

University of Windsor

## Scholarship at UWindor

---

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

---

2019

# Scalable Methods and Algorithms for Very Large Graphs Based on Sampling

Roohollah Etemadi  
*University of Windsor*

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

---

### Recommended Citation

Etemadi, Roohollah, "Scalable Methods and Algorithms for Very Large Graphs Based on Sampling" (2019). *Electronic Theses and Dissertations*. 7700.  
<https://scholar.uwindsor.ca/etd/7700>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email ([scholarship@uwindsor.ca](mailto:scholarship@uwindsor.ca)) or by telephone at 519-253-3000ext. 3208.

# Scalable Methods and Algorithms for Very Large Graphs Based on Sampling

By

**Roohollah Etemadi**

A Dissertation  
Submitted to the Faculty of Graduate Studies  
through the School of Computer Science  
in Partial Fulfillment of the Requirements for  
the Degree of Doctor of Philosophy  
at the University of Windsor

Windsor, Ontario, Canada

2019

©2019 Roohollah Etemadi

Scalable Methods and Algorithms for Very Large Graphs Based on Sampling

by

Roohollah Etemadi

APPROVED BY:

---

J. Huang, External Examiner  
York University

---

M. Ahmadi  
Department of Electrical and Computer Engineering

---

M. Kargar  
School of Computer Science

---

D. Wu  
School of Computer Science

---

J. Lu, Advisor  
School of Computer Science

---

Y. H. Tsin, Co-Advisor  
School of Computer Science

April 23, 2019

## DECLARATION OF CO-AUTHORSHIP/PREVIOUS PUBLICATION

### I. Co-Authorship

I hereby certify that this dissertation incorporates material that is the result of my research conducted under the supervision of my advisors Prof.s Jianguo Lu and Yung H. Tsin. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by the author.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

### II. Previous Publications

This dissertation includes the extended or original version of the papers that have been previously published/submitted for publication in peer reviewed conferences and journals, as follows:

- Roohollah Etemadi, Jianguo Lu, and Yung H. Tsin. 2016. Efficient Estimation of Triangles in Very Large Graphs. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16). ACM. New York. NY. USA. pp. 1251-1260. doi:10.1145/2983323.2983849. The data and code are publicly available at <http://etemadir.myweb.cs.uwindsor.ca/cikm2016/triangles.php> (Chapter 2).
- Roohollah Etemadi, Jianguo Lu. 2017. Bias Correction in Clustering Coefficient Estimation, In Proceedings of the 2017 IEEE International Conference on Big Data (IEEE BigData'17). Boston. USA. pp. 606-615. doi:10.1109/BigData.2017.8257976. The code and data are available online at <http://etemadir.myweb.cs.uwindsor.ca/cbias> (Chapter 4).
- Roohollah Etemadi, Jianguo Lu. 2018. The estimation of bias and variance in clustering coefficient streaming algorithms. arXiv preprint arXiv:1811.01109,

2018. The data along with the code are publicly available at <http://etemadir.myweb.cs.uwindsor.ca/cbiasstream> (Chapter 4).

- Roohollah Etemadi, Jianguo Lu. 2019. PES: Priority Edge Sampling in Streaming Triangle Estimation, *IEEE Transactions on Big Data (TBD)*, 2019 (Under revision). The code and data are available online at <http://etemadir.myweb.cs.uwindsor.ca/PES> (Chapter 3).
- Roohollah Etemadi, Jianguo Lu. 2019. Bias Correction in Streaming and Non-streaming Algorithms in Clustering Coefficient Estimation, *IEEE Transactions on Knowledge and Data Engineering(TKDE)*, 2019 (Under review). The code and data are available online at <http://etemadir.myweb.cs.uwindsor.ca/cc> (Chapter 4).

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

### III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

## ABSTRACT

Analyzing real-life networks is a computationally intensive task due to the sheer size of networks. Direct analysis is even impossible when the network data is not entirely accessible. For instance, user networks in Twitter and Facebook are not available for third parties to explore their properties directly. Thus, sampling-based algorithms are indispensable.

This dissertation addresses the confidence interval (CI) and bias problems in real-world network analysis. It uses estimations of the number of triangles (hereafter  $\Delta$ ) and clustering coefficient (hereafter  $\mathcal{C}$ ) as a case study. Metric  $\Delta$  in a graph is an important measurement for understanding the graph. It is also directly related to  $\mathcal{C}$  in a graph, which is one of the most important indicators for social networks. The methods proposed in this dissertation can be utilized in other sampling problems.

First, we proposed two new methods to estimate  $\Delta$  based on random edge sampling in both streaming and non-streaming models. These methods outperformed the state-of-the-art methods consistently and could be better by orders of magnitude when the graph is very large. More importantly, we proved the improvement ratio analytically and verified our result extensively in real-world networks. The analytical results were achieved by simplifying the variances of the estimators based on the assumption that the graph is very large. We believe that such big data assumption can lead to interesting results not only in triangle estimation but also in other sampling problems.

Next, we studied the estimation of  $\mathcal{C}$  in both streaming and non-streaming sampling models. Despite numerous algorithms proposed in this area, the bias and variance of the estimators remain an open problem. We quantified the bias using Taylor expansion and found that the bias can be determined by the structure of the sampled data. Based on the understanding of the bias, we gave new estimators that correct the bias. The results were derived analytically and verified in 56 real networks ranging in different sizes and structures. The experiments reveal that the bias ranges widely from data to data. The relative bias can be as high as 4% in non-streaming model and 2% in streaming model, or it can be negative. We also derived the variances of the estimators, and the estimators for the variances. Our simplified estimators can be used in practice to control the accuracy level of estimations.

## DEDICATION

To my parents and siblings with gratitude.

To my dear and loving wife with love.

To my parents-in-law with respect.

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors, Professor Jianguo Lu and Professor Yung H. Tsin, for their continuous support of my Ph.D. study and research, for their extensive professional guidance, and for their immense knowledge.

I am greatly indebted to my dissertation committee, Professor Jimmy Huang, Professor Majid Ahmadi, Doctor Dan Wu, and Doctor Mehdi Kargar for participating in my seminars, for taking time to read my dissertation, and for providing me with their valuable comments.

I would also like to thank the Ontario Government, and the committee of Faculty of Graduate Studies of the University of Windsor for awarding an Ontario Graduate Scholarship (OGS) to me.

In addition, my thanks goes to NSERC (Natural Sciences and Engineering Research Council of Canada) and School of Computer Science for supporting my research through my advisors' grants, and for offering me GAships and TAships respectively.

I owe special thanks to the staff of School of Computer Science, Ms. Karen Bourdeau, Ms. Christine Weisener, Ms. Melissa Robinet, Ms. Gloria Mensah, Mr. Maunzer Batal, and Mr. Robert Mavrinac for their great support.

I would like to acknowledge Canterbury College and its outstanding staff's contribution in providing a comfortable and friendly student residence, where I always felt supported and welcome.

It is my privilege to thank my wife, Dr. Sholaeh Hashemi Namin, for her unending inspiration and support throughout my studies.

Last but not least, I am thankful to my parents, my parents-in-law, and my siblings for their unconditional love, encouragement, and support.



## TABLE OF CONTENTS

<b>DECLARATION OF CO-AUTHORSHIP/PREVIOUS PUBLICATION</b>	<b>III</b>
<b>ABSTRACT</b>	<b>V</b>
<b>DEDICATION</b>	<b>VI</b>
<b>ACKNOWLEDGEMENTS</b>	<b>VII</b>
<b>LIST OF TABLES</b>	<b>XI</b>
<b>LIST OF FIGURES</b>	<b>XII</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Challenges in network graphs analytics . . . . .	1
1.3 Framework of sampling methods . . . . .	3
1.3.1 Take samples . . . . .	3
1.3.2 Generalization . . . . .	4
1.3.3 Variance vs sample size . . . . .	6
1.4 Motivation . . . . .	7
1.5 Review of our contributions . . . . .	10
1.6 The structure of the dissertation . . . . .	13
<b>2 Estimation of Triangles</b>	<b>14</b>
2.1 Introduction . . . . .	14
2.2 Related work . . . . .	16
2.3 Methods . . . . .	17
2.3.1 Motivation . . . . .	17
2.3.2 Our method . . . . .	19
2.3.3 Variance of $E_G$ . . . . .	21
2.3.4 Variance of $E_g$ . . . . .	23
2.4 Experiments . . . . .	26
2.4.1 Datasets . . . . .	26
2.4.2 Experimental setup . . . . .	26
2.4.3 Verification of Theorems . . . . .	30
2.5 Discussions and conclusions . . . . .	32
<b>3 PES: Priority Edge Sampling in Streaming Triangle Estimation</b>	<b>35</b>
3.1 Introduction . . . . .	35
3.2 Background and related work . . . . .	36
3.2.1 Node-based methods . . . . .	37
3.2.2 Edge-based methods . . . . .	38
3.3 Naive edge streaming (NES) . . . . .	39

3.4	Priority edge sampling (PES)	42
3.4.1	The algorithm	42
3.4.2	Example	43
3.4.3	The unbiased estimator	45
3.4.4	The variance	47
3.5	Experiments	49
3.5.1	Data	50
3.5.2	Comparison with GPS-In and GPS-Post	50
3.5.3	Validation of Theorems 1 & 3	55
3.5.4	An implication of Theorems 1 & 3	56
3.6	Conclusions and discussions	57
<b>4</b>	<b>The Bias and Variance in Clustering Coefficient Estimation</b>	<b>59</b>
4.1	Introduction	59
4.2	Background and related work	62
4.2.1	Related work	63
4.3	The bias and variance in a non-streaming model	66
4.3.1	The sampling scheme	66
4.3.2	The bias	67
4.3.3	Counting $\Psi$ and $\Omega$	70
4.3.4	The variance	72
4.4	The bias and variance in a streaming model	74
4.4.1	Naive edge streaming (NES)	74
4.4.2	Estimator of the variance	76
4.4.3	The bias-corrected estimator	78
4.4.4	The variances of $\Delta_g$ and $\Lambda_g$ and their covariance	79
4.5	Experiments	81
4.5.1	Datasets	81
4.5.2	The bias in the non-streaming model	81
4.5.3	Positive and negative bias	84
4.5.4	Characterizing bias using second and third moments	85
4.5.5	When the bias is large	87
4.5.6	Verification of Theorem 2	87
4.5.7	Verification of Theorem 3	87
4.5.8	The bias in the streaming model	92
4.6	Discussions and conclusions	93
<b>5</b>	<b>Conclusions and Future Directions</b>	<b>95</b>
5.1	Introduction	95
5.2	Discussions and conclusions	95
5.3	Impact of our dissertation research	97
5.4	Looking into the future	97
	<b>REFERENCES</b>	<b>99</b>



## LIST OF TABLES

2.1	Summary of the notations. . . . .	18
2.2	Properties of the networks in our experiments, sorted by graph size $N$ .	25
3.1	Summary of the notations . . . . .	37
3.2	Properties of the networks in our experiments, sorted by graph size $N$ .	49
4.1	Summary of the notations . . . . .	63
4.2	Properties of the networks in our experiments, sorted by graph size $N$ .	82

## LIST OF FIGURES

1.1	The framework of sampling approaches for network analytics. . . . .	3
1.2	Examples of sampling methods. . . . .	4
1.3	Examples of unbiased and biased estimators. Estimator $\widehat{X}$ is repeated $k$ times and the estimation of metric $X$ in run $i = 1, 2, \dots, k$ is shown by $\widehat{X}_i$ , and the expectation is $\mathbb{E}[\widehat{X}] = \frac{1}{k} \sum_{i=1}^k \widehat{X}_i$ . . . . .	6
1.4	Examples of estimations with small (Panel A) and large (Panel B) variances. . . . .	6
1.5	The Confidence Interval problem of existing methods to estimate $\Delta$ . The properties of the input graph need to be used to construct the confidence interval using $(\epsilon, \sigma)$ -approximation and the variance. . . . .	7
1.6	The bias problem of existing methods to estimate $\mathcal{C}$ . . . . .	8
1.7	Overview of our contributions to estimate $\Delta$ . The estimator of variance used to construct confidence interval. . . . .	10
1.8	Overview of our contributions to estimate $\mathcal{C}$ . The bias quantified and the bias-corrected estimators proposed for the first time. . . . .	11
2.1	Illustration of $E_g$ and $E_G$ sampling. . . . .	20
2.2	Dependent wedges of two shared triangles. . . . .	21
2.3	The observed and estimated RSEs of estimator $E_G$ . Estimated values are obtained from Eq. 10. . . . .	28
2.4	The observed and estimated RSEs of estimator $E_g$ . Estimated values are based on Eq. 11. . . . .	29
2.5	The observed and estimated ratio between the sample sizes of estimators $E_g$ and $E_G$ with the same RSE. Estimated value are obtained based on Eq. 13. . . . .	31
2.6	Improvement as a function of data size $M$ and $\Delta$ . RSE=0.1. . . . .	33
3.1	Steps of applying our PES on a toy graph. . . . .	45

3.2	Sample size ratios of our PES vs. GPS-Post (Panel A), GPS-In (Panel B), and NES (Panel C) when RSE=0.2. . . . .	51
3.3	Our PES uses <b>less sample sizes</b> compared to other methods to obtain an estimation with the same RSE=0.2 on most of the graphs. Note that the sample size include both the size of the subgraph and the reservoir for our PES but for GPS methods extra memory per sampled edge was ignored. . . . .	52
3.4	Our PES outperforms GPS-In in terms of RSEs when both methods are using the same sample sizes. Note that for our PES, the sample size includes both the size of subgraph $g$ and reservoir size, i.e., $ \sigma $ . For GPS-In, we only considered the size of subgraph $g$ as a sample size, and ignored two additional values per sampled edge in $g$ . . . . .	53
3.5	The observed RSEs of $\widehat{\Delta}_{PES}$ support our estimated RSEs based on Eq. 14. . . . .	54
3.6	The observed RSEs of $\widehat{\Delta}_{NES}$ fit very well our estimated RSEs based on Eq. 3. . . . .	55
3.7	Our PES outperforms NES. The observed and estimated ratios between $p_N$ and $p_D$ when the methods achieve the same RSEs between 0.1 and 0.4. The estimated ratios are obtained using Eq. 16. . . . .	57
4.1	Illustration of shared wedges and closed-wedges. (a) A sample graph; (b) Wedge $(l, j, i)$ shares with wedge $(k, j, i)$ ; (c-d) Wedge $(l, j, i)$ shares with closed-wedges $(i, j, k)$ , $(k, i, j)$ ; (e) Wedge $(l, j, k)$ shares with closed-wedge $(j, k, i)$ . The large node in the plot indicates the centre node of the closed-wedge. E.g., in Panel (c) the closed-wedge is $(k, j, i)$ . . . . .	68
4.2	An example for computing $\Psi$ and $\Omega$ in the sample graph in Fig. 4.1 Panel (a). . . . .	71

4.3	The observed vs. estimated RB of $\widehat{\mathcal{C}}$ . The results were obtained over $5 \times 10^4$ independent runs for all graphs except for the large graphs in the last row with $10^4$ independent repetitions. The estimated RBs are obtained using Eq. 17. . . . .	83
4.4	The bias-corrected estimator $\widehat{\mathcal{C}}^+$ vs. biased estimator. The observed RBs were obtained over $5 \times 10^4$ independent runs for all graphs except for the large graphs in the last row with $10^4$ independent repetitions. . . . .	84
4.5	RB depends on the first term and second term of the Taylor expansion. The outliers are the 10 smallest graphs. Observed RBs are taken when RSE=0.2 over $10^5$ independent runs. . . . .	88
4.6	$\frac{\Psi}{\Lambda^2}$ against $\frac{2\langle d^3 \rangle}{N\langle d^2 \rangle^2}$ for 56 graphs. . . . .	89
4.7	A sample of the Stanford Web network. . . . .	89
4.8	The observed vs. the estimated RSEs. Estimated RSEs are obtained based on Eq. 32 over 100 independent runs. . . . .	90
4.9	The observed RSEs of $\widehat{\mathcal{C}}$ fit perfectly the estimated RSEs based on Eq. 39 in representative graphs. . . . .	91
4.10	The estimated RSEs obtained based on Eq. 40 are apt estimations for the observed RSEs of $\widehat{\mathcal{C}}$ . . . . .	91
4.11	The observed RBs of $\widehat{\mathcal{C}}$ (biased estimator) support our estimations of RB based on Eq. 47. . . . .	92
4.12	Our biased-corrected $\widehat{\mathcal{C}}^+$ removes the bias perfectly. . . . .	92
4.13	Comparison of sample sizes, i.e. number of sampled edges, of the methods when RSE=0.2. The sample size of NES is comparable with GPS-In [1]. . . . .	93

---

## CHAPTER 1

# Introduction

---

---

## 1.1 Introduction

Most of the data in real-world are in the form of networks. Online social networks such as Facebook, Twitter, and many more are examples of such network data. In academia, co-authorship and citation networks are other examples. Such network data are modeled by graphs. Analyzing such network graphs attracts increasing attention from industry and academia in recent years. However, the nature of such network graphs bring some challenges. This chapter outlines the main challenges in analyzing real-life network graphs. It also reviews possible solutions in this direction. Then, the main contributions of this dissertation will be outlined. The final section will contain the structure of the rest of this dissertation.

## 1.2 Challenges in network graphs analytics

Many metrics such as network size, average degree, average shortest path length, graph centralities such as PageRank, betweenness, Katz, the number of triangles, and clustering coefficient have been utilized to analyze the complex structure of network graphs. Moreover, a number of tools and APIs such as NoSQL, NetworkX, and GraphX have been developed to compute such metrics in recent years. However, computing exact values of graph metrics are computationally an intensive task or even impossible in the following scenarios.

- **Big data:** When the size of the graph is large or even medium, exact computing of most of the graph metrics is infeasible or even impossible due to the time and space complexities of algorithms. Most of the real-world networks are



massive in their size. For example, Facebook had more than 2 billion monthly active users as the second quarter of 2018 <sup>1</sup>. Another example is the Internet with more than 5.48 billion web pages over the world by March 2019<sup>2</sup>. Neuronal networks, Protein-protein and DNA–protein interaction networks are other examples from the biology domain. Such network data result in constructing graphs with millions of nodes and billions of edges. Thus, applying exact algorithms on massive network graphs is computationally demanding. For example, enumerating triangles using the best-known algorithm has a time complexity of  $O(M^{3/2})$  [2, 3], where  $M$  is the number of edges in the input graph. Obviously, applying such a method for example on Facebook network graph with billion of edges is inefficient.

- **Hidden data:** The entire data are inaccessible for third parties in most of the real networks. For example, the network data of online social networks such as Facebook and Twitter are hidden behind search-able interfaces due to the privacy of their users. Hence, exact computing is impossible when the data are not entirely available.

Thus, designing efficient methods to deal with such challenges are indispensable. One possible solution is to utilize high-speed machines, and parallel and distributed computations. However, such a solution is costly and not available for all. Furthermore, the exact computing is not essential in many real applications. Therefore, the accuracy can be traded against computation time and memory usage. Thus, an estimation with confidence interval using reasonable time and memory is desired. In such a case, sampling techniques have largely been utilized to estimate network graph statistics [4–13]. Sampling methods take a small amount of data from a massive network graph. Then, the properties of the sampled data are generalized to the entire network. Obviously, the sampled data is much smaller than the original one. Thus, analyzing sample data needs less CPU time and memory usage. Furthermore, the entire data are not required in case of hidden data.

---

<sup>1</sup> <https://www.statista.com>

<sup>2</sup> <https://www.worldwidewebsize.com/>

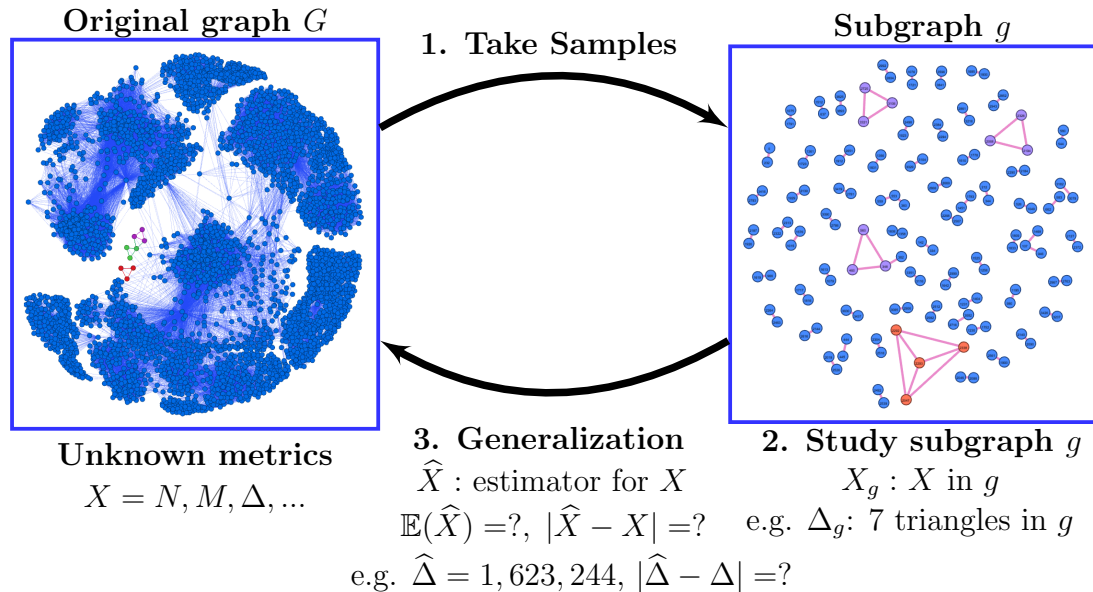


FIGURE 1.1: The framework of sampling approaches for network analytics.

