot |\mathcal{C}_{G_S^*}| \quad (6.19)$$

To prove Theorem 6.3, we need to justify that Lemma 6.1 and Theorem 6.1 (mentioned in § 6.3.2.1 in the context of k -Median algorithm over single-relation graphs) still hold with our concatenated matrix.

First, we verify the correctness of Lemma 6.1 in this case. To extend our objective of counting the number of correction edges from single-relation to multi-relation, we only need to include an additional **sum** operation to Equation 6.5 as follows:

$$|\mathcal{C}| = \frac{1}{2} \sum_{r \in R} \sum_{U, W \in V_S \times V_S} |U||W| \begin{cases} (1 - \alpha_{UW,r}), & \text{if } \alpha_{UW,r} > 0.5 \\ \alpha_{UW,r}, & \text{otherwise} \end{cases} \quad (6.20)$$

Since we have a holistic summary corresponding to same set of supernodes across relations, the edge density between supernodes U and W for each relation r , denoted by $\alpha_{UW,r}$, can be calculated in the same way. Hence, the l_1 reconstruction error (Equation 6.4) can be modified as:

$$RE_1(G, G_S) = 2 \sum_{r \in R} \sum_{(U, W) \in V_S \times V_S} |U||W| \alpha_{UW,r} (1 - \alpha_{UW,r}) \quad (6.21)$$

Based on the above two equations, one can prove an equivalent lemma of our earlier Lemma 6.1, for the multi-relation case. That is, $|\mathcal{C}_{G_S^+}| \leq 2 \cdot |\mathcal{C}_{G_S^*}|$. Here, G_S^+ is the optimal k -summary for G with respect to the l_1 -reconstruction error, and $|\mathcal{C}_{G_S^+}|$ is the correction list size for G_S^+ . Clearly, **Sum** and **Or** operations violate Equation 6.20.

Next, we rewrite the l_1 reconstruction error in its original form (equivalent of Equation 6.1).

$$\begin{aligned} RE_1(G, G_S) &= \sum_{r \in R} \sum_{u=1}^{|V|} \sum_{w=1}^{|V|} \left| A_{G_r}(u, w) - A_{G_{S,r}}^\dagger(u, w) \right| \\ &= \sum_{r \in R} \left\| A_{G_r} - A_{G_{S,r}}^\dagger \right\| \end{aligned} \quad (6.22)$$

where $A_{G_{S,r}}^\uparrow(u, w)$ is the edge density between the supernodes U and W for relation r , such that, $u \in U, w \in W$.

To prove Theorem 6.1, Riondato et al. [146] defined an orthogonal smoothing projection P for a partitioning $\mathcal{P}$ of n nodes (i.e., rows) in the adjacency matrix A_G . Since in our holistic summary, all relations would share the same partitioning $\mathcal{P}$, the same projection P can be applied to each adjacency matrix A_{G_r} . The l_1 -reconstruction matrix for each relation can be computed by $A_{G_{S,r}}^\uparrow(u, w) = PA_{G_r}P$. By definition, $A_{G_r}P$ is the k -Means matrix. And we define the k -Median matrix of relation r as below:

$$B_{G_r}(u, w) = \text{median}(\{A(x, y) | \{x, y\} \in \Pi_{UW}, u \in U, w \in W\}) \quad (6.23)$$

Lemma 1 and Lemma 4 in [146] provide the inequalities bridging the l_1 -reconstruction error $\|A - PAP\|$, k -means cost $\|A - AP\|$ and the k -median cost $\|A - B\|$. Thus, the proof of Theorem 6.1 for our multi-relation case is as follows.

$$\begin{aligned} RE_1(G, G_S^\#) &= \sum_{r \in R} \|A_{G_r} - A_{G_{S,r}}^\uparrow\| \quad \triangleright \text{by Equation 6.22} \\ &= \sum_{r \in R} \|A_{G_r} - P_{G_S^\#} A_{G_r} P_{G_S^\#}\| \\ &\leq 2 \cdot \sum_{r \in R} \|A_{G_r} - A_{G_r} P_{G_S^\#}\| \quad \triangleright \text{Lemma 4, [146]} \\ &\leq 4 \cdot \sum_{r \in R} \|A_{G_r} - B_{G_S^\#}\| \quad \triangleright \text{Lemma 1, [146]} \\ &\leq 4 \cdot \sum_{r \in R} \|A_{G_r} - B_{G_S^+}\| \quad \triangleright G_S^\# \text{ is best for } k\text{-Median} \\ &\leq 4 \cdot \sum_{r \in R} \|A_{G_r} - A_{G_r} P_{G_S^+}\| \quad \triangleright \text{Lemma 1, [146]} \\ &\leq 8 \cdot \sum_{r \in R} \|A_{G_r} - P_{G_S^+} A_{G_r} P_{G_S^+}\| \quad \triangleright \text{Lemma 4, [146]} \\ &= 8 \cdot RE_1(G, G_S^+) \end{aligned} \quad (6.24)$$

Since both Theorem 6.1 and Lemma 6.1 hold for the multi-relation case, one can prove the correctness of 16-approximation result due to k -Median on the concatenated adjacency matrix. Therefore, Theorem 6.3 follows.

6.4.2 Holistic Greedy: Greedy⁺

The Greedy algorithm can be generalized to a holistic algorithm, Greedy⁺ for the Lossless-Sum-Multi problem without changing the workflow. We keep the same definition of Π_{UW} for any pair of supernodes U and W . Obviously, Π_{UW} is the same no matter for which relation. Next, we extend $A_{UW,r} \subseteq \Pi_{UW}$ to be the set of edges actually present in the original graph G for relation r , i.e., $A_{UW,r} = \Pi_{UW} \cap E_r$. Similarly, the cost of a supernode pair (U, W) for relation r can be calculated as:

$$C(U, W, r) = \min\{|\Pi_{UW}| - |A_{UW,r}| + 1, |A_{UW,r}|\} \quad (6.25)$$

Moreover, the cost of maintaining a supernode U would add up across relations as follows:

$$C(U) = \sum_{r \in R} \sum_{X \in N_r(U)} C(U, X, r) \quad (6.26)$$

Example 6.3. We use the example graph in Figure 6.6 for the demonstration of Greedy⁺. At the beginning, the cost of each node equals to the number of edges incident to this node, for all relations. Thus, node a and c will be selected for merging in the first round, resulting in $\frac{7+7-(3+3+1)}{7+7} = 0.5$ cost reduction. Similarly, the nodes b and d will be merged in the second round. The cost again reduces by $\frac{4+4-(3+1)}{4+4} = 0.5$. Then, it can be easily verified that no further merging can result in positive cost reduction. Thus, it returns the optimal summary, S_2 in Figure 6.8, for this example.

6.4.3 Holistic Randomized: Randomized⁺

Randomized⁺ follows the same modification in cost computation as the holistic Greedy⁺ algorithm. The computation step does not change when comparing with the original Randomized algorithm. For SWeG algorithm, it is non-trivial to extend the node set dividing step (Equations 18, 19) to multi-relation case. We, therefore, leave it as a future direction.

6.5 Finding Optimal Number of Supernodes

When the user does not explicitly provide the target number of supernodes, a strategy for determining the optimal value of k allows the solution of k -Lossless-Sum-Multi to adapt to the more general problem, Lossless-Sum-Multi.

The Elbow Method. The elbow method [165] directly verifies the cost of the k -median clustering when varying k . Within each clusters, it computes the median distance between each node to its center, and takes the sum of all k clusters, which is known as the *within-cluster sum-of-squares* (WSS). Then, we plot the WSS against the cluster number k . Ideally, the curve will have an “elbow” shape, that is, starting from some specific value of k , the WSS decreases with a much smaller rate than before. Such k is chosen as the optimal one.

The Silhouette Method. The average silhouette [90] of the data is another useful criterion for assessing the natural number of supernodes. The silhouette of a node v is defined as:

$$sil(v) = \frac{b(v) - a(v)}{\max\{a(v), b(v)\}} \quad (6.27)$$

where $a(v)$ is the median distance between node v and all other nodes in the same supernode, and $b(v)$ is the smallest median distance of node i to all nodes in any other supernode. Notice that $-1 \leq sil(v) \leq 1$. We calculate the average silhouette for all nodes in a range of k , and select the k with the minimum silhouette.