### 1.3 Framework of sampling methods

Suppose network data are modeled by graph  $G$  and metrics of  $G$  such as the network size  $N$ , the number of edges  $M$ , average degree  $\langle d \rangle$ , degree distribution, the number of triangles  $\Delta$ , clustering coefficient  $\mathcal{C}$ , average shortest path length, etc are unknown. The goal is to design a sampling method to take subgraph  $g$  from original graph  $G$  and study the metrics of  $G$  using statistics of sampled graph  $g$ . The framework of such a sampling technique is shown in Fig. 1.1. It takes samples from the original graph  $G$  to create subgraph  $g$  and generalizes the properties of  $g$  to the original graph. In other words, it aims to design estimator  $\hat{X}$  for metric  $X$  in  $G$ . The following steps need to be considered to design estimator  $\hat{X}$ .

#### 1.3.1 Take samples

The first step is to take sample data to create subgraph  $g$ . Different sampling methods can be used to sample data. Such methods depend on the way to access to the original graph  $G$ . Methods can have random or sequential access to the whole or part of the original graph data. Obtaining uniform random samples is desired.

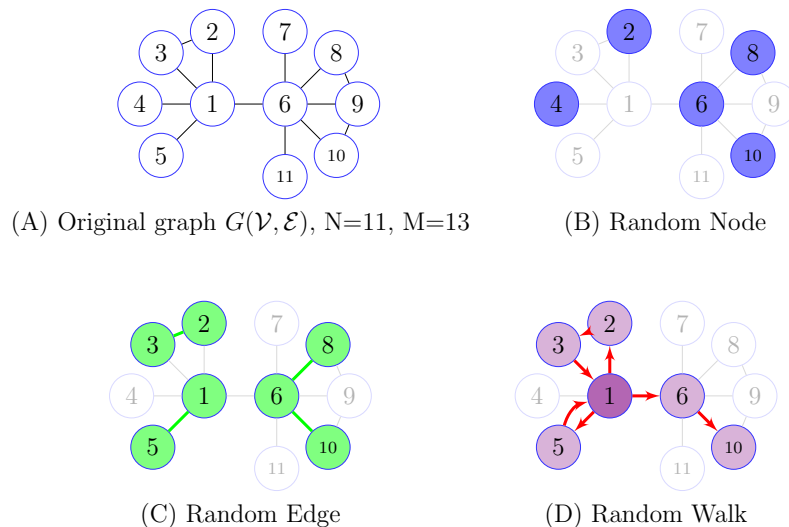


FIGURE 1.2: Examples of sampling methods.

Fig. 1.2 shows three graph sampling methods on a toy graph in Panel A. When random access to graph  $G$  is available, *Random Node* (Panel B) or *Random Edge* (Panel C) methods can be used to take samples. For example, 5 random nodes are selected in Panel B. Random Walk (Panel D) is an option when the network data is not entirely available. It starts from a node (node 1 in Panel D) at random and explores the graph by randomly walking on the edges. Note that different sampling schemes need to be designed to estimate different metrics. In other words, designing a sampling method depends on the metric of interest.

Graph sampling approaches are categorized based on their access to the original graph into two main models: *streaming* and *none-streaming*. In the former model, a limited number of sequential passes over the stream of graph data are used to take samples. In the later model, random access to the original graph data is available. The goal of methods in both models is to obtain an accurate estimation using a limited memory window to store sampled data.

### 1.3.2 Generalization

After taking sampled data, the next step is generalizing the properties of sampled graph  $g$  to the original graph  $G$ . Estimators are used to generalizing the statistics in

sampled data to the original network. Two important tasks need to be done in this step.

- **Unbiasedness vs biasness:** The first task is to show that the estimation is biased or unbiased. The desired estimator is unbiased, i.e. the expectation of the estimations of a metric needs to be the same as the true value of the metric. Let  $\hat{X}$  be an estimator for metric  $X$ . Estimator  $\hat{X}$  is unbiased if  $\mathbb{E}[\hat{X}] = X$ ; otherwise it is biased. For a biased estimator, the bias needs to be quantified and corrected. Examples of unbiased and biased estimators are shown in Fig. 1.3. Suppose estimator  $\hat{X}$  is repeated  $k$  times and  $\hat{X}_i$  is the estimation in repetition  $i$ . Panel (A) shows an unbiased estimator. When  $k$  is large enough,  $\mathbb{E}[\hat{X}] = \frac{1}{k} \sum_{i=1}^k \hat{X}_i = X$ . In contrast, the estimator  $\hat{X}$  in Panel (B) is biased because the expectation of the estimations ( $\mathbb{E}[\hat{X}]$ ) is far away from the true value of metric  $X$ . Thus, for each estimator, one needs to prove the unbiasedness and for biased estimators, the bias needs to be quantified and corrected.
- **Confidence interval:** The other important task is to construct the confidence interval (hereafter CI) of the estimator. In other words, one needs to show that the estimation is how far away from the true value with a specific confidence. Suppose, for example, the number of complete subgraphs with size three (triangles) in graph  $G$  needs to be estimated. As shown in Fig. 1.1, the estimation of  $\Delta$  in  $G$  is 1,623,244 using the number of triangles in  $g$  ( $\Delta_g = 7$ ). The question is that how accurate is  $\hat{\Delta} = 1,623,244$  or  $|\hat{\Delta} - \Delta| = ?$ . To answer such a question, one needs to construct the CI of the estimator. Thus, constructing the CI to understand the error bound of an estimator with a certain confidence level is an indispensable task.

Several methods have been used to form the CI. When the variance of an estimator is not available, Hoeffding's inequality can be used [14]. Chebyshev's inequality is another option to have more accurate bound using the variance of the estimator [15]. However, using such methods has several drawbacks on large graphs. Firstly such inequalities are held for any distribution. Therefore,

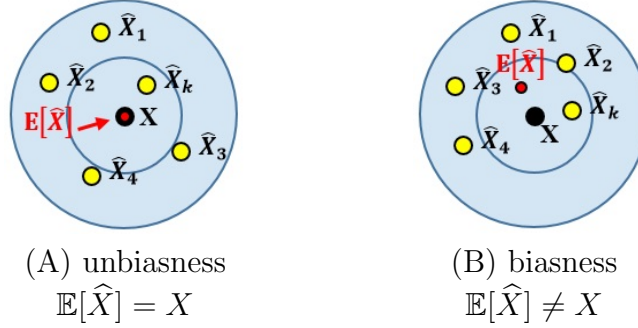


FIGURE 1.3: Examples of unbiased and biased estimators. Estimator  $\hat{X}$  is repeated  $k$  times and the estimation of metric  $X$  in run  $i = 1, 2, \dots, k$  is shown by  $\hat{X}_i$ , and the expectation is  $\mathbb{E}[\hat{X}] = \frac{1}{k} \sum_{i=1}^k \hat{X}_i$ .

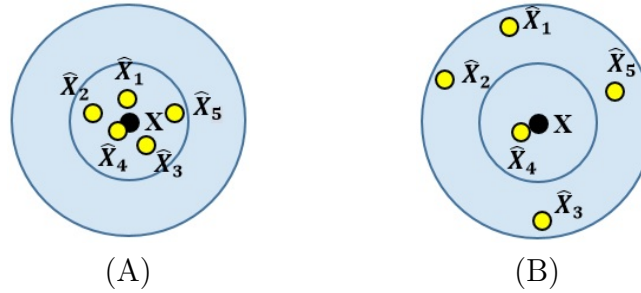


FIGURE 1.4: Examples of estimations with small (Panel A) and large (Panel B) variances.

the resulting bounds are not tight enough. More importantly, the properties of the original graph  $G$  should be known in advance to use such inequalities. Note that the properties of  $G$  are unknown in sampling scenario. Thus, we argue that using  $(\epsilon, \delta)$ -approximation is not practically useful to control the accuracy of estimators on massive networks. In practice, the properties of sampled graph  $g$  need to be used to construct the CI.

### 1.3.3 Variance vs sample size

The main goal of sampling algorithms is to achieve an estimation with a small error bound (variance) by using a small number of samples. Let  $n$  be the sample size of estimator  $\hat{X}$ . The relation between the variance of  $\hat{X}$  and its sample size  $n$  is:

$$\text{var}(\hat{X}) \propto \frac{1}{n}. \quad (1)$$

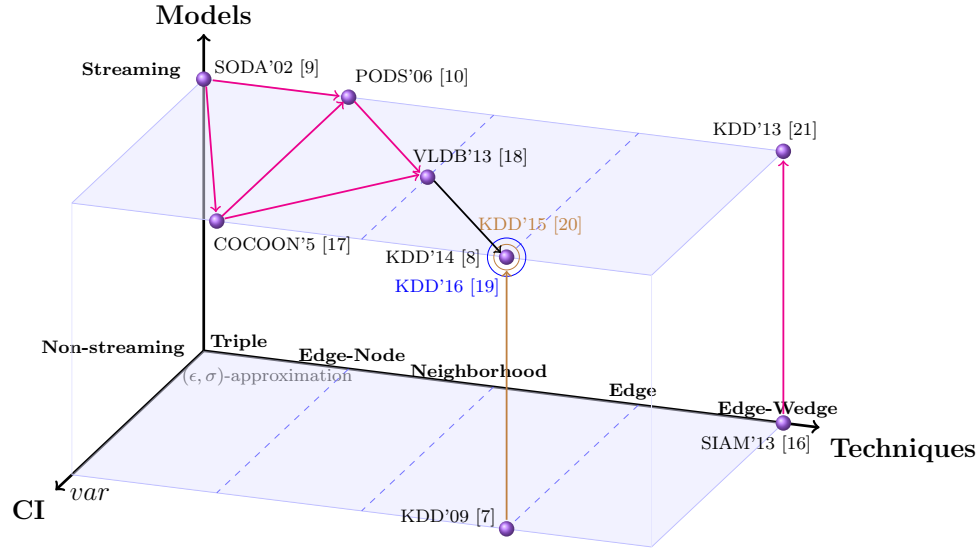


FIGURE 1.5: The Confidence Interval problem of existing methods to estimate  $\Delta$ . The properties of the input graph need to be used to construct the confidence interval using  $(\epsilon, \sigma)$ -approximation and the variance.

According to Eq. 1, the variance of an estimator is inversely correlated with its sample size, i.e. the variance increases by decreasing  $n$ . Thus, designing an estimator to obtain accurate estimations (low variance) using small sample sizes is challenging.

The desired estimator needs to use a small number of samples to obtain estimations with small variance. For example, suppose both the estimators in Panels (A) and (B) in Fig. 1.4 use the same sample sizes to estimate  $X$ . The estimator in Panel (A) is preferable due to the small variation of estimations.

## 1.4 Motivation

A number of sampling-based approaches have been proposed to estimate graph metrics such as network size, e.g. in [11, 12, 22–24], average shortest path length [25, 26], the number of triangles  $\Delta$ , e.g. in [1, 7–10, 19–21, 27, 28], clustering coefficient  $\mathcal{C}$  [1, 8, 29, 30], and many more. We visualized sampling methods to estimate  $\Delta$  in Fig. 1.5 and  $\mathcal{C}$  in Fig. 1.6 in recent years. We make several observations as follows:

- Existing methods use properties of original network graphs to construct CI. As shown in Fig. 1.5, earlier methods use  $(\epsilon, \sigma)$ -approximation to construct

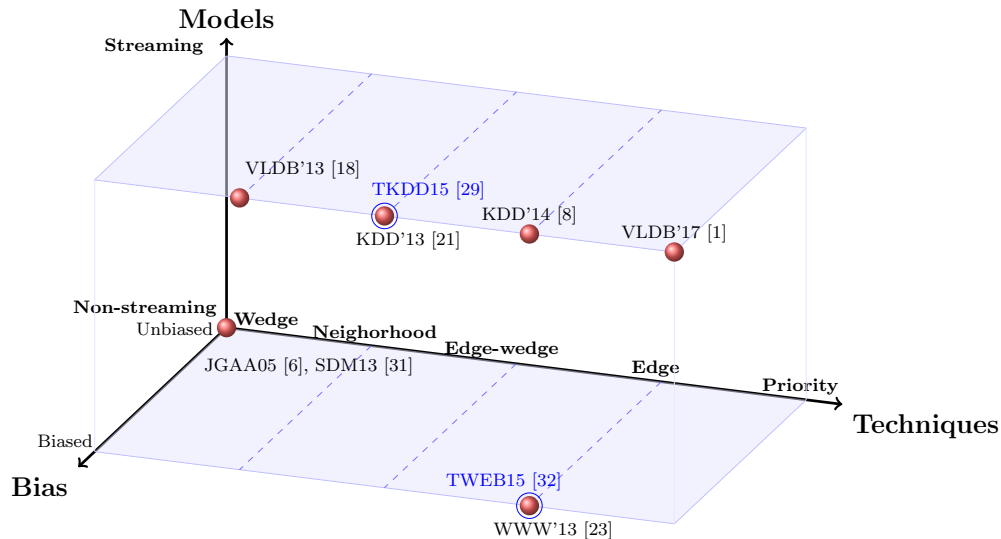


FIGURE 1.6: The bias problem of existing methods to estimate  $\mathcal{C}$ .

confidence interval. Later, the variance has been used by methods to obtain tighter error bounds. However, both  $(\epsilon, \sigma)$ -approximation and variance techniques require the properties of the original network to construct the CI. Thus, such techniques are not useful when the properties of the original network are unknown which is the case in sampling scenarios.

- The random edge based methods are efficient to estimate metric  $\mathcal{C}$ ; but they suffer from the bias problem as shown in Fig. 1.6. The bias problem was noticed in the literature [1, 8, 29]. However, to the best of our knowledge, the bias was not quantified and not corrected. Thus, it is left as an open problem.
- The estimation of  $\Delta$  and  $\mathcal{C}$  is an active and hot topic in top-tier conferences and journals. Thus, designing efficient estimators for the properties of real-life networks is indispensable.

Motivated by those observations, this dissertation addresses the CI and bias problems in graph sampling methods. It uses two case studies, i.e. the estimation of the number of triangles and its close metric, clustering coefficient, to study such problems. However, our techniques to construct CI and to quantify the bias can be used for other sampling problems as well.

Metrics  $\Delta$  and  $\mathcal{C}$  are important to reveal the complex structure of real-world networks specially online social networks. They have been used in many applications such as community detection [33], graph clustering [34], link prediction [35], spam detection [36], finding interesting individuals [37], characterizing the structure of balanced network [38], wireless and ad hoc networks analysis [39], blog analysis in online social networks, DAN sequence analysis [40], prediction of essential proteins [41], microarray data analysis for finding cancer genes [42, 43], identifying modular formations in protein-protein interaction network [44], risk analysis in economy [45], word-learning in education [46], and many others.

Based on the nature of real-life networks, different scenarios have been considered to study the estimation of metrics  $\Delta$  and  $\mathcal{C}$  [7, 10, 23]. In this dissertation, we assume that the network data is modeled as an undirected and simple graph. Hereafter we will use terms original and input graph interchangeably to call the input network graph. Suppose  $G$  is an undirected and simple network graph. We will address the following problems in the rest of this dissertation.

Firstly the estimation of  $\Delta$  and  $\mathcal{C}$  will be discussed in a non-streaming model which the random access to the original graph is provided. Thus, we define the problem as follows:

**Problem 1.** *Suppose that edges/nodes of original graph  $G$  are accessible uniformly at random. Design a sampling method to estimate  $\Delta$  and  $\mathcal{C}$  in graph  $G$ . What is the best sampling method to estimate  $\Delta$  and  $\mathcal{C}$ ? How the sampled data can be used to estimate the error bound of the estimation?*

Next, we study the estimation of the metrics in a streaming model. The sequential access to the network data is provided in this model instead of uniform random access.

The goal is to achieve an accurate estimation by storing a small fraction of data using a single pass or a limited number of passes over the network data. Thus, our second focus will be as follows.

**Problem 2.** *Suppose that edges of graph  $G$  come in an arbitrary (random) order in an edge stream. Design sampling methods to estimate  $\Delta$  and  $\mathcal{C}$  using a limited*



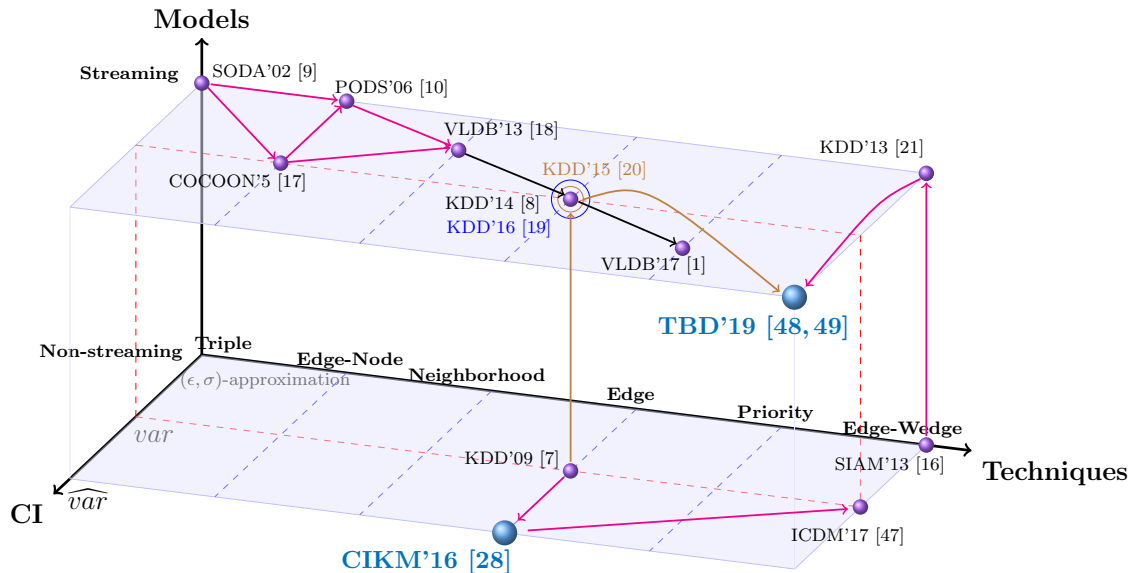


FIGURE 1.7: Overview of our contributions to estimate  $\Delta$ . The estimator of variance used to construct confidence interval.

*memory window to store the sampled data over a single pass on the stream.*

## 1.5 Review of our contributions

This dissertation addressed the CI, and the bias problems in sampling methods to analyze large-scale networks. The main contributions of this dissertation are summarized as follows.

- **To estimate  $\Delta$ :** We proposed two new methods to estimate the number of triangles based on random edge sampling in both streaming and non-streaming models (see Fig. 1.7). Our methods improve the traditional random edge sampling by probing the edges that have a higher probability of forming triangles. The methods outperform the existing methods consistently and can be better by orders of magnitude when the graph is very large. The results were demonstrated on 56 graphs, including the largest graphs we can find. More importantly, we proved the improvement ratio and verified our result on all the datasets. The analytical results were achieved by simplifying the variances of the estimators based on the assumption that the graph is very large. We believe

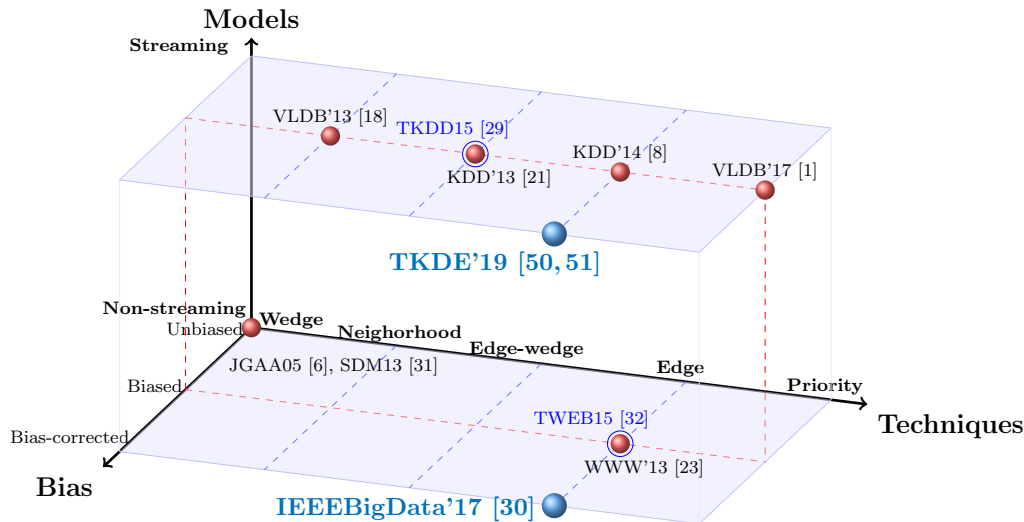


FIGURE 1.8: Overview of our contributions to estimate  $\mathcal{C}$ . The bias quantified and the bias-corrected estimators proposed for the first time.

that such big data assumption can lead to interesting results not only in triangle estimation but also in other sampling problems.

- **To estimate  $\mathcal{C}$ :** Biased-corrected estimators in the streaming and non-streaming models were proposed (see Fig. 1.8). Although edge-sampling based methods are efficient, they result in a biased estimator for  $\mathcal{C}$  noticed in the literature, e.g. in [1, 8, 21, 29]. However, the bias has not been quantified and not corrected. Thus, we used quadratic Taylor expansion to quantify the bias of the estimators. We found that the bias hangs on the structure of input graphs. To the best of our knowledge, we are the first one who could quantify the bias of such estimators. To sum up, our contributions in this direction are proposing the estimators of the bias and the variance of the estimators for  $\mathcal{C}$ . The bias and variance phenomenon varies greatly from a graph to graph. To find out the patterns behind, we conducted extensive experiments with many different kinds of graphs. In total, we utilized 56 real-life graphs from a variety of areas such as online social networks, web graphs, Co-authorship, and citation networks. The graph size also varies from about  $4 \times 10^3$  (very small) to  $65 \times 10^6$  (very large). The experiments reveal that the bias ranges widely from data to data. The relative bias can be as high as 4% in the non-streaming model and 2% in the

streaming model or can be negative. For most of the graphs, the bias is small, although every graph does have a bias as quantified by our analytical results.

- **Analytical results:** We derived the theoretical results of the proposed estimators. Moreover, we quantified the performance ratio between the methods for the first time. We demonstrated that our results are data independent, i.e. proposed methods can be run on any real-world network graph without any restriction on the structure of the graph.
- **Big data assumption:** The analytical results of the estimators were simplified to have better insight into them by assuming that the input graph is very large. Furthermore, we derived the estimators for the variances of the estimators. Our estimator for the variance has two main implications. First, it can be used to quantify the performance ratio between the estimators not only for  $\Delta$  and  $\mathcal{C}$  but also for other metrics. Studying the literature shows that existing methods were compared using experimental results [1, 8, 10, 18, 19, 21, 29]. The drawback of such a comparison is that the results are data dependent and vary from data to data. Thus, we were motivated to give an idea to quantify the performance ratio between the estimators. Second, it can be used to control the accuracy of the estimators in practice which is important for practitioners. To use an estimator in real applications, one needs to decide about the sample size to achieve an estimation in a given confidence interval. Our simplified estimators for the variance can be used to determine the sample size of estimators to obtain an estimation with a given error bound.
- **Publicly available data and code:** Obtaining ground truth and cleaning data for a sampling purpose for large graphs are time-consuming tasks. Two servers with 256GB memory and 24 cores each were used to complete such tasks. To accelerate the research in this direction, we made the data and our codes publicly available.

## 1.6 The structure of the dissertation

The rest of this dissertation is organized as follows. Chapter 2 presents our estimator for  $\Delta$  in a non-streaming model. It contains our analytical and experimental results for proposed and existing estimators. Our estimator for  $\Delta$  in a streaming model will be discussed in Chapter 3. Furthermore, it presents the derivation of the analytical results along with their validations using extensive experiments. Our bias-corrected estimators for  $\mathcal{C}$  are presented in Chapter 4. The conclusions of the dissertation along with our future directions will be presented in Chapter 5.

---

## CHAPTER 2

# Estimation of Triangles

---

---

## 2.1 Introduction

Graphs are used to model interactions in many applications in online social networks, biology, biochemistry, and many other domains. The counts of triangles in such graphs is an important structural property. For example, in online social networks, it is used to measure with what probability friends of friends are also friends (clustering coefficient [52, 53]). Counting triangles has also various applications such as spam detection [36] in computer networks, community detection and blog analysis [33] in social networks, protein identification [54], DNA sequence analysis [40] in biology, study of systemic risk [45], tracking the evolution of international trade [55] in economy, and more.

Enumerating triangles in massive graphs is not practical because the best-known algorithm has a complexity of  $O(M^{1.41})$  in time and  $\Theta(N^2)$  in space using the fastest matrix multiplication [3, 56], where  $N$  and  $M$  are the number of nodes and edges in the input graph. Thus, sampling algorithms are indispensable. Substantial work has been done on the streaming model where data items arrive sequentially and there is a limited memory window [1, 8, 10, 17–20, 29, 57]. Many streaming algorithms are designed specially to tackle such sampling restrictions. This research focuses on a more generic sampling scenario without the streaming restriction. It can be potentially applied to the estimation of triangles in very large graphs, especially when a graph in its entirety is not available. For instance, many large networks, such as

---

A short version of this chapter was published in CIKM'16 [28].

Twitter and Facebook user network, are hidden behind searchable interfaces. Their properties can be only estimated by taking a sample from them.

When estimating the number of triangles, the most natural, and a naive one, is to take triplets (three nodes) uniformly at random, then check whether they form triangles [9]. Obviously, this method is too costly to be of practical use. Most graphs, especially the large ones, are sparse. Hence, the vast majority of the triplets have zero to two edges. It means that the cost of observing even one triangle in this method will be exorbitantly high. Buriol et al. ameliorate this problem by skipping the cases for zero edges [10] called *EN* in this dissertation. They proposed to start with one random edge, then check whether there are triangles surrounding this edge. This method can be interpreted as starting with three random nodes, with the pre-condition that there needs to be at least one edge already in the triplet. When a random edge is given, there are numerous variations to check whether there is a containing triangle. [10] takes a random node from the remaining set; [7] continues to select more random edges hoping to obtain a triangle. The method proposed in [7] can be regarded as a random edge method: it selects random edges, forms a subgraph from the random edges. Then the count of the triangles in the subgraph is used to estimate the number of triangles in the original graph.

Both methods in [10] and [7] still suffer from the scarcity of triangles in the subgraph. In [10], although it skips the triplets with zero edges, it could be better also to skip triplets with one edge only, by starting with the triplets that have at least two edges. For [7], triangle count in the subgraph can increase if we check their edges not only in the subgraph, but also in the original graph.

Motivated by these observations, we present a new sampling method that combines the ideas from both [7] and [10]. The first step is the random edge sampling that is the same as in [7]. Then, for every path of length two in the subgraph, we check the existence of the third edge in the original graph.

In this dissertation, we give the unbiased estimator and its variance for our sampling method. The variance is a long formula that involves several parameters, thereby it does not provide useful insight into the estimator, nor can it be compared with other

sampling methods. Hence, we simplify the formula based on the assumption that the graph is very large. The simplified RSE (relative standard error) is  $1/\sqrt{3\Delta|_g}$ , where  $\Delta|_g$  is the number of triangles restricted to the subgraph  $g$ . Intuitively, from the formula we can infer the 95% confidence interval by looking at the triangles in the subgraph. After doing a similar treatment for the random edge method, we can compare the performance of these two estimators analytically. The analytical analysis demonstrates that our method is always better than the other method. This is confirmed by empirical experiments on 56 graphs, including the largest networks we can find.

Our contribution is twofold, in both the result and the method. For the result, we present a new estimator that outperforms the random edge method by orders of magnitude; For the method, we use the big data assumption to simplify the variances of various estimators. Thereby, performances of different triangle estimators can be compared analytically for the first time.

In presenting our theorems, we do not use the  $\epsilon - \delta$  approximation notation as most other papers do, as it is self-evident from Chebyshev's inequality. What is more, Chebyshev's inequality is valid for any data distribution, hence it gives a loose range that has little practical implication. Estimates produced by multiple runs follow a normal distribution. This is implied by the central limit theorem and is verified by our experiments. The central limit theorem can be applied in this case because each estimation involves the summation (mean) of probabilities for all the triangles being sampled. With such normal distribution, we have a much tighter confidence interval, i.e., 95% confidence interval is within two standard deviations. Hence, in the remaining part of the chapter only RSE and variance are discussed.

## 2.2 Related work

A number of methods have been proposed to estimate  $\Delta$  in streaming and non-streaming models [1, 7–10, 16–21, 29, 47, 58, 59]. In a streaming model, methods have a sequential access to the entire network data and estimate  $\Delta$  using one or several passes

over the network data. In the later model, uniform random access to the network data is required and the methods do not need to access the whole data. Thus, the later model is more general and useful when the whole data are not available and this is the case for real-life networks. Thus, in this chapter, we review methods in a non-streaming model. We also adjusted methods in a streaming model to a non-streaming when it is possible.

A naive method to estimate  $\Delta$  is *triple sampling* [9]. It selects three random nodes to form a triplet and checks the existence of edges among the nodes. Suppose  $n$  is the number of sampled triplets and  $v$  is identified triangles among them. Thus, an unbiased estimator for  $\Delta$  is  $v\binom{N}{3}/n$  and its variance is  $(\Delta\binom{N}{3} - \Delta^2)/n$ . Obviously, this method suffers from the paucity of triangles in the sample in sparse graphs which is the case for real-life networks. Thus, it has been improved by decreasing the sample space from  $\binom{N}{3}$  to  $M(N - 2)$  in [10] and called Edge and Node (EN) sampling here. The idea is to construct a sampled triplet using two connected nodes (a random edge) and a node from the remaining nodes. An unbiased estimator for  $\Delta$  using EN method is  $vM(N - 2)/3n$  and its variance is  $\Delta(MN - 2M - 3\Delta)/3n$ . Although EN has a small variance compared to *triple sampling*, it still suffers from a scarcity of triangles in the sample.

To increase the chance of identifying triangles in the sample, *Edge sampling* was proposed [7]. It selects edges uniformly at random to form a subgraph  $g$ . The number of triangles in the subgraph  $g$  is used to estimate  $\Delta$ . The authors proved that the estimator is unbiased, and derived its variance.

## 2.3 Methods

### 2.3.1 Motivation

Let  $G(\mathcal{V}, \mathcal{E})$  be an undirected graph, where  $\mathcal{V}$  is the set of nodes, and  $\mathcal{E}$  the set of edges. The graph is not a multi-graph and does not have self-loops. Suppose  $N = |\mathcal{V}|$ ,  $M = |\mathcal{E}|$ , and  $\Delta$  denote the number of triangles in  $G$ . A *wedge*  $\mathcal{W}$  is a path  $u - v - w$



TABLE 2.1: Summary of the notations.

Notation	Meaning
$G(\mathcal{V}, \mathcal{E})$	Original graph
$N, M$	Number of nodes and edges in $G$
$n$	Sample size
$\langle d \rangle$	Average degree
$\Delta$	Number of triangles in $G$
$\Phi$	Number of triangle pairs that share an edge
$g$	A subgraph of $G$
$\Delta_g$	Number of triangles in $g$
$\Delta _g$	Number of triangles restricted in $g$
$E_g$	Random edge sampling method
$E_G$	Our method that checks wedge closure in $G$

of length two, where  $u, v, w \in \mathcal{V}$ ,  $u \neq w$ ,  $(u, v) \in \mathcal{E}$ , and  $(v, w) \in \mathcal{E}$ . A wedge  $\mathcal{W}$  is closed if  $(u, w) \in \mathcal{E}$ . Otherwise it is open. Note that each triangle has three (closed) wedges.

Given a subgraph  $g$  of  $G$ , we use  $\Delta_g$  to denote the number of triangles in  $g$ , and  $\Delta|_g$  the number of triangles restricted to the wedges in  $g$ , i.e., for every wedge  $u-v-w$  in  $g$ , we check whether  $(u, w) \in \mathcal{E}$ . More formally,

$$\Delta|_g = \frac{1}{3} |\{(u, v, w) \mid (u, v), (v, w) \in g, (u, w) \in G\}|.$$

To estimate  $\Delta$ , a straightforward algorithm is the random edge sampling proposed by Tsourakakis et al. [60], which is called Doulin in [7], and called  $E_g$  in this dissertation because it depends on the triangles in the sample graph  $g$ . The process is as follows: it selects random edges with an equal probability  $p$  to generate a subgraph  $g$ . Then, the count of triangles in  $g$  is used to approximate  $\Delta$  in  $G$  with the estimator

$$\widehat{\Delta}^{E_g} = \frac{\Delta_g}{p^3}. \quad (1)$$

The problem of the method is the scarcity of triangles in the sample graph. We can verify this by looking at the formula for the expected number of triangles in the

sample graph  $g$ , which is

$$\mathbb{E}(\Delta_g) = \Delta p^3. \quad (2)$$

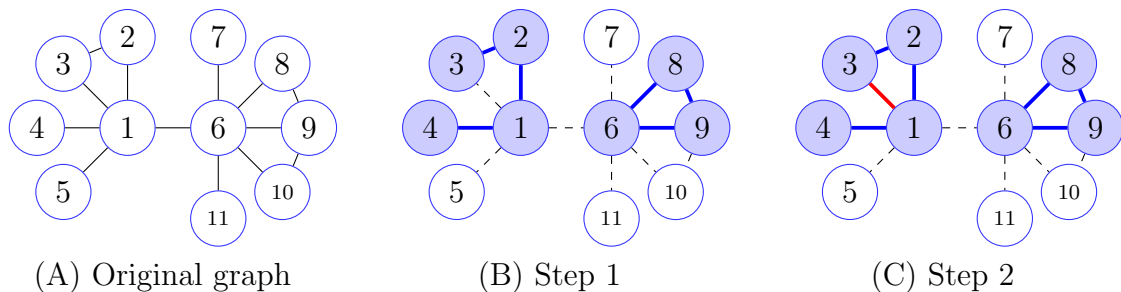
Because of the cubic function for a small  $p$ , we can barely see triangles in a sample graph. This problem is more acute when the graph is very large, henceforth the sampling probability is very small. In our subsequent experiments,  $\Delta$  can be in the order of  $10^{10}$ , and  $p$  is in the order of  $10^{-5}$ . In this scenario, it is obvious that it is far from observing any triangles in  $g$ , let alone enough number of triangles to guarantee the accuracy of estimation. It is necessary to devise a new sampling method that can increase the expected number of triangles in the sample.

### 2.3.2 Our method

The main idea of our method is to sample edges that have a higher probability of forming triangles. In social networks and other information networks, it is established that friends of a friend have a higher probability of being friends as well. Thus, it would be beneficial to sample the edges for open wedges in a partially sampled graph. Following this rationale, our method divides the sampling into two steps. The first step is the same as a normal random edge sampling [60]: we take random edges with equal probability  $p$ . In the second step, in addition to counting the triangles in  $g$ , we also look at the open wedges in  $g$ , and check the closeness of these open wedges in the original graph. The estimator is no longer the one in the  $E_g$  method. Instead, we give the estimator for  $E_G$  as

$$\hat{\Delta}^{E_G} = \frac{\Delta|_g}{p^2}, \quad (3)$$

which will be proved in the next section. Intuitively, we count the number of triangles that are restricted to  $g$ , then multiply it by a factor of  $1/p^2$ . Compared with the  $E_g$  method, the number of observed triangles can be larger by a factor of  $1/p$  under similar sampling cost.

FIGURE 2.1: Illustration of  $E_g$  and  $E_G$  sampling.

**Example 1.** *Fig. 2.1 illustrates our sampling method. In this graph  $G$ ,  $\Delta = 3$ . Suppose that the sampling probability  $p = 0.5$ , and six distinct edges are selected, resulting in a subgraph  $g$  depicted in Panel (B). There is one triangle in  $g$ . Hence the estimate using the random edge method  $E_g$  is*

$$\widehat{\Delta}^{E_g} = \frac{\Delta_g}{p^3} = \frac{1}{0.5^3} = 8. \quad (4)$$

*In our  $E_G$  sampling, the first step is the same as  $E_g$ , i.e., six edges are selected with an equal probability  $p = 0.5$ . Then, there is an additional step to check the closeness of every open wedge. In the example, two wedges  $3-2-1$  and  $4-1-2$  are checked, and it is found that wedge  $3-2-1$  is closed. Recall that there is already one triangle in the subgraph, which is equivalent to three closed wedges. Hence, all together there are four closed wedges, or  $\Delta|_g = 4/3$ . Note that in our sampling method,  $\Delta|_g$  does not have to be an integer because it is  $1/3$  of the closed wedges observed. The sampling cost is 8 because it checked 8 edges in total. The estimate is*

$$\widehat{\Delta}^{E_G} = \frac{\Delta|_g}{p^2} = \frac{4/3}{0.5^2} = \frac{16}{3}. \quad (5)$$

Our method applies extra checks in return for more triangles. One question is whether these additional triangles are worth the checking cost. Intuitively, the checking cost is proportional to  $\mathcal{C}$  (clustering coefficient), which measures the probability of seeing a triangle for an open wedge. If  $w$  is the number of open wedges in  $g$ , we need to conduct closeness check  $w$  times. There will be on average  $w \times \mathcal{C}$  number of additional triangles. In other words,  $1 - \mathcal{C}$  percent of the checks are wasted. Note

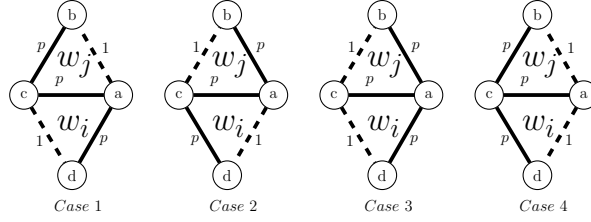


FIGURE 2.2: Dependent wedges of two shared triangles.

that for most networks,  $\mathcal{C}$  is well above 0.01. On the other hand, the vast majority of the edges do not form triangles, especially when the graph is very large and the sample size is small. In those large graphs in our experiments, we need sample edges in the order of  $10^5$  to form one triangle. Compared with this small success ratio, the cost of extra wedge probing is negligible.

In the following, we derive the variance of this estimator, and compare it with  $E_g$ .

### 2.3.3 Variance of $E_G$

Let  $w_i$  be an indicator for the  $i^{\text{th}}$  closed wedge in the input graph  $G$ .  $w_i$  is 1 when the two edges in the  $i^{\text{th}}$  closed wedge are sampled, otherwise it is 0. Since each triangle has three wedges, there are  $3\Delta$  closed wedges. We label them from 1 to  $3\Delta$ . The number of triangles restricted to  $g$   $\Delta|_g = \frac{1}{3} \sum_{i=1}^{3\Delta} w_i$ . For each wedge, the probability of sampling is  $p^2$ . The expected number of closed wedges in  $G$  that are sampled in  $g$  is:

$$3\mathbb{E}(\Delta|_g) = \mathbb{E}\left(\sum_{i=1}^{3\Delta} w_i\right) = \sum_{i=1}^{3\Delta} \mathbb{E}(w_i) = \sum_{i=1}^{3\Delta} p^2 = 3p^2\Delta.$$

Therefore the unbiased estimator for  $E_G$  sampling is

$$\widehat{\Delta}^{E_G} = \frac{\Delta|_g}{p^2}. \quad (6)$$

The variance of the estimator is more complicated due to the covariance between

wedges. By definition,

$$\begin{aligned}
 \text{var}(\hat{\Delta}^{E_G}) &= \text{var}\left(\frac{\Delta|_g}{p^2}\right) \\
 &= \text{var}\left(\frac{1}{3} \sum_{i=1}^{3\Delta} \frac{1}{p^2} w_i\right) \\
 &= \frac{1}{9p^4} \sum_{i=1}^{3\Delta} \sum_{j=1}^{3\Delta} \text{cov}(w_i, w_j) \\
 &= \frac{1}{9p^4} \left( \sum_{i=1}^{3\Delta} \text{var}(w_i) + \sum_{i \neq j} \text{cov}(w_i, w_j) \right) \tag{7}
 \end{aligned}$$

Random variable  $w_i$  follows a binomial distribution, whose variance is  $p^2(1-p^2)$ . The covariance of two independent variables  $w_i$  and  $w_j$  is zero. When  $w_i$  and  $w_j$  are dependent, they share one edge in common. When this happens, there are four cases as depicted in Fig. 2.2. Their covariance is  $\text{cov}(w_i, w_j) = \mathbb{E}(w_i w_j) - \mathbb{E}(w_i)\mathbb{E}(w_j) = p^3 - p^4$ . Let  $K$  denote the total number of pairs of triangles that share one edge in  $G$ . Considering that for each  $\text{cov}(w_i, w_j)$  there is an equal  $\text{cov}(w_j, w_i)$ ,  $\sum_{i \neq j} \text{cov}(w_i, w_j) = 8K(p^3 - p^4)$ . Therefore, we present the following lemma:

**Lemma 1.** *The variance of  $\hat{\Delta}^{E_G}$  is*

$$\text{var}(\hat{\Delta}^{E_G}) = \frac{1}{9p^4} (3\Delta(p^2 - p^4) + 8\Phi(p^3 - p^4)). \tag{8}$$

This result provides little insight into the accuracy of estimator, since it depends on a few parameters including  $p$ ,  $\Phi$ , and  $\Delta$ . We can transform it into relative standard error  $RSE = \sqrt{\text{var}}/\Delta$  as follows:

$$RSE(\hat{\Delta}^{E_G}) = \left[ \frac{1}{3\Delta|_g} \left( 1 - p^2 + \frac{8K}{3\Delta}(p - p^2) \right) \right]^{\frac{1}{2}}. \tag{9}$$

When the sample size is small, i.e., when  $p$  is small, we can see that the first term in Eq. 9 plays a dominant role. Hence, RSE of the estimator can be approximated by the following:

**Theorem 1.** *When the sample size is small, RSE of the  $E_G$  estimator can be approx-*

imated by

$$\widehat{RSE}(\widehat{\Delta}^{E_G}) \approx \frac{1}{\sqrt{3\Delta|_g}}. \quad (10)$$

This result is useful for the comparison with  $E_g$  method that will be discussed in the next section. In addition to that, it gives us a practical guidance for conducting estimations. For example, if we want to have an estimation with 95% confidence interval of  $\Delta \pm 0.1 \times \Delta$ , then we need to have an RSE that is approximately  $0.1/1.96 \approx 0.05$ . According to Eq. 10, the number of triangles we need to see is

$$\Delta|_g = \frac{1}{3 \times RSE^2} = \frac{1}{3 \times 0.05^2} = 133.$$

### 2.3.4 Variance of $E_g$

Although [7] gave the variance for the  $E_g$  estimator, it is a long formula that buries intuitive interpretations. Similar to our previous treatment for the  $E_G$  estimator, we transform the variance to RSE and simplified it into the following theorem:

**Theorem 2.** *When the sample size is small, RSE of the  $E_g$  estimator can be approximated by*

$$\widehat{RSE}(\widehat{\Delta}^{E_g}) \approx \frac{1}{\sqrt{\Delta_g}}. \quad (11)$$

*Proof.* Based on the variance of  $E_g$ , its RSE is as follows.

$$\begin{aligned} RSE(\widehat{\Delta}_{E_g}) &= \left[ \frac{1}{\Delta p^3} \left( 1 - p^3 + \frac{2\Phi}{\Delta} (p^2 - p^3) \right) \right]^{\frac{1}{2}} \\ &= \left[ \frac{1}{\Delta_g} \left( 1 - p^3 + \frac{2\Phi}{\Delta} (p^2 - p^3) \right) \right]^{\frac{1}{2}}. \end{aligned} \quad (12)$$

When a sampling probability  $p$  is small, the terms  $-p^3 + \frac{2\Phi}{\Delta}(p^2 - p^3)$  is neglectable. Thus, the RSE of  $E_g$  is estimated by  $\frac{1}{\sqrt{\Delta_g}}$ .