The X-Median Method. Unlike the Elbow and the Silhouette methods, the X-Median method does not verify a range of k one by one, and chooses the experimental best one. Rather, it works in a top-down, hierarchical clustering manner. In each step, it partitions each current cluster into two new clusters, and adopts the partition if the *Bayesian Information Criterion* (BIC) is optimized. $BIC(C|X) = \log(X|C) - \frac{k}{2} \log n$. When no further partition can improve the BIC score, the number of current k is returned as the optimal. Similarly, all other hierarchical clustering methods *with a criteria of automatical termination* can be applied.

In our implementation, we adopt a combination of aforementioned methods: At first, the Greedy or Randomized algorithm can suggest a k' for k -summary, since they are able to automatically terminate. Then, we vary the k in a range containing k' as the center, e.g., $[k' - 1000, k' + 1000]$, and apply the Elbow method to determine an optimal number k^* for supernodes. In practice, we can replace the clustering cost with our exact summary cost, and the curve becomes “valley” shape instead of “elbow” shape. The experimental results can be found in § 6.6.5.

Table 6.2: Properties of datasets.

Dataset	#Nodes	#Edges	#Relations	Domain
<i>Homo</i>	18 222	153 923	7	genetic
<i>DBLP_10</i>	513 630	1 015 809	10	co-authorship
<i>DBLP_230</i>	704 266	4 727 290	230	co-authorship
<i>Amazon</i>	410 237	8 132 507	4	co-purchasing
<i>Twitter</i>	6 294 565	11 063 034	500	social

6.6 Experimental Results

We conduct experiments to demonstrate the effectiveness (compactness), efficiency, and scalability of our algorithms (averaged over 10 runs). The code is implemented in C++, and is executed on a single core, 40GB, 2.40GHz Xeon server.

6.6.1 Experimental Setup

Datasets. We use five real-world, multi-relation networks, whose main characteristics are listed in Table 6.2. (1) ***Homo*** (<https://comunelab.fbk.eu/data.php>). *Homo* network describes different types of genetic interactions between genes in Homo Sapiens. Nodes are genes and edges denote their interactions. Seven different relations exist, which are: direct interaction, physical association, suppressive genetic interaction defined by inequality, association, colocalization, additive genetic interaction defined by inequality, and synthetic genetic interaction defined by inequality. (2) ***DBLP*** (<http://dblp.uni-trier.de/xml>). *DBLP* is a well known collaboration network. Each node is an author and edges denote their co-author relations. We use two versions of *DBLP* dataset. *DBLP_10* [68] has 10 relations for 10 data mining topics, and *DBLP_230* [91] has 230 relations for 230 keywords extracted from paper titles, based on both their frequency and how well they can represent various sub-areas of computer science. (3) ***Amazon*** (<https://snap.stanford.edu/data/>). *Amazon* is a co-purchasing temporal network with four snapshots between March and June 2003, each as a relation. Nodes are products and edges are their co-purchasing relationships. (4) ***Twitter*** (<https://snap.stanford.edu/data/>). In *Twitter*, nodes are users and edges are their re-tweet relationships. The relations stand for 500 high frequency hashtags.

Competing Algorithms. We compare baseline two-step methods with the proposed holistic algorithms. The two-step algorithms include (1) **Greedy** [134]. It treats each

node in the original graph as a supernode at the beginning, and merges the best pair of supernodes with maximal positive cost reduction in each iteration, until no merging is possible. **(2) Randomized** [134]. Similar to **Greedy**, it starts by keeping every node in the input graph as a supernode, and randomly pick a supernode u to merge with another supernode w with maximal positive cost reduction to u . If no positive cost reduction exists between u to any other supernode, u will be marked as “explored”. The algorithm terminates when all current supernodes are explored. **(3) k -Median** (proposed approximation algorithm for the **Lossless-Sum** problem). It performs k -Median clustering on the rows of the adjacency matrix A_G of the input graph G , and create k supernodes. **(4) SWeG** [156]. It first divides the graph into smaller disjoint groups (§ 6.3.2.3). Then a similar procedure as **Randomized** is conducted within each group to merge supernodes. All methods are coupled with the **Furthest** algorithm [72] for summary aggregation since it works best empirically (§ 6.3.3), and are denoted as **GD**, **RD**, **KM**, and **SWeG**, respectively. For fairness, we only allow sequential execution of the **SWeG** algorithm.

The holistic version of **Greedy**, **Randomized**, and k -Median are represented as **GD+** (**Greedy**⁺), **RD+** (**Randomized**⁺), and **KM+** (k -**Median**⁺), respectively. These holistic algorithms are our proposed algorithms for lossless, multi-relation graph summarization.

In addition, we also consider a method, denoted as **All**, that stores individually optimal summaries (following the k -Median algorithm) for all relations.

The optimal number of supernodes are automatically decided by **Greedy**⁺ and **Randomized**⁺, and they are reported in Table 6.3. For k -**Median**⁺, we decide the optimal number of supernodes empirically, in particular via the combination of **X-Median** and **Elbow** methods. The detailed implementation is discussed in § 6.5, and the corresponding experimental result is shown in § 6.6.5.

Evaluation Metrics. **(1) Relative Size.** We measure the compactness of the obtained summaries by the metric: *relative size*, which is defined as $\frac{|E_S|+|C|}{|E|}$. The numerator is our objective function (Problem 3.4), i.e., cost of the summary, and the denominator is constant for a given graph, i.e., graph size. Smaller relative size means better compactness for the summary. **(2) Running Time.** The total running time for generating a graph summary is reported. For two-step baselines, it contains both the time of producing summaries for all individual relations and the time of aggregating them. **(2) Storage Cost.**

Figure 6.9: Trade-off between relative size and running time. Sub-figure (f) presents breakup on #superedges and #correction-edges for summaries.

Table 6.3: #supernodes, and (#superedges, #correction-edges) in obtained summaries via our holistic methods.

Graph	GD+	RD+	KM+
<i>Homo</i>	440, (4K,18K)	449, (4K,19K)	452, (4K,19K)
<i>DBLP_10</i>	8K, (100K,376K)	8K, (107K,401K)	8K, (99K,381K)
<i>DBLP_230</i>	9K, (373K,1.5M)	9K, (392K,1.6M)	9K, (393K,1.5M)
<i>Amazon</i>	6K, (956K,3.6M)	6K, (1.0M,3.7M)	6K, (960K,3.6M)
<i>Twitter</i>	28K, (1.9M,6.2M)	30K, (1.9M,6.3M)	28K, (1.9M,6.2M)

We report the exact storage cost for original graphs and the corresponding summaries. The mappings from the node set V to the supernode set V_S are included in summaries.

6.6.2 Comparison with Two-Step Baselines

In Figure 6.9, the Y-axis presents the relative size. Meanwhile, the X-axis reports the running time for summary construction. We want the summary construction as fast as possible. Thus, *better solution shall be closer to the origin point* in these plots.

The solid markers in Figure 6.9 stand for our proposed holistic algorithms, while the hollow markers represent the two-step baselines. We observe that the solid markers are closer to the origin point on all datasets, which confirms the superiority of our proposed holistic algorithms, based on both summary compactness and its construction efficiency.

Compactness. For the same kind of algorithm, *the proposed holistic version can return about 5% more compact summary than the corresponding two-step baseline algorithm*, e.g., the k -Median⁺ summary always has at least 3% smaller relative size than the k -Median summary, over all datasets.

Within each algorithm group, we find that **(1)** Greedy returns the most compact summaries, while Randomized produces the worst results; **(2)** the compactness of the summaries by SWeG and k -Median are in the middle range. Usually, they are comparable to the summaries by Greedy. These two observations hold both within the two-step baselines and within the holistic algorithm groups.

The objective function of our summary has two components, the number of superedges $|E_S|$ and the number of correction edges $|\mathcal{C}|$. Figure 6.9(f) demonstrates that *the cost of correction list dominates the total summary cost*, which is about 80% of the total cost.

Efficiency. For efficiency, we have the following observations. **(1)** The holistic algorithms (with solid markers) tend to consume less running time than the two-step baselines. Recalling the complexity analysis in § 6.3.2.5, the two-step baselines repeat the summary generation for every relation. This multiplies the time complexity of single summary computation by q , where q is the total number of relations. For example, the total time cost for producing q single-relation summaries will be $\mathcal{O}((m + nk \log n) \cdot q)$ for k -Median. An additional time for summary aggregation is also required. In contrast, for k -Median⁺, the time complexity is only $\mathcal{O}(m' + nk \log nq)$. The first term m' is the total number of edges across all relations, which is similar to mq . However, the second term is much smaller by taking $\log q$. This explains why our proposed holistic algorithms are faster than the baseline ones. **(2)** Usually, k -Median⁺ is the fastest among the proposed holistic methods, and Greedy⁺ is the slowest. However, k -Median⁺ takes more time than Greedy⁺ in *Twitter*. This dataset has the largest number of nodes, but the graph is sparse. Recalling the complexity analysis, k -Median⁺ is more sensitive to the number of nodes, while Greedy⁺ is more sensitive to the average degree of each node. On the other hand, the efficiency results on a much denser graph, e.g., *Amazon*, show that Greedy⁺ has higher running time.