Next, we want to compare these two methods. One would be tempted to compare

their RSE ratio given a fixed sampling percentage. This approach turns out not ideal for two reasons: one is that given a fixed sampling probability,  $p$  is small for  $E_g$  could be already a very large one for  $E_G$ . Therefore it violates our small sample assumption.

Hence, we compare their sample size to achieve the same RSE. Comparing Eq.s 10 and 11, we obtained the performance ratio between the two methods:

**Corollary 1.** *Let  $n_{E_g}$  and  $n_{E_G}$  be the number of sample edges of  $E_g$  and  $E_G$  respectively for achieving the same RSE in the two methods. A relation between  $n_{E_g}$  and  $n_{E_G}$  is:*

$$\frac{n_{E_g}}{n_{E_G}} \approx \left[ \frac{3M}{n_{E_g}} \right]^{\frac{1}{2}} \quad (13)$$

*Proof.* Let  $p_{E_g}$  and  $p_{E_G}$  be sampling probabilities of  $E_g$  and  $E_G$ , respectively. We aim at getting the same RSE for both methods. Therefore, for small sample sizes the following equation holds

$$\begin{aligned} RSE(\widehat{\Delta}_{E_g}) &= RSE(\widehat{\Delta}_{E_G}) \\ \frac{1}{\sqrt{\Delta_g}} &= \frac{1}{\sqrt{3\Delta|_g}} \\ \Delta_g &= 3\Delta|_g. \end{aligned} \quad (14)$$

Since  $\Delta_g = \Delta p_{E_g}^3$  and  $\Delta|_g = \Delta p_{E_G}^2$ , we get

$$\begin{aligned} \Delta p_{E_g}^3 &= 3\Delta p_{E_G}^2 \\ p_{E_g}^3 &= 3p_{E_G}^2. \end{aligned} \quad (15)$$

By substituting  $p_{E_g} = \frac{n_{E_g}}{M}$  and  $p_{E_G} = \frac{n_{E_G}}{M}$ , in Eq. 15 the Corollary is proved.

Recall that  $M$  is the number of edges in  $G$ , which is always larger than sample size  $n_{E_G}$ . Therefore,  $E_g$  always needs more samples to achieve the same accuracy. When the sample size becomes bigger and approaches the total data size, the difference diminishes.

TABLE 2.2: Properties of the networks in our experiments, sorted by graph size  $N$ .

Dataset	$N(\times 10^6)$	$\langle d \rangle$	$\mathcal{C}$	$\Delta(\times 10^6)$	$\Phi(\times 10^9)$	$R(\Phi/\Delta)$	Description
Ego-facebook [61]	0.004	43.69	0.519	1.6	0.228	141.9	OSN
CA-GrQc [61]	0.005	5.52	0.629	0.04	0.002	42.3	Collaboration
Wiki-vote [61]	0.007	28.32	0.125	0.6	0.04	66.6	OSN
AstroPh [62]	0.01	21.10	0.31	1.3	0.072	53.5	Citation
CA-CondMat [61]	0.02	8.08	0.264	0.17	0.002	13.5	Coauthorship
HepPh [62]	0.02	224.14	0.279	195.7	92.9	474.6	Coauthorship
Enron-email [62]	0.03	10.02	0.085	0.72	0.03	50.2	E-communication
Brightkite [61]	0.05	7.35	0.110	0.49	0.029	59.1	OSN
Facebook [62]	0.06	25.64	0.147	3.5	0.155	44.5	OSN
Epinions [62]	0.07	10.69	0.065	1.6	0.112	69.2	OSN
Slashdot-Zoo [62]	0.07	11.82	0.023	0.53	0.028	52.5	OSN
Prosper [62]	0.08	74.60	0.003	1.1	0.062	54.1	Interaction
Livemocha [62]	0.1	42.13	0.014	3.3	0.183	54.4	OSN
Douban [62]	0.1	4.22	0.01	0.04	0.0001	4.7	OSN
Gowalla [61]	0.1	9.66	0.023	2.2	0.122	53.8	OSN
Libimseti [62]	0.2	155.97	0.007	69.1	19.2	278.3	OSN
Digg [62]	0.2	11.07	0.061	14.2	3.3	233.3	OSN
Web-Stanford [62]	0.2	14.13	0.008	11.3	9.06	800.4	Web graph
Dblp-Coau [61]	0.3	6.62	0.306	2.2	0.105	47.2	Coauthorship
Web-NotreDame [61]	0.3	6.69	0.087	8.9	1.5	174.2	Web graph
Amazon [61]	0.3	5.53	0.205	0.6	0.003	5.2	Co-purchasing
Actor [62]	0.3	78.68	0.166	346.8	91.0	262.3	Collaboration
Citeseer [62]	0.3	9.03	0.049	1.3	0.015	11.6	Citation
Dogster [62]	0.4	40.03	0.014	83.4	42.0	503.8	OSN
Catster [62]	0.6	50.32	0.028	656.3	1,017.4	1,550.0	OSN
Web-Berkeley [62]	0.6	19.40	0.0069	64.6	105.0	1,623.6	Web graph
Web-Google [62]	0.8	9.87	0.055	13.3	0.621	46.3	Web graph
Youtube [61]	1.1	5.27	0.006	3.0	0.251	82.3	OSN
Dblp [62]	1.3	8.16	0.170	12.1	0.436	35.8	Coauthorship
Hypes [62]	1.4	3.96	0.001	0.75	0.029	39.2	OSN
Wiki-Polish [62]	1.5	55.17	0.01	1,134.0	697.4	614.9	Web graph
Trec-wt10g [62]	1.6	8.33	0.014	21.0	11.0	523.2	Web graph
Wiki-Portuguese [62]	1.6	48.19	0.022	1,266.0	958.6	757.1	Web graph
Wiki-Japanese [62]	1.6	69.82	0.021	1,287.9	685.8	532.4	Web graph
Pokec [62]	1.6	27.31	0.046	32.5	0.724	22.2	OSN
As-skitter [61]	1.6	13.08	0.005	28.7	20.5	713.3	Internet topology
Wiki-Italian [62]	1.8	72.90	0.024	3,139.8	2,511.8	800.0	Web graph
Wiki-En [62]	1.8	39.05	0.003	126.6	39.4	311.4	Web graph
Hudong [62]	1.9	14.54	0.003	21.6	5.5	256.9	Web graph
Hollywood [63, 64]	1.9	24.51	0.152	204.7	27.2	132.9	OSN
Baidu [62]	2.1	15.89	0.002	25.2	4.7	186.8	Web graph
Flicker [62]	2.3	19.83	0.107	837.6	613.8	732.8	OSN
Flixster [62]	2.5	6.27	0.013	7.8	0.369	46.8	OSN
Wiki-Russian [62]	2.8	44.20	0.015	1,899.1	1,180.3	621.5	Web graph
Wiki-French [62]	3.0	55.21	0.015	2,281.1	4,237.8	1,857.7	Web graph
Orkut [62]	3.0	76.28	0.041	627.5	67.0	106.9	OSN
Wiki-German [62]	3.2	40.77	0.0088	966.5	560.1	579.5	Web graph
USpatent [62]	3.7	8.75	0.067	7.5	0.083	11.1	Citation
LiveJournal [61]	3.9	17.35	0.125	177.8	39.4	222.0	OSN
Orkut2 [63, 64]	11	56.80	0.0002	223.1	34.6	155.3	OSN
DBpedia [62]	18	13.89	0.0016	328.7	107.0	325.7	Web graph
Web-Arabic [63, 64]	22	48.70	0.031	36,895.3	112,260.9	3,042.6	Web graph
Gsh-2015 [63, 64]	29	9.18	0.007	389.7	103.1	264.7	Web graph
Twitter [62]	41	57.74	0.0008	34,824.9	176,266.1	5,061.4	OSN
MicrosoftAc.G. [65]	46	22.61	0.015	578.1	19.5	33.8	Citation
Friendster [62]	65	55.06	0.017	4,173.7	185.1	44.3	OSN



## 2.4 Experiments

Our analytical results are derived with approximations based on the assumption on the data size and sample size. The experiments are designed to confirm the validity of the analytical results, and empirically demonstrate how much better our method is. In particular, analytical results do not include the cost of additional closeness check. These experiments confirm that such cost does not affect our overall result.

### 2.4.1 Datasets

We used 56 real world graphs to evaluate the algorithms, whose statistics are summarized in Table 2.2. We removed repeated edges and self-loops, and ignored the edge directionality in directed networks. Therefore, some statistics may be different from other papers working on the same datasets. For example, we found that the Twitter contains 18% repeated edges. Such repeated edges have to be removed to guarantee the accuracy of the sampling.

We included almost all the largest graphs that we could find. Examples are the most recent academic citation graph released by Microsoft, which contains 46 million nodes, and the well-known Twitter user network that has 41 million nodes. In addition to these large graphs, we also included some smaller graphs of various scales for comparison. The types of the graphs are also diversified, covering various areas. There are web graphs, online social networks, citation graphs, co-author and co-purchasing relations etc.

The experiments are conducted on two servers with 256GB memory and 24 cores each. The data and code are available on the website <http://etemadir.myweb.cs.uwindsor.ca/cikm2016/triangles.php>.

### 2.4.2 Experimental setup

We verify Theorems 1 and 2 by comparing observed RSEs obtained from running the estimators on the datasets and expected RSEs derived from Eq.s 10 and 11. To

obtain observed RSE, we repeat the estimation  $k$  times using the same sample size, each time obtain an estimate  $\Delta_i$ . Let  $\mu = \frac{1}{k} \sum_{i=1}^k \Delta_i$ . The observed RSE is calculated using

$$RSE = \frac{1}{\Delta} \sqrt{\frac{1}{k} \sum (\Delta_i - \mu)^2}. \quad (16)$$

In our experiments,  $k = 1000$  for all graphs.

For the sample size parameter, most existing methods, such as Doulion [7], use a fixed range of sampling probability  $p$  or percentages of the edges sampled for all graphs. Fixed percentage creates a wide variation for RSEs: one percent of sample data may not be enough to have an accurate estimate for small graphs, but can achieve very good (small) RSE for large graphs. Instead of fixed percentage, we target at a fixed range of RSEs, and choose the sample size that can create the RSEs at the desired range. RSE can reflect the confidence interval of the estimates, which is the main concern of any estimator.

All the experiments target RSEs between the range of 0.05 and 0.4. Next, we need to select sample size  $n$  so that the observed RSE would be in that range. Recall that from Eq.s 10 and 11 we can derive  $\Delta_g$  and  $\Delta|_g$  from desired RSEs. However, we still do not know what is the sample size  $n$  to obtain that number of triangles. We derived the following theorem to decide the sample size for  $E_G$ .

**Theorem 3.** *In  $E_G$  sampling, the relationship between  $n_{E_G}$  and  $\Delta|_g$  is*

$$n_{E_G} \approx \left[ \frac{3N\Delta|_g}{2\mathcal{C}\Gamma} \right]^{\frac{1}{2}}, \quad (17)$$

where  $\mathcal{C}$  is the clustering coefficient, and  $\Gamma = CV^2 + 1$ , where  $CV$  is the coefficient of degree variation.

*Proof.* Based on the estimator  $\hat{N}$  for graph node size [11], we obtain the relation

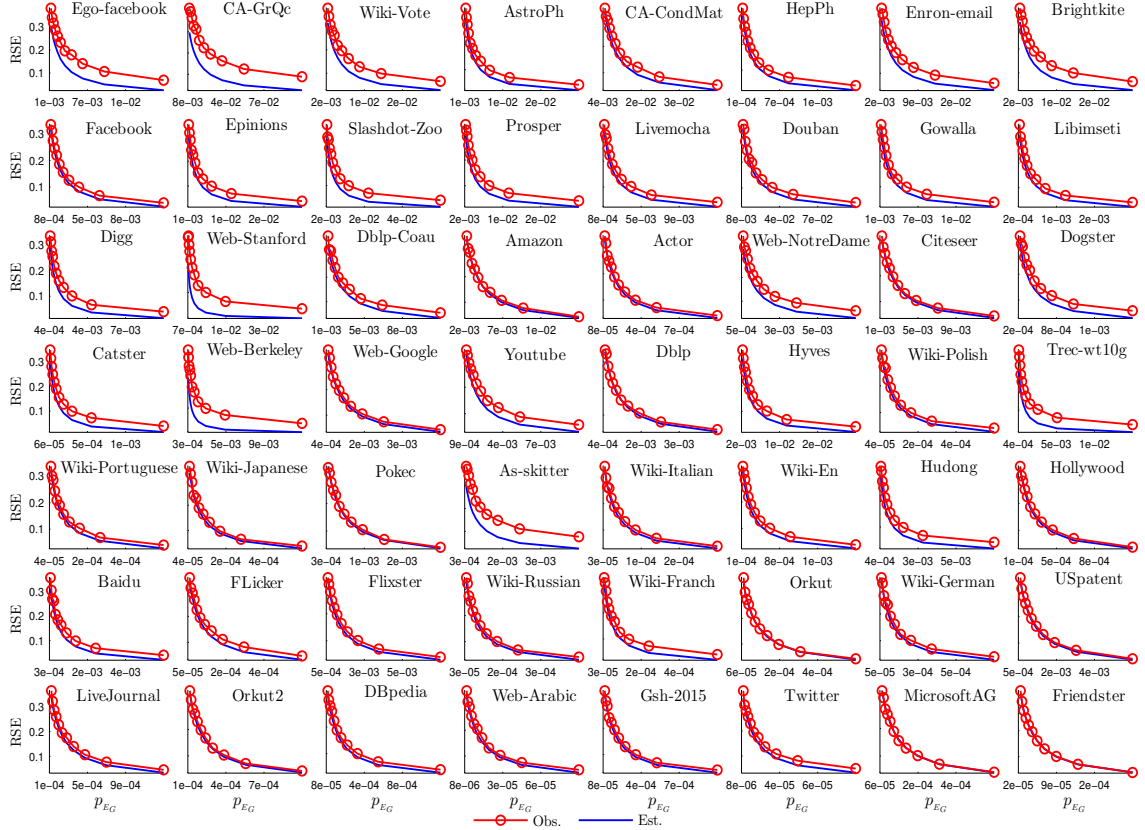


FIGURE 2.3: The observed and estimated RSEs of estimator  $E_G$ . Estimated values are obtained from Eq. 10.

between  $N$  and the sample subgraph  $g$  as follows:

$$N = \frac{1}{w_g} \binom{n}{2} \Gamma. \quad (18)$$

Here,  $n = 2 \times n_{E_g}$  is the total number of times the nodes have been sampled. It is the sum of all the degrees of the sample graph  $g$ , or number of edges times two.  $w_g$  is the number of collisions in [11], which can be interpreted as the number wedges in  $g$ . After rearranging the above formula, we have:

$$n^2 - n = \frac{2Nw_g}{\Gamma}. \quad (19)$$

Recall that  $w_g = \frac{3\Delta|g}{C}$ , and approximate  $n - 1$  with  $n$ , we get:

$$n \approx \left[ \frac{6N\Delta|g}{\mathcal{C}\Gamma} \right]^{\frac{1}{2}}. \quad (20)$$

The approximation  $n \approx n - 1$  can be applied because  $n$  is in the order of  $\sqrt{N}$  so that collisions (wedges) can be observed in the sample graph. When  $N$  is large, say in the order of  $10^6$ ,  $n$  is in the order of  $10^3$ . Substituting  $n$  with  $2 \times n_{EG}$ , we prove the theorem.

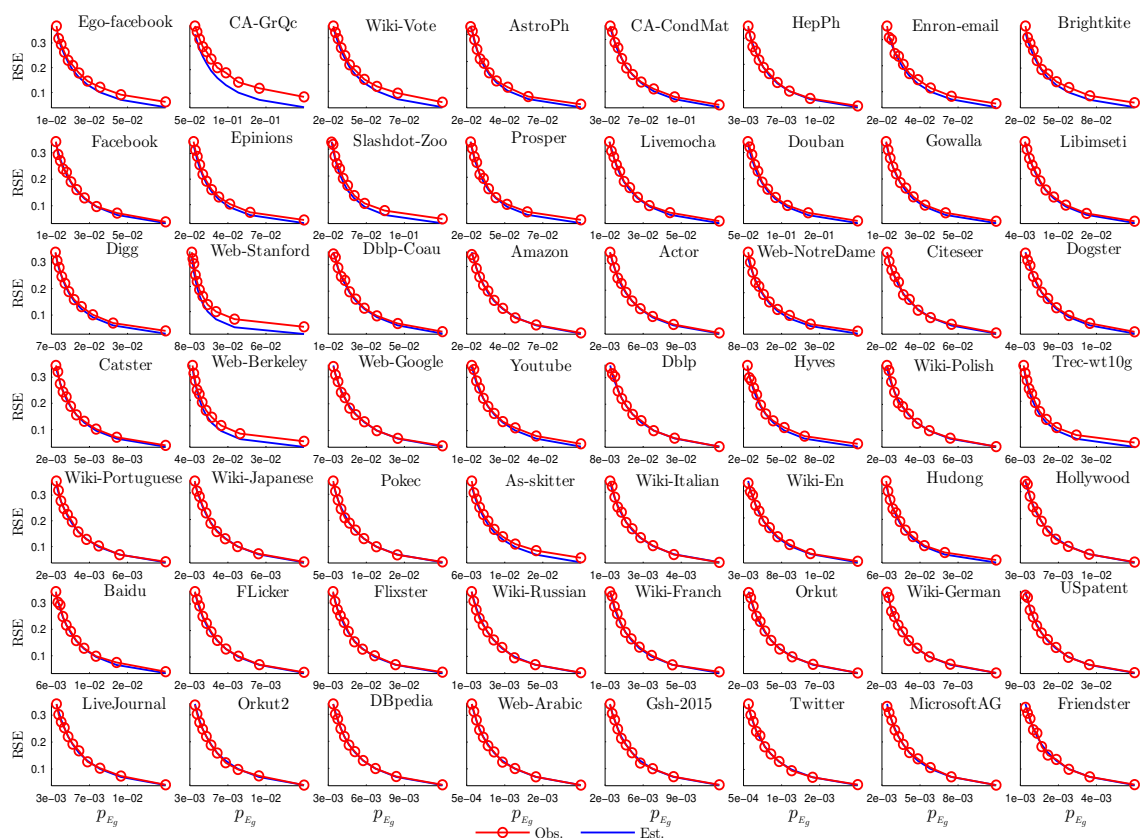


FIGURE 2.4: The observed and estimated RSEs of estimator  $E_g$ . Estimated values are based on Eq. 11.

We want to emphasize that we do not need to know those parameters such as  $\mathcal{C}$ ,  $\Gamma$ ,  $N$  to estimate the number of triangles. Eq. 17 is used only in our experiment to select the sample size so that we know the results are within a certain RSE range. The triangle estimation itself only needs to know the sampling percentage and  $\Delta|g$ .

Under certain circumstances, such as in sampling hidden data sources, the sampling percentage  $p$  is not known since it can not be derived from  $n_{E_G}$ . We do know  $n_{E_G}$ , but  $p = n_{E_G}/M$  depends on  $M$ , the number of edges in the graph. Note that  $M = N \times \langle d \rangle$ . Both number of nodes  $N$  and average degree  $\langle d \rangle$  can be estimated effectively using random edge sampling. We refer to [11] for the estimation of  $N$ , and [66] for the estimation of average degree.

### 2.4.3 Verification of Theorems

Due to the approximations made in our derivations, we need to show the impacts of those approximations for different data sets.

**Theorem 1.** The observed RSEs and the projected RSEs are plotted in Fig. 2.3 for the  $E_G$  method. We can make several observations:

- For all the graphs, Eq. 10 is a good approximation for the real RSE observed. The approximation is lower than the actual value, because we omitted the remaining terms in Eq. 9;
- the approximation is more accurate when the data is large. Recall that the datasets are sorted in increasing order of data size;
- among the large graphs, Twitter and Web-Arabic demonstrate large deviation than other large graphs. A closer check reveals that they both have very large  $\Phi$ 's, and large ratios between  $\Phi$  and  $\Delta$ . According to Eq. 9, the ratio  $\Phi/\Delta$  in the third term plays a key role on the impact of the approximation. Note that  $\Phi$  can be even larger than  $\Delta$ , because it is the number of combinations of triangles that share a common edge.

**Theorem 2.** Fig. 2.4 shows the observed RSEs vs the approximated values derived from Theorem 2. Overall the approximation fits better with the real data than the  $E_G$  method. This is expected because the major term we omitted in Eq. 11

is smaller. It is

$$\frac{2\Phi}{\Delta}p^2, \tag{21}$$

while the term omitted in Eq. 10 is

$$\frac{8\Phi}{3\Delta}p \tag{22}$$

**Corollary.** Fig. 2.5 demonstrates the major result of this work: to what extent our method improves the random edge sampling method. In the experiment, we included the cost for checking wedge closures. Overall, our projected improvement ratio fits well with the observed data. Again, in large graphs the projection fits better with the real data. Two additional observations are:

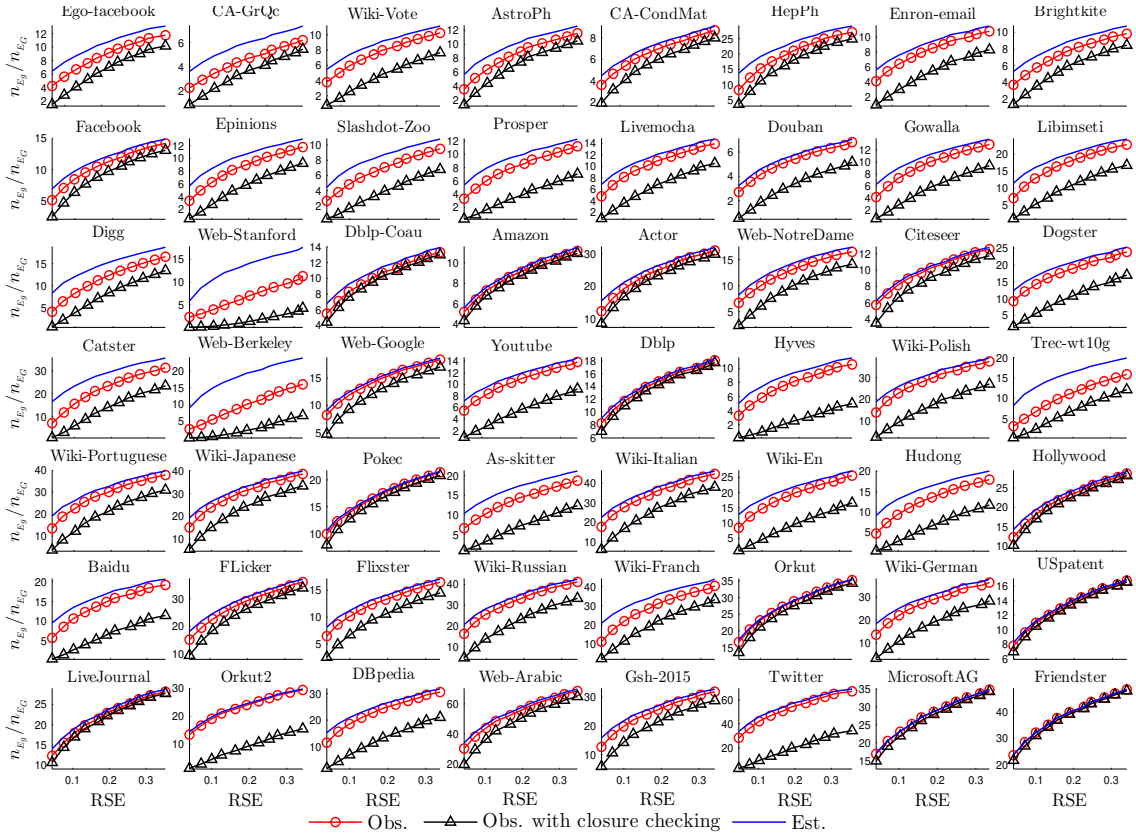


FIGURE 2.5: The observed and estimated ratio between the sample sizes of estimators  $E_g$  and  $E_G$  with the same RSE. Estimated value are obtained based on Eq. 13.

- The advantage of our method grows with the RSE (or decrease with the desired accuracy level). Our method is good when the sample size is small. When a large portion of the data is sampled, our method may not be as good as the random edge method. Recall that in all the derivations, we assumed that the sampling probability  $p$  is small. Despite such assumption, our method is consistently better than  $E_g$  in all datasets. RSE ranges from 0.05 to 0.4. 0.05 is a reasonable RSE from which we can obtain 95% confidence interval between  $\Delta \pm 0.1 \times \Delta$ .
- The improvement grows with data size. When RSE=0.05, it improves by a factor of four for Facebook, and a factor of 30 for Twitter. Fig. 2.6 plots the improvement as a function of data size. It can be seen that the sample size ratio is correlated positively with the data size. The Pearson correlation coefficient is 0.88 for edge size, and 0.94 for triangle size, when both the data size and the improvement ratio are in logarithmic scale. The unlogged correlation coefficient is 0.64 for edge size and 0.69 for triangle size. We can see that the improvement correlates with  $\Delta$  more stronger than  $M$ . This demonstrates that our method is especially good for large graphs with a large number of triangles.

## 2.5 Discussions and conclusions

We propose a triangle estimation method that outperforms the previous one by a factor of up to 30 when RSE is 0.05. The improvement can be higher when RSE is bigger, or the required accuracy is reduced. We proved that the estimator is unbiased, and derived its variance. The variance in the original form lacks intuitive interpretation due to the long formula and multiple variables involved. Based on the big graph (henceforth small sample) assumption, we simplified the RSE to  $(3\Delta|g|)^{-1/2}$ . Such simplified result gives us practical guidance in sampling. We can derive the confidence interval by looking at the triangles observed in the sample, hence we can decide when to stop the sampling. Although several assumptions are made for our

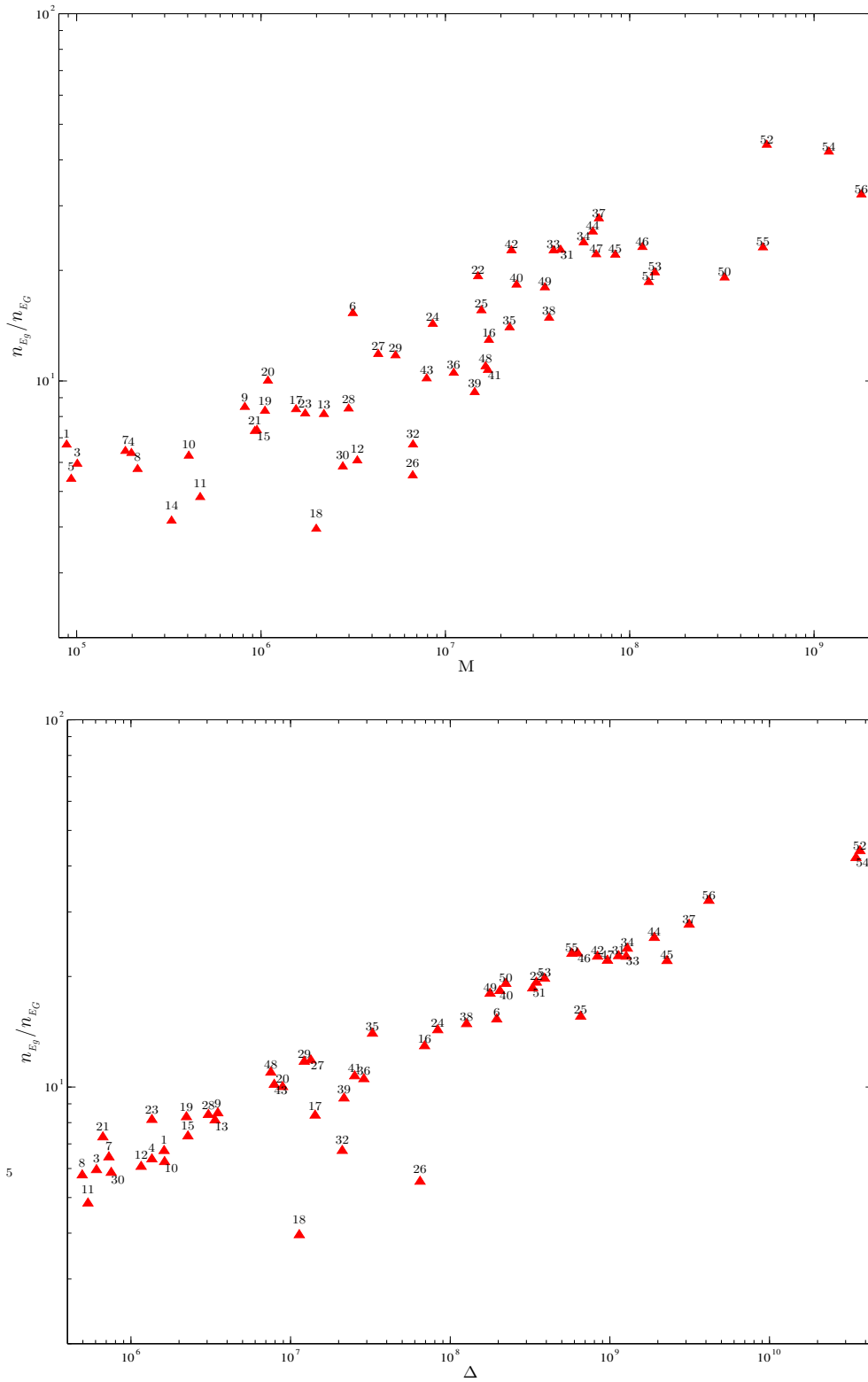


FIGURE 2.6: Improvement as a function of data size  $M$  and  $\Delta$ . RSE=0.1.



conclusions, we empirically showed that our approximation is still very close to the real RSEs observed.

The simplified formula also allows us to compare our method with other methods analytically. In the past, performances for different methods, including various streaming algorithms for triangle counting, are compared empirically. Those results may vary from data to data. Our work is the first to study the performance ratio analytically.

Our method is particularly suitable for very large graphs. It reduces the sample size by orders of magnitude for large graphs. We showed that the performance improves positively correlates with data size. When  $\Delta$  and improvement ratio are logged, their Pearson correlation coefficient is as high as 0.94, almost a linear function for all 56 data sets.

The method is motivated by the scarcity of triangles in sampled graph when the original graph is very large. We can increase the probability of observing more triangles by checking the wedges in the sample graph. This strategy works very well for several reasons: First, most networks tend to cluster together, as friend's of friend's have a tendency to be friends as well. Thus, when checking the closeness of a wedge, it has a high probability that the wedge is closed; Secondly, checking the closeness of a wedge is more efficient than throwing a random edge in identifying a triangle. Throwing a random edge in a very large graph may well end up with an isolated edge, not even connecting with any other edge, let alone forming a triangle. On the other hand, checking the closeness of a wedge works at least in the vicinity of two connected edges. Its chance is higher to form a triangle when the sample size is small.

# **PES: Priority Edge Sampling in Streaming Triangle Estimation**

---

---

## **3.1 Introduction**

Sampling-based algorithms are especially important in the era of big and hidden data. There are numerous massive networks that have billions of nodes. For example, Facebook as an online social network has over two billion users. Many networks are dynamic, both users and connections between users can change over time. Furthermore, networks are often hidden behind access interfaces, and data in its entirety are not available. Therefore, it is essential to design sampling-based methods.

There are two types of methods that estimate triangles and the closely related metric clustering coefficient. One is the direct-sampling that has random access to the nodes/edges of the input graph [7, 28, 58]. The other is the streaming model that scans the nodes/edges of the input graph in an arbitrary order over a stream. In the streaming model, a constant number of passes over the stream are used to estimate  $\Delta$ . The key constraint in streaming models is a limited memory window [1, 8–10, 17, 29]. When there is no limit to the number of passes, it is called a semi-streaming model [67]. This work addresses the estimation of  $\Delta$  in the streaming model.

This research proposes a new streaming algorithm, called PES (Priority Edge Sampling). It is based on edge sampling and gives higher priority to edges that can form triangles. We prove that our estimator is unbiased, and derive the variance of the estimator so that the confidence interval can be obtained when an estimation is given. Empirically, we compare it with the state-of-the-art GPS-In algorithm [1], and demonstrate that PES outperforms GPS-In consistently on most of the 48 real

networks that we have experimented with. More importantly, the performance gain increases with the size of networks. The performance ratio can be as high as 11, meaning that GPS-In needs 11 times more samples to achieve the same accuracy.

Performances of sampling algorithms are often data dependent, especially on the structure of the graphs. To verify our result in addition to empirical comparisons, we conduct analytical comparisons. GPS-In cannot give an analytical variance of the estimation because its sampling probability changes in every step. Hence, the comparison between PES and GPS-In cannot be analytical. To understand the advantage of PES, we compare it with NES (Naive Edge Streaming) that was proposed in [28] [19] [20]. The analytical comparison between PES and NES can shed some lights on understanding the difference between PES and GPS-In.

To summarize, our main contributions are that we have: 1) Given an efficient algorithm PES. 2) Proved the unbiasedness of the estimator and derived variances for PES and NES; 3) Compared PES and NES analytically.

## 3.2 Background and related work

Given a simple graph  $G(\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V}$  stands for the set of nodes, and  $\mathcal{E}$  the set of edges. Let  $N = |\mathcal{V}|$ ,  $M = |\mathcal{E}|$ ;  $\Delta$  and  $\Lambda$  denote the number of triangles and wedges in  $G$ , respectively. Suppose that  $1, 2, \dots, M$  be the labels of the edges in  $\mathcal{E}$  according to their arrival times in a stream. E.g., edge 1 arrives at time 1 and so on. A *wedge*  $\mathcal{W}$  is a path  $(u, v, w)$  of length two, where  $u, v, w \in \mathcal{V}$ ,  $(u, v) \in \mathcal{E}$ , and  $(v, w) \in \mathcal{E}$ . The wedge  $\mathcal{W}$  is closed if  $(u, w) \in \mathcal{E}$ . Otherwise it is open. A closed wedge  $\mathcal{W}$  is also called a triangle. Note that each triangle has three closed wedges. Table 3.1 summarizes the list of the notations used in the rest of this chapter.

Each sampling method takes some sample nodes or edges, or a combination of them, into a subgraph. Then, the number of triangles in the subgraph is used to estimate the triangle count. Depending on the way to take samples, the estimator and its variance change. Intuitively, we want to observe the maximum number of triangles while keeping the sample size small. The bottom line is that we need to

TABLE 3.1: Summary of the notations

Notation	Meaning
$G(\mathcal{V}, \mathcal{E})$	Input graph (undirected and no self-loops)
$g$	A subgraph of $G$
$N, M$	Number of nodes and edges in $G$
$p, q$	Sampling probability
$\Lambda$	# wedges in $G$
$\Delta$	# triangles in $G$
$\sigma$	A wedge pool
$n$	Size of pool $\sigma$
$m$	Sample size
$\Delta_\sigma$	# triangles based on pool $\sigma$ .
$\Lambda_c$	# candidate wedges identified based on $g$
$\Delta_g$	# triangles based on $g$ .
$\Phi$	# pairs of shared triangles in $G$
$\widehat{\Delta}_{NES}$	Naive edge streaming estimator
$\widehat{\Delta}_{PES}$	Priority edge sampling estimator

observe at least one triangle in order to give an estimate.

### 3.2.1 Node-based methods

The most naive method of triangle estimation is to sample three random nodes as a potential triangle, then check the existence of edges among the nodes over a stream. It is called *triple sampling* [9]. This approach needs to sample  $4N^3/(6\epsilon^2\Delta)$  number of triplets to achieve an estimation in interval  $\Delta \pm \epsilon\Delta$  with %95 confidence. Intuitively, the complexity of three nodes combination is  $O(N^3)$ . Obviously, it is not a practical method because the sample size is too large to observe even one triangle. The cost is even higher than direct counting of the triangles.

A more practical method is to sample a random edge and a random node [10], then check whether they form a triangle. This method improves the previous triple sampling by assuming one edge always exists in the triplet. Hence, it only needs to check the existence of the other two edges. Still, it needs to take  $4MN/(3\epsilon^2\Delta)$  triplets to have an estimation in the same confidence interval as in triple sampling.

Large real networks are mostly sparse, hence the probability of having a triangle is still low among two random pairs of nodes. One improvement to the above method

is, instead of choosing a random node in the entire graph, selecting a random node from its neighbourhood. It is called *neighborhood sampling* [17, 18].

### 3.2.2 Edge-based methods

The most straightforward edge sampling is to take edges uniformly at random, then count the triangles in the subgraph [7]. In the streaming model, the corresponding streaming version of the algorithm is to take each edge with an equal probability  $p$  over a stream and create a subgraph  $g$ . The number of triangles in  $g$  is used to estimate  $\Delta$ . Obviously, the sampling probability of a triangle in such method is  $p^3$ . The size of  $g$  needs to be  $1.5M/(\epsilon^2\Delta)^{1/3}$  to obtain an estimation with an additive error  $\pm\epsilon\Delta$  with %95 confidence. When  $p$  is small, which is the case for very large graphs, this algorithm is not efficient.

Instead of using equal probability among three edges, there are methods to assign high probabilities for the second and/or the third edge. For instance, *post-stream priority sampling* (GPS-Post) [1] takes this approach by sampling the third edge with a higher probability.

Another technique is to take edges from the neighborhood of already sampled edges with higher probability. A pair of connected edges (called a wedge) in the sample can be a potential triangle, and its closeness is checked in the rest of the stream [19–21]. Obviously, the probability of forming a wedge is  $p^2$  because its two edges are required to be sampled. [8] improves the previous method as follows. When an edge closes a wedge in a sample it is unconditionally added into the sample; if it is connected to some sampled edges it is chosen with higher probability  $q$ ; otherwise it is taken with probability  $p$ . The number of triangles in the sample is used to estimate  $\Delta$ . Obviously, this method samples triangles with different probabilities, i.e.,  $pq$ ,  $q^2$ ,  $p$ ,  $q$ , and 1. One shortcoming of this approach is that how one can determine  $q$  - sampling probability of a neighbor edge. To overcome such an issue, in our method  $q$  is dynamically adjusted using *reservoir sampling* [68].

More recently, another elegant approach has been proposed by [1] called *in-stream priority sampling* (GPS-In). It preserves edges in a sample with different priorities.

The number of sampled wedges closed by an edge is used as a measure to determine the priority of the edge being preserved in the sample. For each new edge  $e$ , it first counts the number of wedges closed by  $e$  in the sample and computes its priority. Then, the edge is added into the sample. If the number of edges in the sample exceeds the size limit, an edge with lower priority is removed from the sample. In each step, the estimator for  $\Delta$  is updated if edge  $e$  completes some wedges in the sample. It has been shown that GPS-In outperforms the existing methods [1]. Therefore, we consider GPS-In as the state-of-the-art method in this context.

When random access to the input graph is available the ideal method is *wedge sampling*. It selects some wedges uniformly at random and checks their closeness to estimate  $\Delta$ . Unfortunately, taking a wedge uniformly at random in a large graph is costly. Three passes over an edge stream are required to implement wedge sampling in the streaming model [9, 10, 17].

Another direction is indirect sampling. Such methods have been applied when the entire graph is not accessible. They use traversal-based sampling techniques to take a sample from the input graph [23, 27]. Moreover, several works have been conducted to compute clustering coefficient closely related to  $\Delta$  [6, 23, 30, 31].

### 3.3 Naive edge streaming (NES)

As a starting point for understanding our PES algorithm which is described in the next section, we first present a naive algorithm based on edge sampling, called NES (Naive Edge Streaming). It is similar to TRIEST [19] and MASCOT [20]. The details of NES are shown in Alg. 1. For each edge in a stream, NES adds the edge into subgraph  $g$  with probability  $p$  (Line 4). Then, the same edge is used to check how many wedges in current  $g$  are closed by it.  $\Delta_g$  records the sum of such closed wedges(triangles) (Lines 5-7).

The algorithm differs from the one in [28] in that we do not count the triangles in  $g$ . Instead, it checks the closeness of wedges in  $g$  during the streaming process. Clearly, the probability of forming a wedge in  $g$  is  $p^2$ . Note that three edges of a

closed wedge can appear in six different orders in a stream. In two of them, the third edge appears after the first two and the associated closed wedge can be observed. Thus, the probability of identifying a closed wedge is  $p^2/3$ . Because each triangle has three closed wedges, the sampling probability of each triangle is  $p^2/3 \times 3 = p^2$ . Note that each identified closed wedge by NES is considered as a one triangle because only one of three closed wedges of each triangle can be identified in a stream.