Table 6.4: Actual storage cost (MB) for graphs and summaries. The actual storage cost for summaries includes the supernode mapping(s). We also report summary storage percentage w.r.t. original graph storage.

Graph	Original	GD+	KM+	All
<i>Homo</i>	1.7	0.31 (18%)	0.32 (19%)	0.58 (34%)
<i>DBLP_10</i>	15.3	8.3 (54%)	8.5 (56%)	11.4 (75%)
<i>DBLP_230</i>	62.5	27.0 (43%)	28.1 (45%)	110.6 (177%)
<i>Amazon</i>	116.9	68.8 (59%)	69.2 (59%)	68.2 (58%)
<i>Twitter</i>	567.1	420.0 (74%)	441.2 (78%)	1146.3 (202%)

Table 6.5: Actual storage cost (MB) for graphs and summaries with further compression: (super)edges between the same set of (super)nodes over multiple relations are stored as $\langle node_i, node_j, relation_x, relation_y, \dots, relation_z \rangle$. The actual storage cost for summaries includes the supernode mapping(s).

Graph	Original	GD+	KM+	All
<i>Homo</i>	1.3	0.17 (13%)	0.18 (14%)	0.54 (42%)
<i>DBLP_10</i>	13.1	5.6 (43%)	5.8 (44%)	10.1 (77%)
<i>DBLP_230</i>	50.6	16.8 (33%)	17.1 (34%)	106.0 (209%)
<i>Amazon</i>	94.9	50.8 (54%)	52.4 (55%)	60.6 (64%)
<i>Twitter</i>	419.1	303.1 (72%)	317.5 (76%)	1058.6 (253%)

6.6.3 Exact Storage Cost

We empirically study the exact storage cost of the original graphs and the summaries in the memory. Here, we further compare our uniform summary for all relations, against maintaining all the single-relation optimal summaries, denoted as **All**. For the exact storage, in addition to our objective of superedges and correction list, we need to store the mapping M from the original node set V to the supernode set V_S . In the single-relation graph summary, and in our uniform summary for all relations, this mapping M simply has the same size as the cardinality of the original node set V . However, there exist q mappings for the **All** method, since the supernode partitioning can be different across relations. As shown in Table 6.4, in practice, the **All** summaries require more storage overhead than our uniform **Greedy**⁺ or **k-Median**⁺ summary. When the relation number is large, e.g., on *DBLP_230* and *Twitter*, the **All** summaries have even larger storage cost than the original graphs. Table 6.6 provides more insight through decomposing the storage cost.

In the storage format so far (reported in Table 6.4), each entry is represented as $\langle node_i, node_j, relation \rangle$, following our definition in § 6.1. However, the storage cost can further reduce when there exist (super)edges between the same set of (super)nodes

Table 6.6: Breakup of the actual storage cost (MB) for summaries with further compression (shown in Table 6.5). E_S denotes superedges, C denotes correction edges, and M denotes the node mapping.

Graph	GD+			All		
	$ E_S $	$ C $	$ M $	$ E_S $	$ C $	$ M $
<i>Homo</i>	0.02	0.10	0.05	0.05	0.15	0.34
<i>DBLP_10</i>	1.2	3.9	0.6	2.0	3.7	4.4
<i>DBLP_230</i>	3.5	12.5	0.8	7.9	12.9	85.2
<i>Amazon</i>	11.0	38.4	1.4	15.2	36.6	8.8
<i>Twitter</i>	58.4	241.0	3.6	73.3	242.7	742.5

(a) Varying graph size

(b) Varying number of relations

Figure 6.10: Scalability analysis.

over multiple relations. In such scenarios, a simple way to further reduce the exact storage cost is to keep them as $\langle node_i, node_j, relation_x, relation_y, \dots, relation_z \rangle$. As shown in Table 6.5, our uniform summaries benefit more from this type of storage format, compared to the All summaries.

6.6.4 Scalability Analysis

We test scalability of our methods on the largest *Twitter* graph.

Graph Size. The *Twitter* dataset has about 6.3 million nodes, we select 1M, 2M, 3M, 4M, 5M, and 6M nodes uniformly at random to generate six graphs with all relations, and apply our algorithms on them. Figure 6.10(a) demonstrates that *all our proposed holistic algorithms scale linearly in graph size*.

Number of Relations. *Twitter* dataset has 500 relations. We randomly choose 100, 200, 300, 400, and all 500 relations, and apply the algorithms on the full graphs of selected relations. Figure 6.10(b) shows that *the running times of all our algorithms increase linearly with more relations*. In the time complexity of k -Median⁺: $\mathcal{O}(m' + nk \log nq)$, m'

Figure 6.11: Sensitivity analysis with #supernodes k , *Amazon*.

scales about linearly in the number of relations, while the second term $nk \log nq$ keeps nearly the same with increasing q . Overall, the experiments confirm good scalability of our holistic methods.

6.6.5 Varying Number of Supernodes

We further investigate the sensitivity with respect to the number of supernodes, k , for the k -Median⁺ algorithm. In Figures 6.11(a), we find that the relative size plots are valley-shaped. Too small or too large k both result in less compact summaries. The vertically dashed lines denote the number of supernodes found with the Greedy⁺ algorithm, which are located in the optimal ranges of k for k -Median⁺. This demonstrates that starting from the supernode number k' of Greedy⁺, we can vary the k in a small range centered at k' to find the optimal one for k -Median⁺. The running time is about linear in the number of supernodes, k . Due to the space limit, we omit the results on other datasets since they are in general similar to that on *Amazon* dataset.

6.6.6 Case Study on DBLP: Visualization

We visualize the summary of *DBLP_230* dataset, and present some interesting case studies in Figure 6.12. In the first case shown in Figure 6.12(a), we find that all researchers in each supernode actively collaborate with others on the topics such as “data stream” and “approximation”. They have both self-loops and links to each other with such topics. However, a few different topics also appear within each group. For example, *Graham Cormode* also has quite a few “private”-related (i.e., privacy) joint works with *Divesh Srivastava*. Another interesting finding is that both the left-side and the top supernodes contain researchers from database community, since they have more publications

Figure 6.12: Case study on *DBLP*.

in *SIGMOD* and *VLDB*, while the right-side supernode consists of theory community researchers, who publishes more in *STOC*, *SODA*, etc.

The second case in Figure 6.12(b) presents a group of people with close collaboration. One can verify that they are all from same geographical location, i.e., *Hong Kong* in this example. This is a frequent pattern in *DBLP*.

Figure 6.12(c) first shows that very senior researchers tend to be kept as a single node, since they work on quite diverse topics and with many researchers. Here, *Jiawei Han* and his ex-students, *Jian Pei* and *Xifeng Yan*, are all kept as a singleton supernode, and *Jiawei Han* has different collaboration topics with them. His recently graduated student, *Xiang Ren* (in 2018), and some other current students are grouped together, since they collaborate frequently with each other on topics related to knowledge extraction. Such correlations across different relations and node set could not be immediately inferred if one keeps individual summaries for all relations separately.

Figure 6.13: Case study on *Autism*. The partitionings of the same set of 14 nodes are quite different for ASD brain networks and for healthy brain networks. The numbers on superedges denote the number of relations having such superedges.

6.6.7 Case Study on Brain Network Dataset: Classification and Visualization

For our second case study, we use the *Autism* dataset [51] containing 96 brain networks. 48 of them are collected from ASD (Autism spectrum disorder) patients, and the other 48 brain networks are of healthy people³. If we treat the 48 ASD brain networks as a multi-relation graph, GD+ returns a summary with 0.516 relative size (0.517 by KM+). Meanwhile, the relative size of GD+ multi-relation summary for the 48 healthy people is 0.526 (0.529 by KM+). However, if we randomly choose 24 networks from each group, and generate a multi-relation summary for them, the relative size will be 0.559 by GD+ (0.560 by KM+, both are the average value over 20 attempts). Clearly, it is harder to summarize the mixed group, which implies that *the networks of ASD patients have different structure to those of healthy people*, and inspires us to *utilize the multi-relation summaries for ASD patient detection*.

Classification. We randomly choose two brain networks, one for ASD patient and one for healthy people. A multi-relation summary S_1 is generated for the rest 47 ASD brain networks, and another multi-relation summary S_2 for the rest 47 healthy ones. Then, we alternatively apply S_1 and S_2 as summaries for the two selected networks, and suggest the label “ASD” to a network if S_1 results in lower cost than S_2 . We repeat the procedure 20 times. Within the 20 true ASD patients, 19 of them are correctly detected. For the 20 healthy people, 3 of them are wrongly labelled as “ASD”. It can be easily calculated that

³In the original dataset, there are 49 networks of ASD patients and 52 networks of healthy people. To maintain a balanced number of networks in each group, and for the ease of our comparison, we only keep the first 48 networks in each group here. All networks share the same set of 116 nodes. The average number of edges in each network is 1336.8 (1336.2 for ASD patients, and 1337.5 for healthy people).

our classification precision is 0.86, recall is 0.95, and F1-score is 0.90. This demonstrates the effectiveness of our multi-relation summaries in this classification task.

Visualization. Finally, we visualize some subgraphs of our multi-relation summaries for the two groups of brain networks. As shown in Figure 6.13, the partitionings of the same set of 14 nodes are quite different for ASD brain networks and for healthy brain networks. Although these supernodes are well-connected to each other, they can not be further merged since they have different superedges.

6.7 Summary

In this work, we introduced and investigated the novel problem of multi-relation graph summarization. To solve the problem, we first studied the baseline two-step approaches: first generate a summary for each relation, and then properly aggregate them. When revisiting the single relation graph summarization algorithms, we proved that the k -Median clustering on the rows of adjacency matrix provides a bounded solution to our objective. We further demonstrated and discussed the limitations of these baselines, and proposed holistic solutions to overcome them. Among them, the holistic k -Median⁺ is able to maintain the approximation guarantee over multi-relation graphs. Our experimental results and case studies validated the effectiveness and efficiency of our algorithms.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

With the advent of complex graphs in big data era, effective and efficient graph querying and mining techniques are emerging. My thesis focuses on analysing large graphs with uncertainty or multi-relations, or even both. Fundamental contributions are made in designing effective and scalable techniques. The thesis is rich in contents, ranging from survey and experimental analyses, algorithm design, to real world applications.

For fundamental s - t reliability query in uncertain graphs, we are the first to compare the state-of-the-art algorithms with each other. While introducing them, we correct some issues of past algorithms, and significantly improve their performance. We implement all methods on a same platform, design unified metrics, collect medium to large real datasets, and generate same workloads to conduct a fair, throughout experimental evaluation. We summarize their trade-offs and provide the guidelines for practitioners.

For reliability maximization problem, we design smart algorithms to select a small budget of edges to add in the input graph, in order to maximize the reliability of a targeted pair. The search space elimination strategy improves the efficiency, while the budget path selection methods improves the accuracy. We further generalize our algorithms to multiple sources and targets case, and study a restricted version of only considering most reliable paths. The experimental results and the rich real applications validate the effectiveness and the efficiency of our solutions.

For the problem of jointly finding the influential seed users and topics for targeted influence maximization, we utilize both uncertain graph and multi-relation graph model

to make the influence maximization problem more practical, and deeply characterize the hardness of proposed problem. To solve it, we refine both state-of-the-art seeds finding and tags finding algorithms together with indexing and batch-path selection methods, and thus improve their efficiency by 30% and accuracy up to 30%, respectively. Finally, we develop an iterative framework, together with smart initialization techniques, that alternatively optimizes the top- k seeds and the top- r tags, and we prove that the influence spread monotonically increases until convergence.

For PERC system, it is a successful attempt to apply uncertain graph model in real-world application. Here, we model the crowd errors as uncertain graph, and design a novel metric, called the “reliability” of a clustering, to assist the next crowdsourcing question selection for crowdsourcing entity resolution task. Based on detailed empirical results with four real-world datasets, PERC improves the accuracy by 15%, and reduces the crowdsourcing cost by 50%. We also develop a demo software for the PERC system.

For multi-relation graph summarization project, we formally characterize the proposed problem, and clearly distinguish it with correlation clustering and graph partitioning. To solve the problem, we first study the baseline two-step approaches: first generating a summary for each relation, and then properly aggregating them. When revisiting the single relation graph summarization algorithms, we prove that k -median clustering on the rows of adjacency matrix could provide a bounded solution to our objective. We further demonstrate and discuss the limitations of these baselines, and propose holistic solutions to overcome them. Among them, the holistic k -median⁺ is able to maintain the approximation guarantee over multi-relation graphs. Our experimental results and case studies validate the effectiveness and efficiency of our algorithms.

7.2 Future Directions

First, my research opens several potential directions for future works.

- The current s - t reliability algorithms can still be improved with advanced index structures or sampling strategies. Moreover, in a dynamic or streaming graph setting, the reliability between a given pair of nodes may vary over time, how to query it efficiently with provable accurate answer is not yet studied.

- In current reliability maximization problem, we assume a fixed individual budget on each new edge. However, in practice, we often have a total budget instead. Thus, a more challenging version of our problem can be maintaining a total budget, but varying number of new edges, each with different cost of budget.
- For influence maximization, one interesting direction for me shall be incorporating the algorithmic bias. We may further allow type categories on nodes, and target at getting rid of algorithmic bias, e.g., our algorithms shall always select balanced number of male and female seed users. Other direction for our original proposed problem may be designing a holistic method with a provable guarantee.
- The uncertain graph model can be applied to other crowdsourcing tasks. For example, the data annotation rules finding task in [186] aims to generate high-quality rules to largely reduce the labeling cost while preserving quality. A good annotation rule shall be able to cover large volume of records with high precision. The validation phrase of an annotation rule is very similar to crowdsourcing entity resolution task. Thus, the coverage of an annotation rule can be modeled as the size of the cluster of corresponding records, and the precision can be characterized using our “reliability” metric in PERC system.
- The successful of PERC system inspires me to apply the graph models to solve many other real-world problems in other domains, including health care, financial transactions, and so on.
- The current techniques for multi-relation graph summarization can return a bounded size answer only when supernode number k is given. How to get rid of k shall be investigated. Moreover, how to generalize it to dynamic/streaming setting is also an interesting topic.

In a long run, I would like to explore more research domains in big and complex graph area. My past works are mostly about the reliability, and the graph summarization is a first attempt to other topic. In the next step, I would like to investigate some other problems about graph querying and mining. Another direction can be incorporating more kinds of complexities over graphs, e.g., temporal information and geographic information. Finally, I would like to consider graph querying and mining as a service. How to make our techniques easy and safe to the end users is an interesting research direction.

Publications

1. **TKDE 2020**, Xiangyu Ke, Arijit Khan, Mohammad Al Hasan, and Rojin Rezvansangsari, “*Reliability Maximization in Uncertain Graphs*”.
2. **VLDB 2019**, Xiangyu Ke, Arijit Khan, and Leroy Lim Hong Quan, “*An In-Depth Comparison of s - t Reliability Algorithms over Uncertain Graphs*”.
3. **SIGMOD 2018**, Xiangyu Ke, Arijit Khan, and Gao Cong, “*Finding Seeds and Relevant Tags Jointly: For Targeted Influence Maximization in Social Networks*”.
4. **VLDB 2018 (Demo)**, Xiangyu Ke, Michelle Teo, Arijit Khan, and Vijaya Krishna Yalavarthi, “*A Demonstration of PERC: Probabilistic Entity Resolution With Crowd Errors*”.
5. **CIKM 2017**, Vijaya Krishna Yalavarthi, Xiangyu Ke, and Arijit Khan, “*Select Your Questions Wisely: For Entity Resolution with Crowd Errors*”.
6. **(Under Submission) VLDB 2020**, Xiangyu Ke, Arijit Khan, and Francesco Bonchi, “*Multi-Relation Graph Summarization*”.