Suppose  $\delta_i$  be an indicator for the  $i^{th}$  triangle in the original graph  $G$ . Indicator  $\delta_i$  is one when the  $i^{th}$  triangle is identified over the stream; otherwise it is zero. Recall that  $\Delta_g$  is the number of triangles identified by NES based on  $g$  over a stream. The expectation of  $\Delta_g$  is

$$\mathbb{E}(\Delta_g) = \mathbb{E}\left(\sum_{i=1}^{\Delta} \delta_i\right) = \sum_{i=1}^{\Delta} \mathbb{E}(\delta_i) = \sum_{i=1}^{\Delta} p^2 = p^2 \Delta. \quad (1)$$

Thus, the unbiased estimator for  $\Delta$  using NES is

$$\hat{\Delta}_{NES} = \frac{\Delta_g}{p^2}. \quad (2)$$

---

**Algorithm 1:** Naive Edge Streaming (NES)
 

---

**Input:**  $p$   
**Output:**  $\hat{\Delta}$ ,  $\text{RSE}(\hat{\Delta})$

```

1 begin
2    $\Delta_g = 0, g = \phi.$ 
3   while new edge e do
4     Add  $e$  into  $g$  with probability  $p.$ 
5     foreach wedge  $w \in g$  closed by  $e$  do
6        $\Delta_g + = 1.$ 
7     end
8   end
9    $\hat{\Delta}_{NES} = \Delta_g / p^2.$ 
10   $\text{RSE}(\hat{\Delta}_{NES}) \approx \Delta_g^{-1/2}.$ 
11 end
    
```

---

Next we need to understand the variance of  $\hat{\Delta}_{NES}$ . Although MASCOT gave a similar algorithm, they only give upper-bound of its variance. We derived the variance of  $\hat{\Delta}_{NES}$  and present it in the form of Relative Standard Error ( $\text{RSE} = \sqrt{\text{var}}/\Delta$ ) in

Theorem 1. We use RSE instead of variance that is commonly used. This is because variance depends on the ground truth, which changes from data to data. This is especially inconvenient when evaluating multiple data sets—a larger variance in one data may be better than a smaller variance in another data. The variance of NES is adapted but different from the direct sampling algorithm in [28] to accommodate the streaming model. The main difference is that in NES, to identify a closed wedge over a stream, first its two edges need to be added into  $g$ ; then its third edge needs to be visited in the rest of the stream.

**Theorem 1.** *The RSE of  $\widehat{\Delta}_{NES}$  is estimated by*

$$\widehat{RSE}(\widehat{\Delta}_{NES}) \approx \Delta_g^{-1/2}. \quad (3)$$

*Proof.* Let  $\Delta_g$  be the number of triangles identified by the NES method based on wedges in  $g$ . Let  $\delta_i$  be an indicator for  $i^{\text{th}}$  triangle in the original graph.  $\delta_i$  is 1 when the  $i^{\text{th}}$  triangle is identified; otherwise it is 0. Applying the variance on  $\widehat{\Delta}_{NES}$  we get:

$$\text{var}(\widehat{\Delta}_{NES}) = \text{var}\left(\frac{\Delta_g}{p^2}\right) = \frac{1}{p^4} \text{var}\left(\sum_{i=1}^{\Delta} \delta_i\right) \quad (4)$$

$$= \frac{1}{p^4} \left( \sum_{i=1}^{\Delta} \text{var}(\delta_i) + \sum_{i \neq j} \text{cov}(\delta_i, \delta_j) \right). \quad (5)$$

The probability of identifying triangle  $\delta_i$  by NES is  $p^2$ . Hence, the variance of  $\delta_i$  is  $\mathbb{E}(\delta_i) - \mathbb{E}^2(\delta_i) = p^2 - p^4$ . Therefore the cost of the first term in Eq. 5 is  $\Delta(p^2 - p^4)$ . Clearly, the covariance of two independent triangles is zero. Thus, we need to consider the covariance of dependent pairs. Each pair of shared triangles has five edges. To identify such a pair by NES, a shared edge and one of the other two edges of each triangle need to be sampled with probability  $p$ . Therefore, a chance of identifying such a pair is  $p^3$ . Obviously, the five edges of a pair of shared triangles appears in 120 different orders in a stream; in 64 out of 120 cases, such a pair can be identified. Clearly, the cost of covariance of two shared triangles is  $p^3 - p^4$ . Recall that the total number of pairs of shared triangles is denoted by  $\Phi$ . Therefore, the cost of covariance



term is  $\frac{16}{15}\Phi(p^3 - p^4)$ . Thus, by adding all the costs we have

$$\text{var}(\widehat{\Delta}_{NES}) = \frac{\Delta(1 - p^2)}{p^2} + \frac{16\Phi(1 - p)}{15p}. \quad (6)$$

Translate Eq. 6 into RSE, we got

$$RSE(\widehat{\Delta}_{NES}) \approx \left[ \frac{1}{\Delta p^2}(1 - p) + \frac{16\Phi}{15\Delta^2 p^2}(p - p^2) \right]^{1/2}.$$

When sampling probability  $p$  is small, terms  $-p$  and  $\frac{16\Phi}{15\Delta}(p - p^2)$  are neglectable. Therefore, the RSE is  $(\Delta p^2)^{-1/2}$ . Replace  $\Delta p^2$  by  $\Delta_g$ , we prove the theorem.

Theorem 1 shows that the variance depends on the number of triangles in the sampled graph  $g$ . To reduce RSE with the same subgraph size  $g$ , we need to sample more triangles while keeping the same sampling probability for the first edge. This prompts us to increase the sampling probability for the second edge of a triangle.

## 3.4 Priority edge sampling (PES)

### 3.4.1 The algorithm

PES improves NES by increasing the probability of capturing triangles in the sample graph. To do so, we maintain a pool of wedges as well as a subgraph  $g$ . Edges that can form a wedge in  $g$  will have a higher priority being sampled. Hence, we call it Priority Edge Sampling. It is impossible and not necessary to keep all the wedges. Instead, we maintain a small fixed-size pool of wedges  $\sigma$ . For each triangle, the first edge will be sampled with probability  $p$ , which is the same as NES. The difference is in the second edge. When the second edge is scanned, the associated wedges are added into  $\sigma$  with probability  $q$ . Later we will show that  $q$  is normally much larger than  $p$ , especially when the graph is large. The closeness of wedges in the pool is checked in the rest of the stream. Therefore, PES identifies a triangle with probability  $pq$ , which is greater than  $p^2$  in NES.

The details of PES are summarized in Alg. 2. Input  $p$  is the sampling probability of edges,  $n$  is the pool size. In our experiments, we simply set  $n = |g|$  for the convenience of performance comparison.  $\Lambda_c$  counts the wedges formed in  $g$ . Some of these wedges may be added to  $\sigma$  with a changing probability  $q$ . Hence we call them *candidate* wedges and denoted by  $\Lambda_c$ .  $\Delta_\sigma$  counts the triangles formed from  $\sigma$  and  $g$ . When a new edge  $e$  is visited, it is added into subgraph  $g$  with probability  $p$  (Line 4). Then, the closeness of wedges in pool  $\sigma$  is checked (Lines 5-8). Once a closed wedge is identified, the number of triangles  $\Delta_\sigma$  captured so far is increased by 1 (Line 7). Next, each candidate wedge formed using the new edge  $e$  and edge  $f$  in  $g$  like  $w(e, f)$  is considered to be added into pool  $\sigma$  with probability  $q$  (Lines 9-21). Note that probability  $q$  is dynamically computed over the stream using  $n$  and  $\Lambda_c$  (Line 14). We explain the steps in the following illustrative example.

### 3.4.2 Example

We illustrate PES with a toy graph in Fig. 3.1 with detailed steps. Each row in the table represents one step. Column  $e$  shows the edge stream. Column  $g$  displays the sampled edges in subgraph  $g$ . In this example, each edge in the stream is added into  $g$  with probability  $p = 0.2$ . When edge  $(1, 4)$  arrives, PES adds it to  $g$  with probability  $p$ . Suppose that it is not added, and  $g$  remains empty. Next edge in the stream is  $(6, 8)$ . Suppose that it is added to  $g$  this time. It can not form any wedges in the fourth column.

The third edge  $(6, 7)$  is not added into  $g$ , but we still check its neighbours in  $g$  for closed wedges and *candidate* wedges. The *candidate* wedges constructed in each step are demonstrated in the fourth column. When edge  $(6, 7)$  is encountered in step 3, a wedge  $(7, 6, 8)$  is formed since edge  $(6, 8)$  is already in the subgraph  $g$ . In the pool for each wedge, we keep a label to show its closeness. The open wedge  $(7, 6, 8)$  is denoted as  $(7, 6, 8)^-$ . Column  $\Lambda_c$  records the number of such candidate wedges. It can be larger than the pool size. When edge  $(6, 11)$  arrives, it forms a candidate wedge  $(8, 6, 11)$ , hence  $\Lambda_c$  is increased by one, but it is not added into the pool  $\sigma$ .

Not every candidate wedge is added into the pool. The pool has a fixed size,

---

**Algorithm 2:** Priority Edge Sampling (PES)
 

---

**Input:**  $p, n$ .  
**Output:**  $\hat{\Delta}$ ,  $\text{RSE}(\hat{\Delta})$

```

1 begin
2    $\Lambda_c = 0, \Delta_\sigma = 0, \sigma = \phi, g = \phi$ .
3   while new edge  $e$  do
4     Add edge  $e$  into  $g$  with probability  $p$ ;
5     foreach wedge  $w$  in  $\sigma$  closed by  $e$  do
6       label  $w$  as closed.
7        $\Delta_\sigma + = 1$ .
8     end
9     foreach wedge  $w(e, f)$  where edge  $f \in g$  do
10       $\Lambda_c + = 1$ .
11      if  $|\sigma| < n$  then
12         $\sigma = \sigma \cup \{w\}$ .
13      else
14         $q = n/\Lambda_c$ .
15        if  $\text{Random}[0,1) < q$  then
16          Select random wedge  $w'$  from  $\sigma$ .
17          if  $w'$  is closed then  $\Delta_\sigma - = 1$ .
18           $\sigma = \sigma - \{w'\}$ .
19           $\sigma = \sigma \cup \{w\}$ .
20        end
21      end
22    end
23     $\hat{\Delta}_{PES} = \Delta_\sigma/pq$ .
24     $\text{RSE}(\hat{\Delta}_{PES}) \approx \Delta_\sigma^{-1/2}$ .
25 end
    
```

---

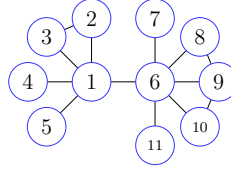
functioning as a reservoir. In this example, its capacity is  $n = 2$ . The candidate wedge is added into the pool unconditionally only when it is not full yet. Hence, wedge (7, 6, 8) and the wedge in the subsequent step (1, 6, 8) are added into the pool.

When the pool is full, the candidate wedge will replace a random wedge in the pool with probability  $q$ . In step 9, edge (6,10) forms a candidate wedge (8,6,10) with edge (6,8). Now the forth wedge (8,6,10) can not be added into  $\sigma$  directly because the pool has reached its limit 2. Instead, we replace one of the wedges in the pool with a probability  $q = n/\Lambda_c = 2/4$ . Suppose that by chance, this wedge replaces (7,6,8) in the pool. The candidate wedge in Step 10 does not replace any wedge in the pool by chance. For the candidate wedge (9,6,8) in step 11, suppose that it replaces an

existing wedge (1,6,8) in the pool. Step 12 has another wedge being replaced.

The last edge in the stream is (8,9). It closes the wedge (9,6,8)<sup>-</sup> that is obtained in previous steps. Hence, the label of this wedge is changed to +; and  $\Delta_\sigma$  is increased by 1. At this point,  $\Lambda_g = 8$ . This means eight candidate wedges are identified in total over the stream; the probability of preserving a wedge in  $\sigma$  is  $q = 2/8$ . Thus, the unbiased estimator for  $\Delta$  is

$$\widehat{\Delta}_{PES} = \frac{\Delta_\sigma}{pq} = \frac{1}{0.2 \times 0.25} = 20. \quad (7)$$



(A) An example graph.

	$e$	$g$	$w(e, f), f \in g$	$\sigma$	$\Lambda_e$	$q$	$\Delta_\sigma$
1	(1,4)	$\phi$	-	$\phi$	0	-	0
2	(6,8)	<b>(6,8)</b>	-	$\phi$	0	-	0
3	(6,7)	(6,8)	(7,6,8)	<b>(7,6,8)<sup>-</sup></b>	<b>1</b>	1	0
4	(1,6)	(6,8)	(1,6,8)	(7,6,8) <sup>-</sup> , <b>(1,6,8)<sup>-</sup></b>	<b>2</b>	1	0
5	(6,11)	(6,8)	(8,6,11)	(7,6,8) <sup>-</sup> , (1,6,8) <sup>-</sup>	<b>3</b>	0.66	0
6	(2,3)	(6,8)	-	(7,6,8) <sup>-</sup> , (1,6,8) <sup>-</sup>	3	0.66	0
7	(9,10)	(6,8)	-	(7,6,8) <sup>-</sup> , (1,6,8) <sup>-</sup>	3	0.66	0
8	(1,2)	(6,8), <b>(1,2)</b>	-	(7,6,8) <sup>-</sup> , (1,6,8) <sup>-</sup>	3	0.66	0
9	(6,10)	(6,8), (1,2)	(8,6,10)	<b>(8,6,10)<sup>-</sup></b> , (1,6,8) <sup>-</sup>	<b>4</b>	0.5	0
10	(1,5)	(6,8), (1,2)	(2,1,5)	(8,6,10) <sup>-</sup> , (1,6,8) <sup>-</sup>	<b>5</b>	0.4	0
11	(6,9)	(6,8), (1,2)	(9,6,8)	(8,6,10) <sup>-</sup> , <b>(9,6,8)<sup>-</sup></b>	<b>6</b>	0.33	0
12	(1,3)	(6,8), (1,2)	(2,1,3)	<b>(2,1,3)<sup>-</sup></b> , (9,6,8) <sup>-</sup>	<b>7</b>	0.28	0
13	(8,9)	(6,8), (1,2)	(6,8,9)	(2,1,3) <sup>-</sup> , (9,6,8) <sup>+</sup>	<b>8</b>	0.25	<b>1</b>

(B) Steps on the graph in Panel (A) with  $p = 0.2, n = 2$ .

FIGURE 3.1: Steps of applying our PES on a toy graph.

### 3.4.3 The unbiased estimator

The proof that  $\widehat{\Delta}_{PES}$  is unbiased is presented in this section. Let  $\delta_i$  be the indicator function for the  $i^{th}$  triangle in the input graph. It is one when the  $i^{th}$  triangle is

sampled; otherwise it is zero. For each triangle, the probability of sampling the first edge is  $p$ , the probability of sampling the second edge is  $q$ . Note that the closeness of a wedge is checked every time when a wedge emerges in the pool. Hence the probability of sampling a triangle is  $pq$ . The expectation of  $\delta_i$  is  $pq$  and the expectation of  $\Delta_\sigma$  is:

$$\mathbb{E}(\Delta_\sigma) = \mathbb{E}\left(\sum_{i=1}^{\Delta} \delta_i\right) = \sum_{i=1}^{\Delta} \mathbb{E}(\delta_i) = \sum_{i=1}^{\Delta} pq = pq\Delta. \quad (8)$$

Thus, the unbiased estimator is as follows.

**Theorem 2.** *The unbiased estimator for PES algorithm is*

$$\widehat{\Delta}_{PES} = \frac{\Delta_\sigma}{pq}. \quad (9)$$

An interesting part of the algorithm is that  $q$  decreases over time, and the sampling probability of the second edge in Eq. 9 is the  $q$  in the final step, not the larger  $q$  values in earlier steps. Intuitively, edges sampled in earlier steps have a higher probability of being replaced during the process. The earlier the edge being scanned, the bigger the  $q$  is at that moment. But it also has a higher probability of being replaced in a later stage. Hence the overall probability is the same as the final  $q$ . Detailed proof is similar to reservoir sampling [68] using inductive inference, and is given as follows.

For the last candidate wedge at arrival time  $\Lambda_c$ , it is easy to understand that the second edge has a sampling probability  $q = n/\Lambda_c$ . Other wedges arrived before also has a sampling probability  $q$ , following reservoir sampling [68] as explained in the following inductive inference:

When  $\Lambda_c = n + 1$ , the sampling probability for wedges arrived before time  $n$  is:

$$1 \times \left( \frac{1}{n+1} + \frac{n}{n+1} \frac{n-1}{n} \right) = \frac{n}{n+1}. \quad (10)$$

This is because that there is a probability of  $1/(n+1)$  that the new wedge won't replace any old wedge; and there is a probability of  $n/(n+1)$  that an old wedge will be replaced. For each replacement, the probability of one particular wedge not being

replaced is  $n - 1/n$ .

Suppose that the old wedges are kept with probability  $n/(n+x)$  when  $\Lambda_c = n+x$ . When  $\Lambda_c = n+x+1$ , the sampling probability for wedges arrived before time  $n+x+1$  is:

$$\frac{n}{n+x} \times \left( \frac{1}{n+x+1} + \frac{n+x}{n+x+1} \times \frac{n+x-1}{n+x} \right) = \frac{n}{n+x+1}. \quad (11)$$

### 3.4.4 The variance

The variance of the estimator is complicated because of the involvement of two different sampling techniques—uniform sampling and reservoirs sampling. In PES, a wedge as a possible triangle is formed uniformly at random with probability  $p$  over an edge stream; and it is preserved with probability  $q$  in pool  $\sigma$ . Applying the variance on the estimator we get:

$$\begin{aligned} \text{var}(\widehat{\Delta}_{PES}) &= \text{var}\left(\frac{\Delta_\sigma}{pq}\right) = \text{var}\left(\sum_{i=1}^{\Delta} \frac{\delta_i}{pq}\right) \\ &= \frac{1}{(pq)^2} \sum_{i=1}^{\Delta} \sum_{j=1}^{\Delta} \text{cov}(\delta_i, \delta_j) \\ &= \frac{1}{(pq)^2} \left( \sum_{i=1}^{\Delta} \text{var}(\delta_i) + \sum_{i \neq j}^{\Delta} \text{cov}(\delta_i, \delta_j) \right). \end{aligned} \quad (12)$$

Recall that  $\delta_i$  is the indicator for the  $i^{\text{th}}$  triangle as defined before. By the definition of variance,  $\text{var}(\delta_i)$  is  $\mathbb{E}(\delta_i) - \mathbb{E}(\delta_i)\mathbb{E}(\delta_i)$ ; therefore, the cost of the first term in Eq. 12 is  $\Delta(pq - (pq)^2)$ . For the covariance, let  $\Phi$  be the number of pairs of triangles with a common edge. To identify such a case by PES, the common edge should be added into  $g$  with probability  $p$ . Furthermore, the other two edges need to be preserved in pool  $\sigma$  with probability  $(n^2-n)/(p^2\Lambda^2-p\Lambda)$ . Thus, the probability of sampling such a dependent pair is  $pq'^2$  where  $q'^2$  is  $(n^2-n)/(p^2\Lambda^2-p\Lambda)$ . Recall that  $n$  is the size of pool  $\sigma$ . Each dependent pair has five edges and the common edge should be visited before the other four and needs to be sampled with probability  $p$ . Clearly, the five edges can

arrive in 120 different orders in a stream; and in one-fifth of them, the common edge is the first one in the stream. Note that each dependent pair  $(\delta_i, \delta_j)$  appears twice in the covariance term. Thus, the cost of  $\Phi$  dependent cases is  $\frac{2\Phi}{5}(pq'^2 - (pq)^2)$ . Because the reservoir sampling is used to preserve wedges in pool  $\sigma$  we need to consider the cost of  $(\Delta^2 - 2\Phi - \Delta)$  independent pairs. Obviously, the probability of selecting a pair of independent triangles is  $p^2q'^2$ . By the definition of covariance, i.e.  $\mathbb{E}(\delta_i\delta_j) - \mathbb{E}(\delta_i)\mathbb{E}(\delta_j)$ , the cost of independent cases is  $(\Delta^2 - 2\Phi - \Delta)(p^2q'^2 - (pq)^2)$ . Substituting the costs in Eq. 12 and after some math simplification, the variance of the estimator is given by the following theorem.

**Lemma 1.** *Let  $\Delta$  be the true number of triangles and  $\widehat{\Delta}_{PES}$  be its estimation by PES. The variance of  $\widehat{\Delta}_{PES}$  is*

$$\text{var}(\widehat{\Delta}_{PES}) = \frac{\Delta(1 - pq)}{pq} + \frac{2\Phi(q'^2 - pq^2)}{5pq^2} + \frac{\Phi'(q'^2 - q^2)}{q^2}. \quad (13)$$

where  $\Phi$  is the number of pairs of shared triangles,  $q = n/p\Lambda$ ,  $q'^2 = (n^2 - n)/(p^2\Lambda^2 - p\Lambda)$ , and  $\Phi' = (\Delta^2 - 2\Phi - \Delta)$ .

The variance of the estimator depends on several metrics including  $\Delta$ ,  $\Phi$ ,  $p$  and  $q$ . In practice, we do not have the knowledge of these metrics. For example,  $\Delta$  is exactly what we are estimating. Hence, in order to know the performance of the estimator, we need to estimate the variance. Thus, we simplify the variance to have better insight into it. To do so, we translate the variance into RSE and use big data assumption to present the following theorem.

**Theorem 3.** *The RSE of  $\widehat{\Delta}_{PES}$  is estimated by*

$$\widehat{RSE}(\widehat{\Delta}_{PES}) \approx \Delta_\sigma^{-1/2}. \quad (14)$$

*Proof.* Translate  $\text{var}(\widehat{\Delta}_{PES})$  into the RSE =  $\sqrt{\text{var}}/\Delta$ . When the input graph is large, approximating  $n - 1 \approx n$  and  $p\Lambda - 1 \approx p\Lambda$  are valid. Thus, we get:

$$RSE(\widehat{\Delta}_{PES}) \approx \left[ \frac{1}{\Delta pq} \left( 1 - pq + \frac{2\Phi}{5\Delta}(q - pq) \right) \right]^{1/2}. \quad (15)$$

TABLE 3.2: Properties of the networks in our experiments, sorted by graph size  $N$ .