References

- [1] E. Adar and C. Re, “Managing Uncertainty in Social Networks,” *IEEE Data Eng. Bull.*, 2007.
- [2] A. Aggarwal, A. Deshpande, and R. Kannan, “Adaptive Sampling for k-means Clustering,” *APPROX-RANDOM*, pp. 15–28, 2009.
- [3] C. Aggarwal, *Managing and Mining Uncertain Data*. Springer, 2009.
- [4] K. K. Aggarwal, K. B. Misra, and J. S. Gupta, “Reliability Evaluation A Comparative Study of Different Techniques,” *Micro. Rel.*, vol. 14, no. 1, 1975.
- [5] K. J. Ahn, S. Guha, and A. McGregor, “Graph Sketches: Sparsification, Spanners, and Subgraphs,” in *PODS*, 2012.
- [6] K. J. Ahn, S. Guha, and A. McGregor, “Analyzing Graph Structure via Linear Measurements,” in *SODA*, 2012.
- [7] R. Albert, H. Jeong, and A.-L. Barabasi, “Error and Attack Tolerance of Complex Networks,” *Nature*, vol. 406, pp. 378–382, 2000.
- [8] O. Alonso, D. E. Rose, and B. Stewart, “Crowdsourcing for Relevance Evaluation,” *SIGIR Forum*, vol. 42, no. 2, pp. 9–15, 2008.
- [9] C. Aslay, N. Barbieri, F. Bonchi, and R. A. Baeza-Yates, “Online Topic-aware Influence Maximization Queries,” in *EDBT*, 2014.
- [10] M. O. Ball, “Computational Complexity of Network Reliability Analysis: An Overview,” *IEEE Tran. on Reliability*, 1986.

REFERENCES

- [11] M. O. Ball and J. S. Provan, “Disjoint Products and Efficient Computation of Reliability,” *Oper. Res.*, 1988.
- [12] S. Bandyopadhyay, M. Mehta, D. Kuo, M.-K. Sung, R. Chuang, E. J. Jaehnig, B. Bodenmiller, K. Licon, W. Copeland, M. Shales, D. Fiedler, J. Dutkowski, A. Guénolé, H. van Attikum, K. M. Shokat, R. D. Kolodner, W.-K. Huh, R. Aebbersold, M.-C. Keogh, N. J. Krogan, and T. Ideker, “Rewiring of Genetic Networks in Response to DNA Damage,” *Science*, vol. 330, no. 6009, 2010.
- [13] N. Bansal, A. Blum, and S. Chawla, “Correlation Clustering,” *Machine Learning*, vol. 56, no. 1-3, pp. 89–113, 2004.
- [14] N. Barbieri, F. Bonchi, and G. Manco, “Topic-Aware Social Influence Propagation Models,” in *ICDM*, 2012.
- [15] N. Barbieri and F. Bonchi, “Influence Maximization with Viral Product Design,” in *SDM*, 2014.
- [16] M. A. Beg, M. Ahmad, A. Zaman, and I. Khan, “Scalable Approximation Algorithm for Graph Summarization,” in *PAKDD*, 2018.
- [17] M. Besta and T. Hoefler, “Survey and Taxonomy of Lossless Graph Compression and Space-Efficient Graph Representations,” *CoRR*, vol. abs/1806.01799, 2018.
- [18] A. Beygelzimer, G. Grinstein, R. Linsker, and I. Rish, “Improving Network Robustness by Edge Modification,” *Physica A: Stat. Mech. and its Appl.*, vol. 357, pp. 593–612, 2005.
- [19] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl, “Mining Coherent Subgraphs in Multi-layer Graphs with Edge Labels,” in *KDD*, 2012.
- [20] P. Boldi, F. Bonchi, A. Gionis, and T. Tassa, “Injecting Uncertainty in Graphs for Identity Obfuscation,” in *VLDB*, 2012.
- [21] P. Boldi, M. Rosa, M. Santini, and S. Vigna, “Layered Label Propagation: A Multiresolution Coordinate-Free Ordering for Compressing Social Networks,” in *WWW*, 2011.

REFERENCES

- [22] P. Boldi and S. Vigna, “The WebGraph framework I: Compression techniques,” in *WWW*, 2004.
- [23] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, “Maximizing Social Influence in Nearly Optimal Time,” in *SODA*, 2014.
- [24] T. B. Brecht and C. J. Colbourn, “Lower Bounds on Two-Terminal Network Reliability,” *Disc. App. Math.*, 1988.
- [25] N. R. Brisaboa, S. Ladra, and G. Navarro, “k2-Trees for Compact Web Graph Representation,” in *SPIRE*, 2009.
- [26] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, “Min-wise Independent Permutations,” *J. Comput. System Sci.*, vol. 60, pp. 630–659, 2000.
- [27] G. Buehrer and K. Chellapilla, “A Scalable Pattern Mining Approach to Web Graph Compression with Communities,” in *WSDM*, 2008.
- [28] D. Bulka and J. B. Dugan, “Network s-t Reliability Bounds using a 2-Dimensional Reliability Polynomial,” *IEEE Tran. Rel.*, 1994.
- [29] A. Cardillo, J. Gómez-Gardeñes, M. Zanin, M. Romance, D. Papo, F. del Pozo, and S. Boccaletti, “Emergence of Network Features from Multiplexity,” *CoRR*, vol. abs/1212.2153, 2012.
- [30] C. Chai, G. Li, J. Li, D. Deng, and J. Feng, “Cost-Effective Crowdsourced Entity Resolution: A Partial-Order Approach,” in *SIGMOD*, 2016.
- [31] H. Chan and L. Akoglu, “Optimizing Network Robustness by Edge Rewiring: a General Framework,” *Data Min. Knowl. Disc.*, vol. 30, pp. 1395–1425, 2016.
- [32] V. Chaoji, S. Ranu, R. Rastogi, and R. Bhatt, “Recommendations to Boost Content Spread in Social Networks.” in *WWW*, 2012.
- [33] M. Charikar, V. Guruswami, and A. Wirth, “Clustering with Qualitative Information,” *J. Comput. Syst. Sci.*, vol. 71, no. 3, pp. 360–383, 2005.

REFERENCES

- [34] C. Chen, C. X. Lin, M. Fredrikson, M. Christodorescu, X. Yan, and J. Han, “Mining Graph Patterns Efficiently via Randomized Summaries,” *PVLDB*, vol. 2, no. 1, pp. 742–753, 2009.
- [35] —, “Mining Large Information Networks by Graph Summarization,” in *Link Mining: Models, Algorithms, and Applications*, 2010, pp. 475–501.
- [36] C. Chen, H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos, “Eigen-Optimization on Large Graphs by Edge Manipulation,” *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 4, pp. 49:1–49:30, 2016.
- [37] C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu, “Graph OLAP: Towards Online Analytical Processing on Graphs,” in *ICDM*, 2008.
- [38] L. Chen, D. Lee, and T. Milo, “Data-driven Crowdsourcing: Management, Mining, and Applications,” in *ICDE*, 2015.
- [39] S. Chen, J. Fan, G. Li, J. Feng, K. L. Tan, and J. Tang, “Online Topic-aware Influence Maximization,” *PVLDB*, vol. Volume 8 Issue 6, pp. 666–677, 2015.
- [40] W. Chen, L. V. S. Lakshmanan, and C. Castillo, *Information and Influence Propagation in Social Networks*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.
- [41] W. Chen, C. Wang, and Y. Wang, “Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks,” in *KDD*, 2010.
- [42] Y. Cheng, Y. Yuan, L. Chen, and G. Wang, “The Reachability Query over Distributed Uncertain Graphs,” in *ICDCS*, 2015.
- [43] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan, “On Compressing Social Networks,” in *KDD*, 2009.
- [44] Y. Choi and W. Szpankowski, “Compression of Graphical Structures: Fundamental Limits, Algorithms, and Experiments,” *IEEE Trans. Information Theory*, vol. 58, no. 2, pp. 620–638, 2012.

REFERENCES

- [45] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye, “KATARA: Reliable Data Cleaning with Knowledge Bases and Crowdsourcing,” *PVLDB*, vol. 8, no. 12, pp. 1952–1955, 2015.
- [46] C. J. Colbourn, “Edge-packing of Graphs and Network Reliability,” *Discrete Mathematics*, vol. 72, pp. 49 – 61, 1988.
- [47] D. J. Cook and L. B. Holder, “Substructure Discovery Using Minimum Description Length and Background Knowledge,” *J. Artif. Intell. Res.*, vol. 1, pp. 231–255, 1994.
- [48] G. Cormode and S. Muthukrishnan, “Space efficient mining of multigraph streams,” in *PODS*, 2005.
- [49] F. Coro, G. D’Angelo, and Y. Velaj, “Recommending Links to Maximize the Influence in Social Networks.” in *IJCAI*, 2019.
- [50] M. Coscia, G. Rossetti, D. Pennacchioli, D. Ceccarelli, and F. Giannotti, “”You Know because I Know”: A Multidimensional Network Approach to Human Resources Problem,” in *ASONAM*, 2013.
- [51] C. Craddock, Y. Benhajali, C. Chu, F. Chouinard, A. Evans, A. Jakab, B. S. Khundrakpam, J. D. Lewis, Q. Li, M. Miham, C. Yan, and P. Bellec, “The Neuro Bureau Preprocessing Initiative: Open Sharing of Preprocessed Neuroimaging Data and Derivatives,” *Frontiers in Neuroinformatics*, vol. 41, 2013.
- [52] L. da F. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas, “Characterization of Complex Networks: a Survey of Measurements,” *Adv. Phys.*, vol. 56, pp. 167–242, 2007.
- [53] O. Dekel and O. Shamir, “Vox Populi: Collecting High-Quality Labels from a Crowd,” in *COLT*, 2009.
- [54] E. D. Demaine and M. Zadimoghaddam, “Minimizing the Diameter of a Network Using Shortcut Edge,” *SWAT. ser.Lec. Notes in Comput. Sci.*, pp. 420–431, 2010.

REFERENCES

- [55] G. Demartini, D. E. Difallah, and P. C.-Mauroux, “ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-scale Entity Linking,” in *WWW*, 2012.
- [56] L. Dhulipala, I. Kabiljo, B. Karrer, G. Ottaviano, S. Pupyrev, and A. Shalita, “Compressing Graphs and Indexes with Recursive Graph Bisection,” in *KDD*, 2016.
- [57] M. E. Dickison, M. Magnani, and L. Rossi, *Multilayer Social Networks*. Cambridge University Press, 2016.
- [58] A. Elmagarmid, I. F. Ilyas, M. Ouzzani, J.-A. Quiané-Ruiz, N. Tang, and S. Yin, “NADEEF/ER: Generic and Interactive Entity Resolution,” in *SIGMOD*, 2014.
- [59] M. Elsner and W. Schudy, “Bounding and Comparing Methods for Correlation Clustering Beyond ILP,” in *ILP*, 2009.
- [60] D. Eppstein, “Finding the k Shortest Paths,” *SIAM J. Comput.*, vol. 28, no. 2, pp. 652–673, 1998.
- [61] W. Fan, J. Li, X. Wang, and Y. Wu, “Query Preserving Graph Compression,” in *SIGMOD*, 2012.
- [62] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang, “Graph Distances in the Data-Stream Model,” *SIAM J. Comput.*, vol. 38, no. 5, pp. 1709–1727, 2008.
- [63] G. S. Fishman, “A Comparison of Four Monte Carlo Methods for Estimating the Probability of s-t Connectedness,” *IEEE Tran. Rel.*, 1986.
- [64] L. Fu, X. Fu, Z. Xu, Q. Peng, X. Wang, and S. Lu, “Determining Source-Destination Connectivity in Uncertain Networks: Modeling and Solutions,” *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3237–3252, 2017.
- [65] L. Fu, X. Wang, and P. R. Kumar, “Optimal Determination of Source-destination Connectivity in Random Graphs,” in *MobiHoc*, 2014.
- [66] X. Fu, Z. Xu, Q. Peng, L. Fu, and X. Wang, “Complexity vs. optimality: Unraveling Source-Destination Connection in Uncertain Graphs,” in *INFOCOM*, 2017.

REFERENCES

- [67] E. Galimberti, F. Bonchi, and F. Gullo, “Core Decomposition and Densest Subgraph in Multilayer Networks,” in *CIKM*, 2017.
- [68] E. Galimberti, F. Bonchi, and R. Gullo, “Core Decomposition and Densest Subgraph in Multilayer Networks,” in *CIKM*, 2017.
- [69] J. Galtier, A. Laugier, and P. Pons, “Algorithms to Evaluate the Reliability of a Network,” in *DRCN*, 2005.
- [70] L. Getoor and A. Machanavajjhala, “Entity Resolution for Big Data,” in *KDD*, 2013.
- [71] J. Ghosh, H. Q. Ngo, S. Yoon, and C. Qiao, “On a Routing Problem Within Probabilistic Graphs and its Application to Intermittently Connected Networks,” in *INFOCOM*, 2007.
- [72] A. Gionis, H. Mannila, and P. Tsaparas, “Clustering Aggregation,” *TKDD*, vol. 1, 2007.
- [73] R. Gomes, P. Welinder, A. Krause, and P. Perona, “Crowdclustering,” in *NIPS*, 2011.
- [74] X. Gou, L. Zou, C. Zhao, and T. Yang, “Fast and Accurate Graph Stream Summarization,” in *ICDE*, 2019.
- [75] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, “A Data-Based Approach to Social Influence Maximization,” *PVLDB*, vol. Volume 5 Issue 1, pp. 73–84, 2011.
- [76] A. Goyal, W. Lu, and L. V. S. Lakshmanan, “CELFF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks,” in *WWW*, 2011.
- [77] A. Gruenheid, D. Kossmann, S. Ramesh, and F. Widmer, “Crowdsourcing Entity Resolution,” Technical Report Nr. 785, Systems Group, Computer Sc., ETH Zurich, Tech. Rep., 2012.
- [78] G. Hardy, C. Lucet, and N. Limnios, “K-Terminal Network Reliability Measures With Binary Decision Diagrams,” *IEEE Tran. Rel.*, 2007.

REFERENCES

- [79] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller, “Framework for Evaluating Clustering Algorithms in Duplicate Detection,” in *VLDB*, 2009.
- [80] M. Hua and J. Pei, “Probabilistic Path Queries in Road Networks: Traffic Uncertainty aware Path Selection,” in *EDBT*, 2010.
- [81] P. Indyk, “Stable Distributions, Pseudorandom Generators, Embedding, and Data Stream Computation,” *J. ACM*, vol. 53, 2006.
- [82] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri, “Querying Knowledge Graphs by Example Entity Tuples,” *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2797–2811, 2015.
- [83] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy, “Pay-as-you-go User Feedback for Dataspace Systems,” in *SIGMOD*, 2008.
- [84] R. Jin, L. Liu, B. Ding, and H. Wang, “Distance-Constraint Reachability Computation in Uncertain Graphs,” in *VLDB*, 2011.
- [85] M. Joglekar, H. G.-Molina, and A. G. Parameswaran, “Evaluating the Crowd with Confidence,” in *KDD*, 2013.
- [86] U. Kang and C. Faloutsos, “Beyond ‘Caveman Communities’: Hubs and Spokes for Graph Compression and Mining,” in *ICDM*, 2011.
- [87] U. Kang, S. Papadimitriou, J. Sun, and H. Tong, “Centralities in Large Networks: Algorithms and Observations,” in *SDM*, 2011.
- [88] U. Kang, H. Tong, J. Sun, C. Lin, and C. Faloutsos, “GBASE: A Scalable and General Graph Management System,” in *KDD*, 2011.
- [89] D. R. Karger, S. Oh, and D. Shah, “Iterative Learning for Reliable Crowdsourcing Systems,” in *NIPS*, 2011.
- [90] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

REFERENCES

- [91] X. Ke, A. Khan, and G. Cong, “Finding Seeds and Relevant Tags Jointly: For Targeted Influence Maximization in Social Networks,” in *SIGMOD*, 2018.
- [92] X. Ke, A. Khan, M. A. Hasn, and R. Rezvan, “Reliability Maximization in Uncertain Graphs (Extended Version),” <https://arxiv.org/abs/1903.08587>, 2019.
- [93] X. Ke, A. Khan, and L. Lim, “An In-Depth Comparison of s-t Reliability Algorithms over Uncertain Graphs (Extended Version),” in <https://arxiv.org/pdf/1904.05300.pdf>, 2019.
- [94] D. Kempe, J. M. Kleinberg, and E. Tardos, “Maximizing the Spread of Influence through a Social Network,” in *KDD*, 2003.
- [95] A. Khan and C. C. Aggarwal, “Toward Query-Friendly Compression of Rapid Graph Streams,” *Social Netw. Analys. Mining*, vol. 7, no. 1, pp. 23:1–23:19, 2017.
- [96] A. Khan, S. S. Bhowmick, and F. Bonchi, “Summarizing static and dynamic big graphs,” *PVLDB*, vol. 10, no. 12, pp. 1981–1984, 2017.
- [97] A. Khan, F. Bonchi, A. Gionis, and F. Gullo, “Fast Reliability Search in Uncertain Graphs,” in *EDBT*, 2014.
- [98] A. Khan, F. Bonchi, F. Gullo, and A. Nufer, “Conditional Reliability in Uncertain Graphs,” *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 1, pp. 2078–2092, 2018.
- [99] A. Khan and L. Chen, “On Uncertain Graphs Modeling and Queries,” in *VLDB*, 2015.
- [100] A. Khan, F. Gullo, T. Wohler, and F. Bonchi, “Top-k Reliable Edge Colors in Uncertain Graphs,” in *CIKM*, 2015.
- [101] A. Khan, Y. Ye, and L. Chen, *On Uncertain Graphs*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.
- [102] A. Khan, B. Zehnder, and D. Kossmann, “Revenue Maximization by Viral Marketing: A Social Network Host’s Perspective,” in *ICDE*, 2016.