Dataset	$N(\times 10^6)$	$\langle d \rangle$	$\mathcal{C}$	Type	Dataset	$N(\times 10^6)$	$\langle d \rangle$	$\mathcal{C}$	Type
1. Ego-facebook [61]	0.004	43.69	0.519	OSN <sup>1</sup>	25. Youtube [61]	1.1	5.27	0.006	OSN
2. CA-GrQc [61]	0.005	5.52	0.629	COL <sup>2</sup>	26. Dblp [62]	1.3	8.16	0.170	COA
3. Wiki-vote [61]	0.007	28.32	0.125	OSN	27. Wiki-Polish [62]	1.5	55.17	0.01	WEB
4. AstroPh [62]	0.01	21.10	0.31	CIT <sup>3</sup>	28. Trec-wt10g [62]	1.6	8.33	0.014	WEB
5. CA-CondMat [61]	0.02	8.08	0.264	COA <sup>4</sup>	29. Wiki-Portuguese [62]	1.6	48.19	0.022	WEB
6. HepPh [62]	0.02	224.14	0.279	COA	30. Wiki-Japanese [62]	1.6	69.82	0.021	WEB
7. Enron-email [62]	0.03	10.02	0.085	ECO <sup>5</sup>	31. Pokec [62]	1.6	27.31	0.046	OSN
8. Brightkite [61]	0.05	7.35	0.110	OSN	32. As-skitter [61]	1.6	13.08	0.005	INT <sup>8</sup>
9. Facebook [62]	0.06	25.64	0.147	OSN	33. Wiki-Italian [62]	1.8	72.90	0.024	WEB
10. Epinions [62]	0.07	10.69	0.065	OSN	34. Hudong [62]	1.9	14.54	0.003	WEB
11. Slashdot-Zoo [62]	0.07	11.82	0.023	OSN	35. Hollywood [63,64]	1.9	24.51	0.152	OSN
12. Livemocha [62]	0.1	42.13	0.014	OSN	36. Flicker [62]	2.3	19.83	0.107	OSN
13. Douban [62]	0.1	4.22	0.01	OSN	37. Flixster [62]	2.5	6.27	0.013	OSN
14. Gowalla [61]	0.1	9.66	0.023	OSN	38. Wiki-Russian [62]	2.8	44.20	0.015	WEB
15. Libimseti [62]	0.2	155.97	0.007	OSN	39. Wiki-French [62]	3.0	55.21	0.015	WEB
16. Digg [62]	0.2	11.07	0.061	OSN	40. Orkut [62]	3.0	76.28	0.041	OSN
17. Dblp-Coau [61]	0.3	6.62	0.306	COA	41. Wiki-German [62]	3.2	40.77	0.0088	WEB
18. Web-NotreDame [61]	0.3	6.69	0.087	WEB <sup>6</sup>	42. USpatent [62]	3.7	8.75	0.067	CIT
19. Amazon [61]	0.3	5.53	0.205	COP <sup>7</sup>	43. LiveJournal [61]	3.9	17.35	0.125	OSN
20. Actor [62]	0.3	78.68	0.166	COL	44. DBpedia [62]	18	13.89	0.0016	WEB
21. Citeseer [62]	0.3	9.03	0.049	CIT	45. Web-Arabic [63,64]	22	48.70	0.031	WEB
22. Dogster [62]	0.4	40.03	0.014	OSN	46. Gsh-2015 [63,64]	29	9.18	0.007	WEB
23. Catster [62]	0.6	50.32	0.028	OSN	47. MicrosoftAc.G. [65]	46	22.61	0.015	CIT
24. Web-Google [62]	0.8	9.87	0.055	WEB	48. Friendster [62]	65	55.06	0.017	OSN

<sup>1</sup> Online Social Network   <sup>2</sup> Collaboration   <sup>3</sup> Citation   <sup>4</sup> Coauthorship   <sup>5</sup> E-communication   <sup>6</sup> Web Graph   <sup>7</sup> Co-purchasing   <sup>8</sup> Internet topology

When graph is large sampling probabilities  $p$  and  $q$  is very small and terms  $-pq$  and  $+\frac{2\Phi}{5\Delta}(q - pq)$  in Eq. 15 are neglectable. Thus, Eq. 15 is simplified as  $(\Delta pq)^{-1/2}$ . Because  $\Delta_\sigma = \Delta pq$ , the theorem thus follows.

### 3.5 Experiments

We conduct experiments to 1) compare our algorithms with the state-of-the-art algorithms GPS-In and GPS-Post [1]. Other algorithms are not compared because it is already demonstrated that they are inferior to GPS-In; and 2) to verify our analytical results presented in Theorem 1 and 3. This is needed because there are approximations in the derivation. The precise results are long formulas that depend on the structure of the graph, such as the number of triangles ( $\Delta$ ) and the count of dependent triangles ( $\Phi$ ). Theorems 1 and 3 give more concise results by omitting some terms in the long formula by assuming the graph is large and  $p$  is small. How good is such approximation needs to be evaluated empirically.

The code along with all the data, including some intermediate data, are available



at <http://cs.uwindsor.ca/~etemadir/PES>.

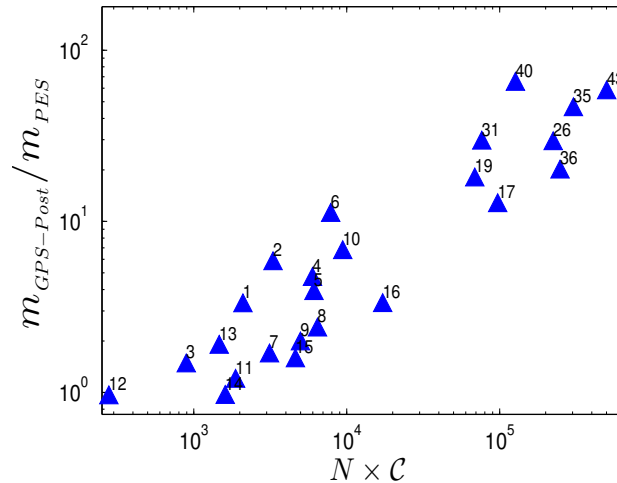
### 3.5.1 Data

Because the performance of sampling algorithms often varies from data-to-data, especially depends on the structure of the graphs, we verify our results extensively with many (48) real networks with different size from varieties of domains. The size ranges from 4 thousand to 65 million nodes. The domains include online social networks (OSN), web graphs, citation and co-authorship networks, etc. In some figures, we only plot half of the datasets (24) to save space. Other datasets have similar behaviours.

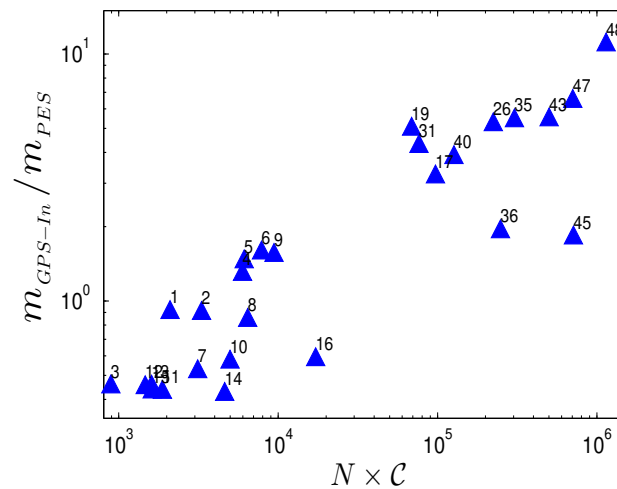
It is computationally costly to obtain the ground truth of large graphs. Luckily, we have access to two servers each with 24 cores and 256 GB RAM to carry out such intensive computing. Table 3.2 summarizes the networks and their statistics. The graphs are sorted by their node size  $N$ . In the table  $\langle d \rangle$  is average degree, and  $\mathcal{C}$  is clustering coefficient ( $\mathcal{C} = 3\Delta/\Lambda$ ). We executed the estimators on the graphs and reported the results along with our observations in the following sections. The results were obtained over 1000 independent runs for the graphs except for the four largest graphs that are repeated 500 times.

### 3.5.2 Comparison with GPS-In and GPS-Post

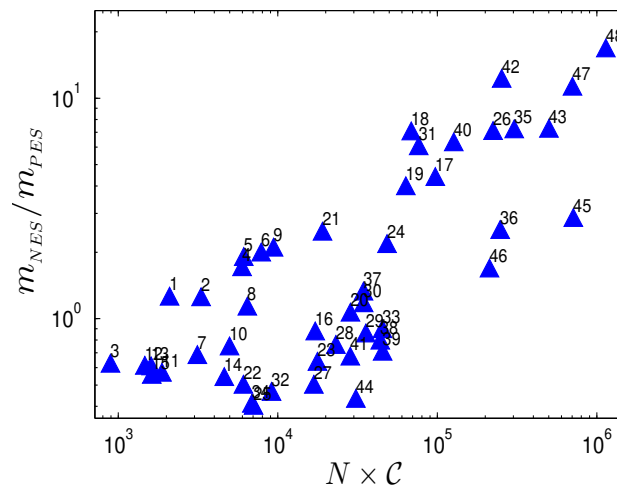
Fig. 3.2 summarizes the comparison of PES with the state-of-the-art methods GPS-Post (Panel A) and GPS-In (Panel B) [1]. We also compared NES and our PES algorithm in Panel C. We set the sampling probability of the estimators to obtain the same RSE. Here we report the ratios between sample sizes when RSE=0.2. Similar phenomenon is observed for other RSEs. In each panel, the Y-axis is the ratios, and the X-axis is the graph size that is represented by the node size  $N$  multiplied by clustering coefficient. In all the methods,  $m$  is the 'sample size'. Algorithms differ in the definition of 'sample size' because some algorithms maintain a reservoir of wedges in addition to subgraph  $g$  or use extra memory per sampled edge to store information



(A) Our PES vs. GPS-Post



(B) Our PES vs. GPS-In



(C) Our PES vs. NES

FIGURE 3.2: Sample size ratios of our PES vs. GPS-Post (Panel A), GPS-In (Panel B), and NES (Panel C) when RSE=0.2. 51

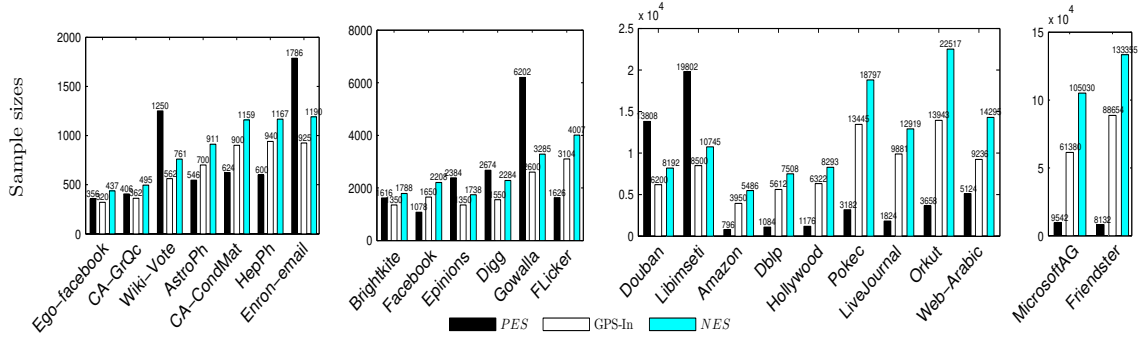


FIGURE 3.3: Our PES uses **less sample sizes** compared to other methods to obtain an estimation with the same  $RSE=0.2$  on most of the graphs. Note that the sample size include both the size of the subgraph and the reservoir for our PES but for GPS methods extra memory per sampled edge was ignored.

about sampled edges in subgraph  $g$ . NES has  $g$  only. Hence the sample size  $m$  is the number of edges (denoted by  $|g|$ ), which is equal to  $pM$ . PES maintains a wedge pool  $\sigma$ . Hence the sample size is  $|g| + |\sigma|$ . GPS -In and -Post also store subgraph  $g$  and two additional values per edge in  $g$ . However, we consider their sample sizes as  $|g|$ .

In the panels, each marker represents one of the 48 graphs described in Table 3.2. From the figure we make several observations:

- Our PES outperforms GPS-In and GPS-Post consistently in terms of sample size. All the ratios are above one in Panel (A), meaning that PES needs fewer samples than GPS-Post for all the datasets. For instance, take Orkut (labeled 43) in Panel A has ratio 73, meaning that GPS-Post needs 73 times more sampled edges compared to our PES. Compare to GPS-In, our PES also needs less sample size in most of the graphs. For example, LiveJournal (labeled 43) in Panel A has the ratio 5.4, meaning that GPS-In requires 5.4 times more sampled edges compared to PES to obtain an estimation with the same RSE. The improvement margin is higher for GPS-Post, which is expected since GPS-In improves GPS-Post. Take the same LiveJournal data for example, as shown in Panel A, the ratio is 66, much higher than 5.4.
- The ratio grows almost exponentially with data size. In other words, compared with PES, the sample size of other algorithms grows exponentially with graph

size. This result has high implication for very large graphs: although other algorithms can deal with current data, their performance will deteriorate exponentially with graph size. The Pearson correlation coefficient between the ratios and the size of data is 82 for GPS-In, 80 for GPS-Post, and 79 for NES.

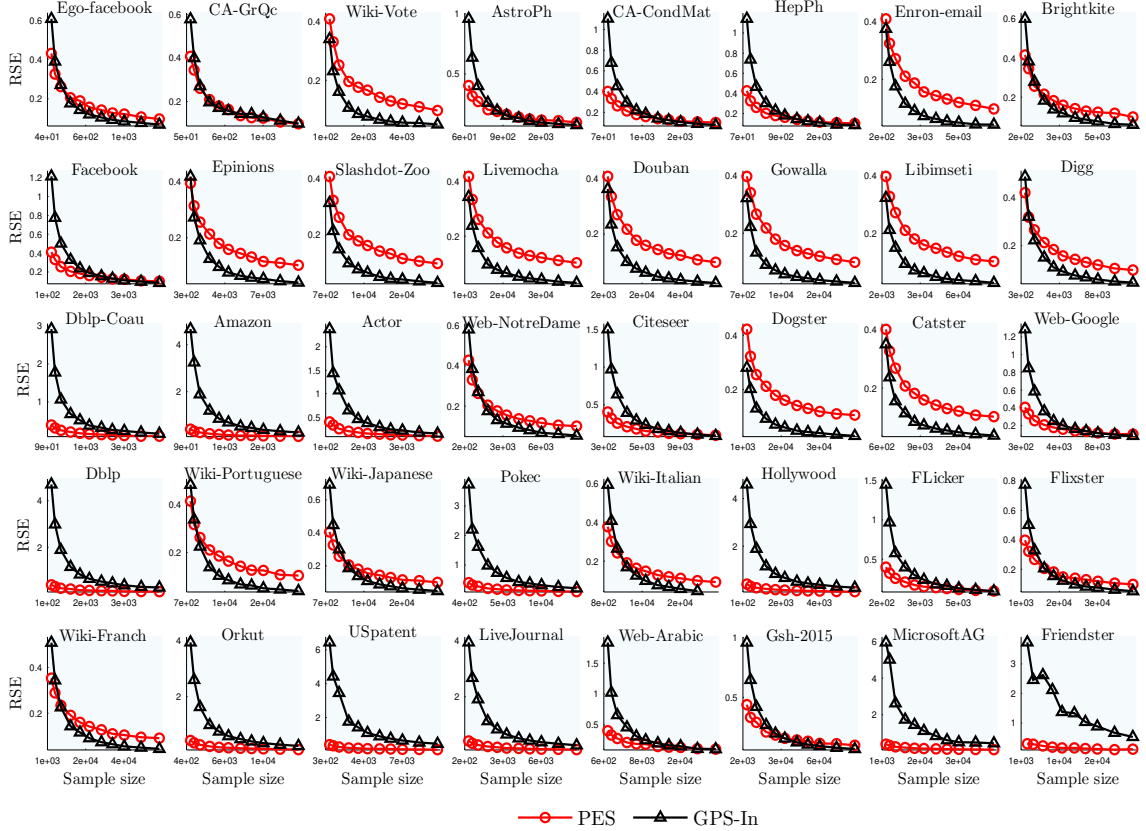


FIGURE 3.4: Our PES outperforms GPS-In in terms of RSEs when both methods are using the same sample sizes. Note that for our PES, the sample size includes both the size of subgraph  $g$  and reservoir size, i.e.,  $|\sigma|$ . For GPS-In, we only considered the size of subgraph  $g$  as a sample size, and ignored two additional values per sampled edge in  $g$ .

Fig. 3.3 compares the actual sample sizes of the three methods side by side. The sample sizes are the ones to achieve the same  $RSE=0.2$ . Take the Friendster data for example, the samples for PES is 8,132, meaning that the subgraph size is 4,066, and the reservoir size is 4,066 to achieve  $RSE=0.2$ . On the other hand, the sample size of GPS-In is 88,654, meaning that  $|g| = 88,654$ . Similarly, the sample size of NES is 133,355, meaning that  $|g| = 133,355$ . Note that even though GPS-In outperforms

NES in terms of  $|g|$ , the overall cost including both  $g$  and extra data for sampled edges in  $g$  is actually higher. As shown in the figure, our PES outperforms the other methods in most of the graphs. It is obvious that, the performance ratios grow by increasing the size of graphs. For example, all the methods need almost the same sample sizes for Ego-facebook graph (the smallest graph in our dataset) to achieve the estimation with the same  $RSE=0.2$ . Take the Friendster data as the largest graph in the dataset, PES needs 10.9 times less sample size compare to GPS-In.

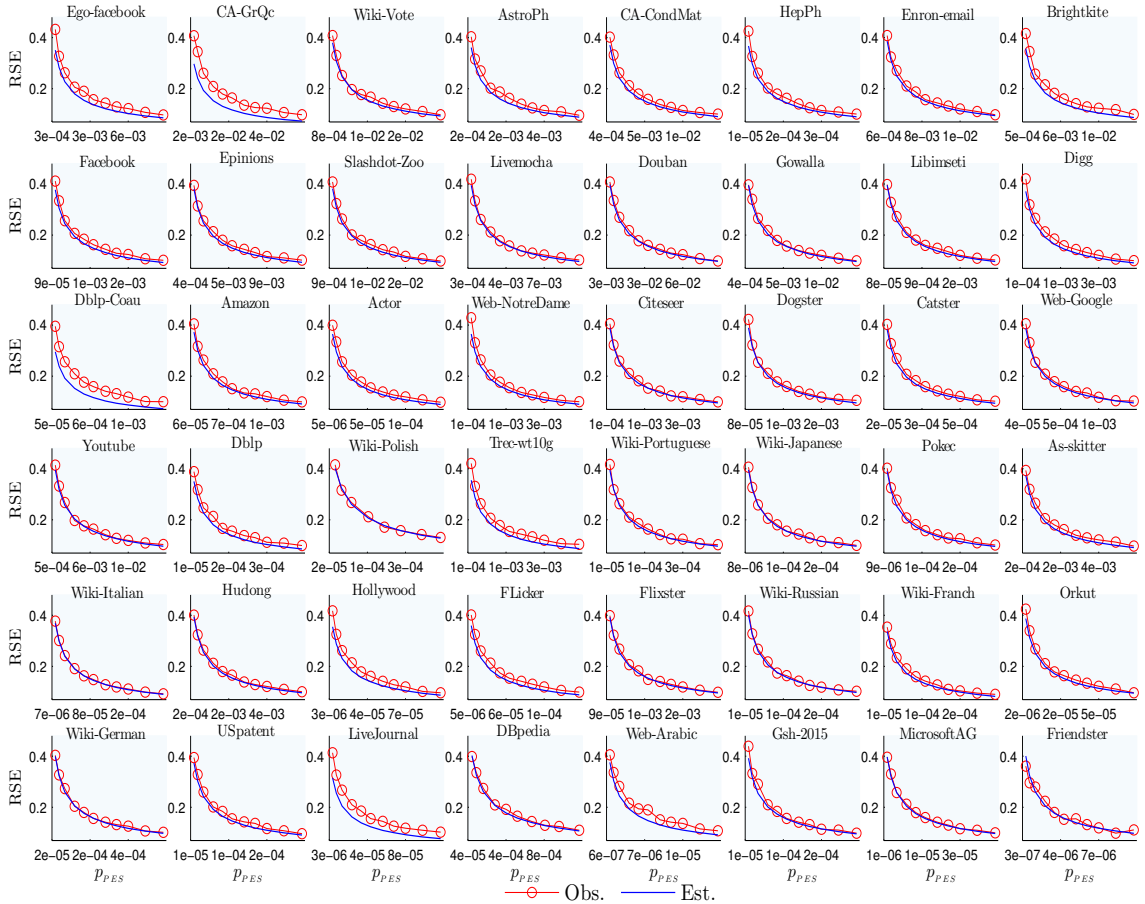


FIGURE 3.5: The observed RSEs of  $\hat{\Delta}_{PES}$  support our estimated RSEs based on Eq. 14.

Next, we investigate how the performance ratios between the methods change by increasing the accuracy of estimators (decreasing the RSE). To do so, we set the parameters of PES to obtain the RSEs between 0.1 and 0.4. Then, GPS-In was run using the same sample sized used in PES. Note that we considered both the size of

subgraph  $g$  and the pool  $\sigma$  as a sample size of PES. We only report the observed RSEs of our PES vs. GPS-In in Fig. 3.4 because GPS-In is more efficient compared to other existing methods. It can be seen that by increasing the sample size, the gap between the RSEs of the methods diminishes. Still, PES outperforms GPS-In in terms of obtaining accurate estimation using the same sample sizes for large graphs, as we can see in the last row of the plot.