REFERENCES

- [103] K. U. Khan, W. Nawaz, and Y.-K. Lee, “Set-Based Unified Approach for Attributed Graph Summarization,” in *BDCLOUD*, 2014.
- [104] M. Kimura and K. Saito, “Tractable Models for Information Diffusion in Social Networks,” in *PKDD*, 2006.
- [105] A. Konak and A. Smith, “An Improved General Upper Bound for All-Terminal Network Reliability,” in *Industrial Eng. Res. Conf.*, 1998.
- [106] N. Koudas, S. Sarawagi, and D. Srivastava, “Record Linkage: Similarity Measures and Algorithms,” in *SIGMOD*, 2006.
- [107] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos, “Summarizing and Understanding Large Graphs,” *Statistical Analysis and Data Mining*, vol. 8, no. 3, pp. 183–202, 2015.
- [108] D. Koutra, J. Vreeken, and F. Bonchi, “Summarizing Graphs at Multiple Scales: New Trends,” in *ICDM*, 2018.
- [109] O. Kuchaiev, M. Rasajski, D. J. Higham, and N. Przulj, “Geometric De-noising of Protein-Protein Interaction Networks,” *PLOS Computational Biology*, vol. 5, no. 8, pp. 1–10, 08 2009.
- [110] K. A. Kumar and P. Efstathopoulos, “Utility-Driven Graph Summarization,” *PVLDB*, vol. 12, no. 4, pp. 335–347, 2018.
- [111] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila, “Finding Effectors in Social Networks,” in *KDD*, 2010.
- [112] K. LeFevre and E. Terzi, “GraSS: Graph Structure Summarization,” in *SDM*, 2010.
- [113] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, “Cost-effective Outbreak Detection in Networks,” in *KDD*, 2007.
- [114] R. Li, J. X. Yu, R. Mao, and T. Jin, “Recursive Stratified Sampling: A New Framework for Query Evaluation on Uncertain Graphs,” *IEEE TKDE*, vol. 28, no. 2, pp. 468–482, 2016.

REFERENCES

- [115] X. Li, R. Cheng, Y. Fang, J. Hu, and S. Maniu, “Scalable Evaluation of k-NN Queries on Large Uncertain Graphs,” in *EDBT*, 2018.
- [116] Y. Li, J. Fan, D. Zhang, and K.-L. Tan, “Discovering Your Selling Points: Personalized Social Influential Tags Exploration,” in *SIGMOD*, 2017.
- [117] Y. Li, D. Zhang, and K. Tan, “Real-time Targeted Influence Maximization for Online Advertisements,” *PVLDB*, vol. Volume 8 Issue 10, pp. 1070–1081, 2015.
- [118] S.-D. Lin, M.-Y. Yeh, and C.-T. Li, “Sampling and Summarization for Social Networks,” in *SDM*, 2013.
- [119] X. Lin, Y. Peng, B. Choi, and J. Xu, “Human-Powered Data Cleaning for Probabilistic Reachability Queries on Uncertain Graphs,” *IEEE TKDE*, vol. 29, no. 7, pp. 1452–1465, 2017.
- [120] X. Liu, M. Lu, B. C. Ooi, Y. Shen, S. Wu, and M. Zhang, “CDAS: A Crowdsourcing Data Analytics System,” *PVLDB*, vol. 5, no. 10, pp. 1040–1051, 2012.
- [121] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, “Graph Summarization Methods and Applications: A Survey,” *ACM Comput. Surv.*, vol. 51, no. 3, pp. 62:1–62:34, 2018.
- [122] W. Lu, W. Chen, and L. V. S. Lakshmanan, “From Competition to Complementarity: Comparative Influence Diffusion and Maximization,” *PVLDB*, vol. 9, no. 2, pp. 60–71, 2015.
- [123] S. Maniu, R. Cheng, and P. Senellart, “An Indexing Framework for Queries on Probabilistic Graphs,” *ACM Trans. Database Syst.*, vol. 42, no. 2, pp. 13:1–13:34, 2017.
- [124] M. Mao, J. Lu, G. Zhang, and J. Zhang, “Multirelational Social Recommendations via Multigraph Ranking,” *IEEE Trans. Cybernetics*, vol. 47, no. 12, pp. 4049–4061, 2017.
- [125] A. Marcus and A. G. Parameswaran, “Crowdsourced Data Management: Industry and Academic Perspectives,” *Foundations and Trends in Databases*, vol. 6, no. 1-2, pp. 1–161, 2015.

REFERENCES

- [126] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller, “Demonstration of Qurk: A Query Processor for Human Operators,” in *SIGMOD*, 2011.
- [127] H. Maserrat and J. Pei, “Neighbor Query Friendly Compression of Social Networks,” in *KDD*, 2010.
- [128] —, “Community Preserving Lossy Compression of Social Networks,” in *ICDM*, 2012.
- [129] T. Matisziw and A. Murray, “Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure,” *J. Comput. Oper. Res.*, vol. 36, pp. 16–26, 2009.
- [130] S. Medya, J. Vachery, S. Ranu, and A. Singh, “Noticeable Network Delay Minimization via Node Upgrades,” *PVLDB*, vol. Volume 11 Issue 9, pp. 988–1001, 2018.
- [131] R. R. Mettu and C. G. Plaxton, “The Online Median Problem,” *SIAM J. Comput.*, vol. 32, pp. 816–832, 2003.
- [132] A. Meyerson and B. Tagiku, “Minimizing Average Shortest Path Distances via Shortcut Edge Addition,” in *Approx., Rand., and Comb. Opt. Algo. and Tech.*, 2009, pp. 272–285.
- [133] A. Natanzon, R. Shamir, and R. Sharan, “Complexity Classification of Some Edge Modification Problems,” in *WG*, ser. Springer Lecture Notes in Computer Science, vol. 1665, 1999.
- [134] S. Navlakha, R. Rastogi, and N. Shrivastava, “Graph Summarization with Bounded Error,” in *SIGMOD*, 2008.
- [135] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An Analysis of Approximations for Maximizing Submodular Set Functions,” *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.
- [136] M. Papagelis, “Refining Social Graph Connectivity via Shortcut Edge Addition.” *TKDD*, vol. 10, 2015.

REFERENCES

- [137] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi, “The Pursuit of a Good Possible World: Extracting Representative Instances of Uncertain Graphs,” in *SIGMOD*, 2014.
- [138] N. Parotsdis, E. Pitoura, and P. Tsaparas, “Centrality-Aware Link Recommendations,” in *WSDM*, 2016.
- [139] J. Peng, B. Zhang, and S. Li, “Towards Uncertain Network Optimization,” *J. Uncertainty Analysis and Applications*, vol. 3, no. 1, 2015.
- [140] D. L. Phuoc, H. N. M. Quoc, H. N. Quoc, T. T. Nhat, and M. Hauswirth, “The Graph of Things: A Step towards the Live Knowledge Graph of Connected Things,” *J. Web Semant.*, vol. 37-38, pp. 25–35, 2016.
- [141] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios, “k-Nearest Neighbors in Uncertain Graphs,” *PVLDB*, 2010.
- [142] J. S. Provan and M. O. Ball, “Computing Network Reliability in Time Polynomial in the Number of Cuts,” *Operations Research*, 1984.
- [143] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. S. Subrahmanian, “Fast Influence-based Coarsening for Large Networks,” in *KDD*, 2014.
- [144] Q. Qu, S. Liu, C. S. Jensen, F. Zhu, and C. Faloutsos, “Interestingness-Driven Diffusion Process Summarization in Dynamic Networks,” in *ECML PKDD*, 2014.
- [145] S. Raghavan and H. Garcia-Molina, “Representing Web Graphs,” in *ICDE*, 2003.
- [146] M. Riondato, D. García-Soriano, and F. Bonchi, “Graph Summarization with Quality Guarantees,” in *ICDM*, 2014.
- [147] J. Rissanen, “Modelling by the Shortest Data Description,” *Automatica*, vol. 14, 1978.
- [148] R. A. Rossi and R. Zhou, “GraphZIP: A Clique-based Sparse Graph Compression Method,” *J. Big Data*, vol. 5, p. 10, 2018.

REFERENCES

- [149] G. Rubino, *Network Reliability Evaluation*, 1999.
- [150] K. Saito, R. Nakano, and M. Kimura, “Prediction of Information Diffusion Probabilities for Independent Cascade Model,” in *KES*, 2008.
- [151] B.-S. Seah, S. S. Bhowmick, and C. F. D. Jr, “DiffNet: Automatic Differential Functional Summarization of dE-MAP Networks,” *Methods*, vol. 63, no. 3, 2014.
- [152] B.-S. Seah, S. S. Bhowmick, C. F. D. Jr, and H. Yu, “FUSE: A Profit Maximization Approach for Functional Summarization of Biological Networks,” *BMC Bioinformatics*, vol. 13, 2012.
- [153] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos, “TimeCrunch: Interpretable Dynamic Graph Summarization,” in *KDD*, 2015.
- [154] A. Sharafat and O. Ma’rouzi, “All-Terminal Network Reliability Using Recursive Truncation Algorithm,” *IEEE Tran. on Rel.*, 2009.
- [155] L. Shi, S. Sun, Y. Xuan, Y. Su, H. Tong, S. Ma, and Y. Chen, “TOPIC: Toward Perfect Influence Graph Summarization,” in *ICDE*, 2016.
- [156] K. Shin, A. Ghoting, M. Kim, and H. Raghavan, “SWeG: Lossless and Lossy Summarization of Web-Scale Graphs,” in *WWW*, 2019.
- [157] C. Song, W. Hsu, and M. L. Lee, “Targeted Influence Maximization in Social Networks,” in *CIKM*, 2016.
- [158] S. Song, Z. Zou, and K. Liu, “Triangle-Based Representative Possible Worlds of Uncertain Graphs,” in *DASFAA*, 2016.
- [159] M. Stella, C. S. Andreazzi, S. Selakovic, A. Goudarzi, and A. Antonioni, “Parasite Spreading in Spatial Ecological Multiplex Networks,” *J. Complex Networks*, vol. 5, no. 3, pp. 486–511, 2017.
- [160] F. Sun and M. Shayman, “On pairwise connectivity of wireless multihop networks,” *Int. J. Network Security*, vol. 2, pp. 37–49, 2007.

REFERENCES

- [161] K. K. T. Akiba, C. Sommer, “Shortest-path queries for complex networks: Exploiting low tree-width outside the core,” in *EDBT*, 2012.
- [162] N. Tang, Q. Chen, and P. Mitra, “Graph Stream Summarization: From Big Bang to Big Crunch,” in *SIGMOD*, 2016.
- [163] Y. Tang, Y. Shi, and X. Xiao, “Influence Maximization in Near-Linear Time: A Martingale Approach,” in *SIGMOD*, 2015.
- [164] Y. Tang, X. Xiao, and Y. Shi, “Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency,” in *SIGMOD*, 2014.
- [165] R. L. Thorndike, “Who Belongs in the Family?” *Psychometrika*, vol. 18, pp. 267–276, 1953.
- [166] Y. Tian, R. A. Hankins, and J. M. Patel, “Efficient Aggregation for Graph Summarization,” in *SIGMOD*, 2008.
- [167] Y. Tian and J. M. Patel, “Interactive Graph Summarization,” in *Link Mining: Models, Algorithms, and Applications*, 2010, pp. 389–409.
- [168] H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka, “Compression of weighted graphs,” in *KDD*, 2011.
- [169] Y. Tong, C. C. Cao, C. J. Zhang, Y. Li, and L. Chen, “CrowdCleaner: Data Cleaning for Multi-Version Data on the Web via Crowdsourcing,” in *ICDE*, 2014.
- [170] I. Tsalouchidou, G. D. F. Morales, F. Bonchi, and R. Baeza-Yates, “Scalable dynamic graph summarization,” in *BigData*, 2016.
- [171] L. G. Valiant, “The Complexity of Enumeration and Reliability Problems,” *SIAM J. on Computing*, 1979.
- [172] V. Verroios and H. G.-Molina, “Entity Resolution with Crowd Errors,” in *ICDE*, 2015.
- [173] N. Vesdapunt, K. Bellare, and N. Dalvi, “Crowdsourcing Algorithms for Entity Resolution,” *PVLDB*, vol. 7, no. 12, pp. 1071–1082, 2014.

REFERENCES

- [174] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, “CrowdER: Crowdsourcing Entity Resolution,” *PVLDB*, vol. 5, no. 11, pp. 1483–1494, 2012.
- [175] J. Wang, G. Li, T. Kraska, M. J. Franklin, and J. Feng, “Leveraging Transitive Relations for Crowdsourced Joins,” in *SIGMOD*, 2013.
- [176] S. Wang, X. Xiao, and C.-H. Lee, “Crowd-Based Deduplication: An Adaptive Approach,” in *SIGMOD*, 2015.
- [177] Y. Wang, J. X. Cao, R. D. Wang, and X. X. Li, “Research on Uncertain Network Design Problem,” in *Advances in Transportation*, ser. Applied Mechanics and Materials, vol. 505. Trans Tech Publications, 2014, pp. 613–618.
- [178] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, “Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint,” in *SRDS*, 2003.
- [179] Z. Wang, E. Chen, Q. Liu, Y. Yang, Y. Ge, and B. Chang, “Maximizing the Coverage of Information Propagation in Social Networks,” in *IJCAI*, 2015.
- [180] F. L. Wauthier, N. Jojic, and M. I. Jordan, “Active Spectral Clustering via Iterative Uncertainty Reduction,” in *KDD*, 2012.
- [181] F. Wei, “TEDI: Efficient Shortest path query answering on graphs,” in *SIGMOD*, 2010.
- [182] S. E. Whang, P. Lofgren, and H. G.-Molina, “Question selection for crowd entity resolution,” *PVLDB*, vol. 6, no. 6, pp. 349–360, 2013.
- [183] Y. Wu, S. Yang, M. Srivatsa, A. Iyengar, and X. Yan, “Summarizing Answer Graphs Induced by Keyword Queries,” *PVLDB*, vol. 6, no. 14, pp. 1774–1785, 2013.
- [184] V. K. Yalavarthi, X. Ke, and A. Khan, “Select Your Questions Wisely: For Entity Resolution with Crowd Errors,” in *CIKM*, 2017.
- [185] X. Yan, X. J. Zhou, and J. Han, “Mining Closed Relational Graphs with Connectivity Constraints,” in *KDD*, 2005.

REFERENCES

- [186] J. Yang, J. Fan, Z. Wei, G. Li, T. Liu, and X. Du, “Cost-Effective Data Annotation using Game-Based Crowdsourcing,” *PVLDB*, vol. Volume 12 Issue 1, 2019.
- [187] Y. Yang, X. Mao, J. Pei, and X. He, “Continuous Influence Maximization: What Discounts Should We Offer to Social Network Users?” in *SIGMOD*, 2016.
- [188] Y. Yuan, L. Chen, and G. Wang, “Efficiently Answering Probability Threshold-Based Shortest Path Queries over Uncertain Graphs,” in *DASFAA*, 2010.
- [189] J. Zhang, S. S. Bhowmick, H. H. Nguyen, B. Choi, and F. Zhu, “DaVinci: Data-driven Visual Interface Construction for Subgraph Search in Graph Databases,” in *ICDE*, 2015.
- [190] X. Zhang and T. Özsu, “Correlation Constraint Shortest Path over Large Multi-Relation Graphs,” *PVLDB*, vol. 12, no. 5, pp. 488–501, 2019.
- [191] P. Zhao, C. C. Aggarwal, and M. Wang, “gSketch: On Query Estimation in Graph Streams,” *PVLDB*, vol. 5, no. 3, pp. 193–204, 2011.
- [192] F. Zhou, S. Mahler, and H. Toivonen, “Network Simplification with Minimal Loss of Connectivity,” in *ICDM*, 2010.
- [193] K. Zhu, W. Zhang, G. Zhu, Y. Zhang, and X. Lin, “BMC: An Efficient Method to Evaluate Probabilistic Reachability Queries,” in *DASFAA*, 2011.
- [194] R. Zhu, Z. Zou, and J. Li, “Top-k Reliability Search on Uncertain Graphs,” in *ICDM*, 2015.
- [195] L. Zou, P. Peng, and D. Zhao, “Top-K Possible Shortest Path Query over a Large Uncertain Graph,” in *WISE*, 2011.
- [196] Z. Zou, F. Li, J. Li, and Y. Li, “Scalable Processing of Massive Uncertain Graph Data: A Simultaneous Processing Approach,” in *ICDE*, 2017.