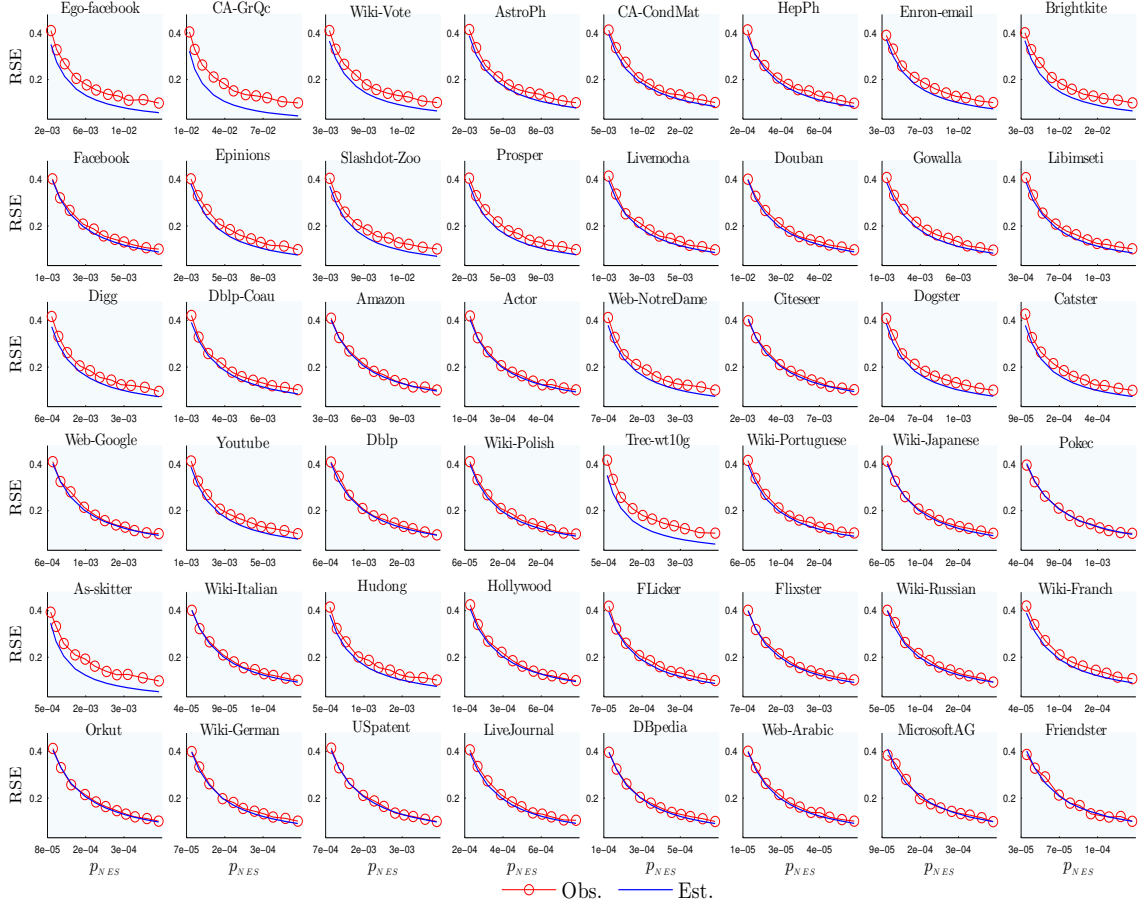


FIGURE 3.6: The observed RSEs of  $\hat{\Delta}_{NES}$  fit very well our estimated RSEs based on Eq. 3.

### 3.5.3 Validation of Theorems 1 & 3

We conduct experiments to verify our approximations used in the derivations of Theorems 1 & 3. Thus, sampling probability  $p$  of the PES and NES were initialized in a way that the estimators achieve the RSEs between 0.1 and 0.4 to get estimations

in range  $[\Delta \pm 0.8\Delta, \Delta \pm 0.2\Delta]$  with 95% confidence. The observed and estimated RSEs are reported in the plots of Fig. 3.5 and 3.6. We report the results for 24 representative graphs. Similar patterns are observed for the remaining data sets.

As shown in the plots, in both theorems our approximations work very well. It can be seen that our estimated RSEs (blue lines in the plots) fit perfectly the observed ones (red lines with circle markers) not only for large graphs but also for small-sized ones. Thus, in practice the theorems can be used to control the accuracy of the estimators. Moreover, they can be used to quantify the performance ratio between the methods as in the following section.

### 3.5.4 An implication of Theorems 1 & 3

We use Theorems 1 & 3 to quantify the performance ratio between NES and PES. Suppose  $p_{NES}$  and  $p_{PES}$  are sampling probability of NES and PES respectively to achieve the same RSE. Using the result of Theorems 1 and 3, we need to have  $\Delta_\sigma^{-1/2} \approx \Delta_g^{-1/2}$ . Replace  $\Delta_\sigma = p_{PES}q\Delta$  and  $\Delta_g = p_{NES}^2\Delta$ . Recall that  $q$  is the sampling probability of preserving candidate wedges in pool  $\sigma$ . Suppose the size of pool  $\sigma$  is the same as the size of  $|g|$ , i.e.  $|\sigma| = |g|$ . Thus,  $q \approx M/\Lambda$ . After some math simplifications, we get

**Corollary 1.** *Let pool size be  $|\sigma| = |g| = p_{PES}M$  in PES. The ratio between sampling probabilities of PES and NES to achieve the same RSE is given by*

$$\frac{p_{NES}}{p_{PES}} \approx \frac{M}{p_{NES}\Lambda}. \quad (16)$$

Corollary 1 says that the sample size ratio between PES and NES depends on  $M$ ,  $\Lambda$ , and sampling probability of NES ( $p_{NES}$ ). Recall that  $M$  and  $\Lambda$  are the number of edges and the count of wedges in the input graph.

To verify the corollary, the parameters of the methods were set to achieve the RSEs between 0.1 and 0.4. Note that we set up the size of pool as  $p_{PES}M$  in PES, i.e.  $|\sigma| = |g|$ . The observed and estimated ratios based on Eq. 16 are reported in Fig. 3.7. It can be seen that the observed ratios support our theoretical results in Eq. 16,

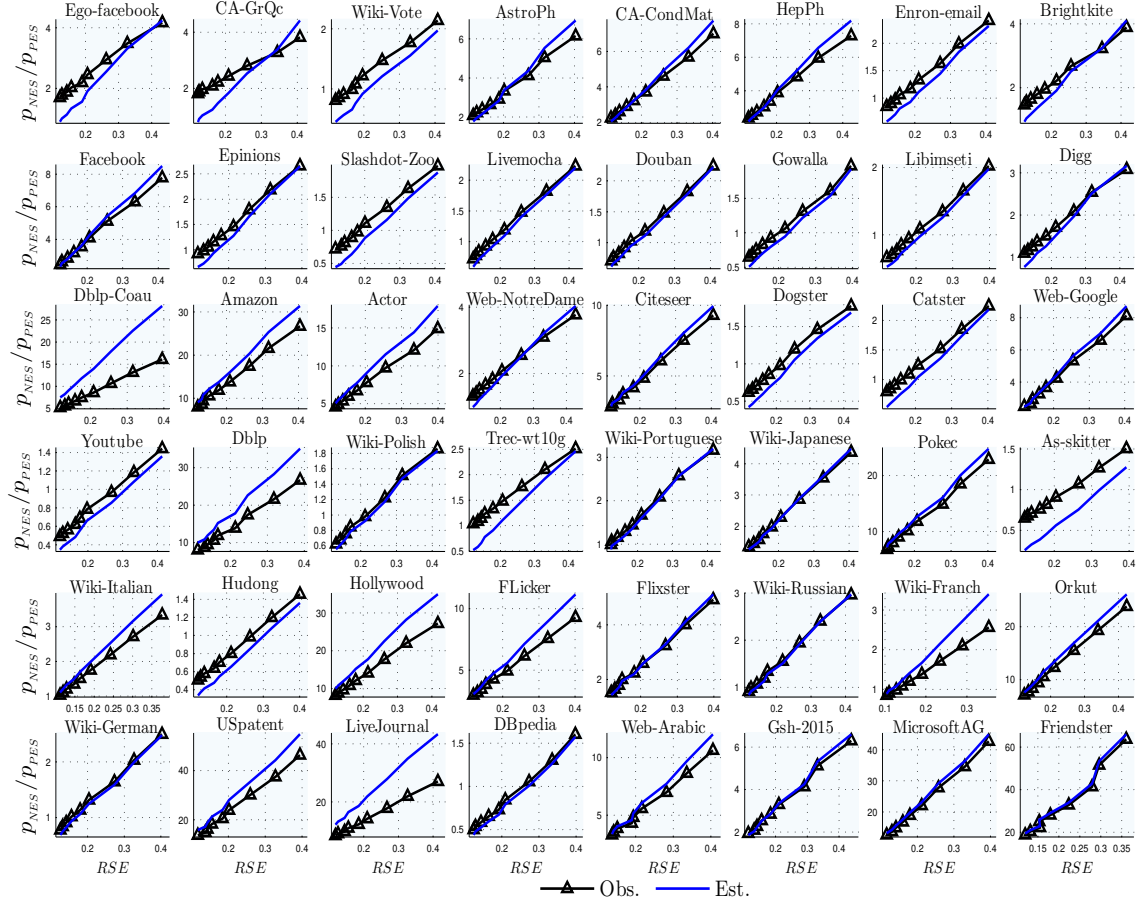


FIGURE 3.7: Our PES outperforms NES. The observed and estimated ratios between  $p_N$  and  $p_D$  when the methods achieve the same RSEs between 0.1 and 0.4. The estimated ratios are obtained using Eq. 16.

i.e., the estimated ratios based on Eq. 16 fit the observed values very well in most of the representative graphs. However, as expected there is a small gap between the observed and estimated ratios in a few cases.

### 3.6 Conclusions and discussions

This work proposes a streaming algorithm called PES. It improves NES by increasing the chance of observing a triangle over a stream from  $p^2$  in NES to  $pq$ , where  $q$  is greater than  $p$  and it is automatically adjusted over the stream. PES outperforms GPS-In consistently in all the datasets that have been tested. The performance ratio can be as high as 11. An important observation is that the performance ratio grows



exponentially with data size, indicating that we could observe higher performance gain in larger datasets. We have tested on networks with 65 million nodes. Due to the prohibitive cost to calculate the ground truths (such as triangle, wedges, and shared wedges and triangles) of very large graph, we did not experiment with even larger networks. We should note that real networks often have billions of nodes, much larger than our experimented data. We expect that our algorithm would be particularly useful in such very large networks.

In retrospect, the key to improve the performance is to identify triangles as many as possible during the sampling process. In the streaming model, we need to scan each edge anyway. Thus, NES fits naturally with the streaming model because the closeness check almost comes free, especially because the sample size is small compared with the original graph. PES improves NES further by increasing the sampling probability of the second edge of the triangle. It improves GPS-In because GPS-In does not always add the second edge as we did in PES.

Most algorithms are compared empirically only. This is limited, and conclusions may not be true for other datasets. We compare NES and PES analytically, and quantify the performance gain. The analytical comparison also gives us a deeper understanding as for when PES is better. PES hinges on the value of  $q$ . Probability  $q$  becomes larger than  $p$  when the graph becomes larger.

---

## CHAPTER 4

# The Bias and Variance in Clustering Coefficient Estimation

---

---

### 4.1 Introduction

Clustering coefficient ( $\mathcal{C}$ ) is one of the important metrics to analyze networks. It has been used in many applications including graph clustering and community detection [34], spam detection [69], link prediction [35], microarray data and DNA sequence analysis [42, 43], word-learning [46], etc. Computing  $\mathcal{C}$  on large networks is a computationally intensive task. The time complexity of the state-of-the-art method is  $\Theta(M^{1.41})$ , where  $M$  is the number of edges in the network [3]. Thus, sampling-based algorithms are indispensable and studied extensively, e.g., in [1, 8, 21, 29, 30, 70].

Despite extensive work in this area, there is a lack of formal analyses of the algorithms, in particular the lack of estimators for the variance and bias of the estimation algorithms. Most algorithms are compared empirically, e.g. in [1, 8, 21, 29], thus their performances are often data dependent. Some algorithms do discuss the bias and variance problems, but they all fail to give the exact formulas. E.g., [1, 8] give a variance that is generic for every problem; [1, 8, 21, 29] mentioned about the existence of the bias.

This chapter gives the bias and variance for both streaming and non-streaming algorithms. In addition to the bias and variance formulas, a more important issue is their estimation without the global data. In order to have the confidence interval for an estimate, we need to know the variance and bias during the sampling and

---

A short version of this chapter was published in IEEE BigData'17 [30].

estimation process, not the variance and bias that is derived from the knowledge of the global data. In other words, we need to estimate the variance and bias.

We derived the variance and bias so that algorithms can be compared analytically. The formulas are long and tedious, hard to be applied in practice. A more interesting result of our work is the simplification of the formulas so that it can be applied easily. Based on the assumption that the data is big, we show that the RSE (Relative standard error, the normalized square root of variance) can be estimated by the reciprocal of the square-root of the triangles observed in samples. In other words, the accuracy grows at the rate of  $\sqrt{\Delta_g}$ , where  $\Delta_g$  is the observed sample closed wedges in the streaming process.

The simplified result is especially important in the era of big data. When networks are small, we can hardly predict the behaviour of the sampling algorithms. However, when the data is big, the performance can be characterized simply by the closed wedges observed in the process. This result is also supported by our extensive experiments on 56 real networks of different size and structure.

Let  $\Lambda$  denote the number of wedges (or paths of length two),  $\Delta$  the number of closed-wedges. Metric  $\mathcal{C}$  is defined as the ratio between the two, i.e.,

$$\mathcal{C} = \frac{\Delta}{\Lambda}. \tag{1}$$

Note that here we restrict our discussion to the global  $\mathcal{C}$  as defined above for the sake of simplicity. There are other notions of  $\mathcal{C}$ s, such as local and average  $\mathcal{C}$ , that are beyond the scope of this work. However, our method can be easily extended to those  $\mathcal{C}$ s.

Suppose that  $\hat{\Delta}$  and  $\hat{\Lambda}$  are unbiased estimators for  $\Delta$  and  $\Lambda$ , respectively. In other words,  $\mathbb{E}(\hat{\Delta}) = \Delta$ , and  $\mathbb{E}(\hat{\Lambda}) = \Lambda$ . It has been taken for granted, e.g., in [8, 21, 23] and [32], that the  $\mathcal{C}$  estimator  $\hat{\mathcal{C}}$  is:

$$\hat{\mathcal{C}} = \frac{\hat{\Delta}}{\hat{\Lambda}}. \tag{2}$$

Unfortunately, this is a biased estimator as we can see from the fact that  $\mathbb{E}(X/Y) \neq \mathbb{E}(X)/\mathbb{E}(Y)$ . More precisely, by applying expectation on the estimator, we have:

$$\mathbb{E}(\widehat{\mathcal{C}}) = \mathbb{E}\left(\frac{\widehat{\Delta}}{\widehat{\Lambda}}\right) \neq \frac{\mathbb{E}(\widehat{\Delta})}{\mathbb{E}(\widehat{\Lambda})} = \frac{\Delta}{\Lambda} = \mathcal{C}.$$

While it is easy to understand the existence of bias, quantifying and correcting the bias is a challenging task. Recently, Jha et al. [21, 29] and Ahmad et al. [1, 8] noticed the bias problem and left it as an open problem to solve. In 2017, Ahmad et al. discussed the bias problem again, but could not quantify it [1].

The analysis of the bias needs to be embedded in a concrete sampling method. We base our following discussions on random edge sampling and streaming sampling. Random edge sampling has been widely used for estimating  $\mathcal{C}$  [8, 21, 29], triangles [7, 10, 17, 20, 28, 71], and other graph properties [11, 13, 72]. It is also closely related to other sampling methods. For example, random walk [23, 32, 73] is an approximation of random edge sampling in that their node sampling probabilities are asymptotically equal in undirected graphs. Random node sampling can also be associated with random edge sampling—when we sample node with probability proportional to its size (PPS), it is actually a kind of random edge sampling in the sense that sampling probability of the node is the same in two sampling schemes.

For this random edge sampling scheme, we quantify the bias using the ‘power method’ [74]. It involves a Taylor expansion that results in a long formula. The intuitive understanding is lost in the complex formula without simplification. Hence, we simplify the formula, and derive an adjusted estimator as follows:

$$\widehat{\mathcal{C}}^* = \frac{\widehat{\Delta}}{\widehat{\Lambda}} \left[1 + \frac{r}{p}\right]^{-1}, \quad (3)$$

where  $p$  is the sampling probability,  $r$  is a constant determined by the graph topology that will be explained later in Section 4.3. Roughly speaking,  $r$  is dominated by the ratio of the third moment and the square of the second moment of the degrees of the graph. The corrected estimator in Eq. 3 highlights the importance of the problem

particularly in the age of big data: when the graph is very large, only a small fraction of the graph is needed to achieve high accuracy, resulting in a very small  $p$ . Although  $r$ , in general, is a very small number,  $r/p$  may not be neglectable in this case. We will show that  $r/p$  can be as high as 0.04 in certain cases.

Eq. 3 is good for understanding the nature of the bias problem. However, it cannot be used for estimation in practice because  $r$  is unknown from the sample. In other words, we also need to estimate  $r$  to correct the bias. Thus, we derive a corrected estimator for random edge sampling as below:

$$\hat{c}^+ = \frac{\Delta_g}{\Lambda_g} \left[ 1 + r_g \right]^{-1}, \quad (4)$$

where  $r_g = 2\Psi_g/\Lambda_g^2 - \Omega_g/\Delta_g\Lambda_g$ . Variable  $\Psi_g$  is the number of shared wedges,  $\Omega_g$  is the number of shared wedges and closed-wedges, all in sample graph  $g$ . We show that the result can be simplified further by taking the first term only in the above formula, assuming that the graph is large. Based on this, the bias can be quantified by the second and third moments of the degrees of the nodes in the graph. Since the simplified result is derived using several approximations, we need to empirically evaluate the approximation using real graphs. The result is confirmed and explained on 56 real-world graphs.

## 4.2 Background and related work

Let  $G(\mathcal{V}, \mathcal{E})$  be a simple graph, where  $\mathcal{V}$  and  $\mathcal{E}$  are the set of nodes and the set of edges, respectively. Let  $N = |\mathcal{V}|$ ,  $M = |\mathcal{E}|$ , and each node is labeled as  $1, 2, \dots, N$ . Let  $d_i$  denote the degree of node  $i$ , for  $i \in \{1, 2, \dots, N\}$ . A wedge  $\mathcal{W}$  is a triplet  $(u, v, w)$ , where  $u, v, w \in \mathcal{V}$  are three distinct nodes,  $(u, v) \in \mathcal{E}$ , and  $(v, w) \in \mathcal{E}$ . Wedge  $\mathcal{W}$  is closed if  $(u, w) \in \mathcal{E}$ . Otherwise it is open. A triangle consists of three (closed) wedges. Let  $\Lambda_i = d_i(d_i - 1)/2$  denote the number of wedges for node  $i$ , and  $\Delta_i$  the number of closed-wedges for node  $i$ . Clustering coefficient is defined as the

TABLE 4.1: Summary of the notations

Notation	Meaning
$G(\mathcal{V}, \mathcal{E})$	Input graph (undirected and no self-edges)
$g$	A subgraph of $G$
$N, M$	Number of nodes and edges in $G$
$n$	Sample size
$d_i$	Degree of node $i$ in $G$
$\Lambda$	# wedges in $G$
$\Delta$	# closed-wedges in $G$
$\Phi$	Number of triangle pairs that share an edge
$\Lambda_g$	# wedges in $g$
$\Delta_g$	# closed-wedges in $g$ . Closeness checked on $G$
$\Lambda_i$	# wedges of node $i$
$\Delta_i$	# closed-wedges of node $i$
$\Psi$	# pairs of shared wedges in $G$
$\Omega$	# pairs of a wedge and a closed-wedge sharing one edge in $G$
$\Phi_g$	$\Phi$ counted in $g$
$\Psi_g$	$\Psi$ counted in $g$
$\Omega_g$	$\Omega$ counted in $g$
$\langle d \rangle$	Average degree of $G$ . $\langle d \rangle = \frac{1}{N} \sum_{i=1}^N d_i$ .
$\langle d^2 \rangle$	Second moment. $\langle d^2 \rangle = \frac{1}{N} \sum_{i=1}^N d_i^2$ .
$\langle d^3 \rangle$	Third moment. $\langle d^3 \rangle = \frac{1}{N} \sum_{i=1}^N d_i^3$ .

proportion of the wedges that are closed [6, 31], i.e.,

$$\mathcal{C} = \frac{\sum_{i=1}^N \Delta_i}{\sum_{i=1}^N \Lambda_i} = \frac{\Delta}{\Lambda}. \quad (5)$$

Table 4.1 summarizes a list of notations used in this chapter.

### 4.2.1 Related work

A number of sampling-based methods have been proposed in recent years to estimate  $\mathcal{C}$ , e.g., in [1, 6, 8, 21, 23, 29]. Such approaches can be categorized into two models: *streaming* and *none-streaming*. The former model requires a sequential access to the edge/node list of the entire input graph. The goal is to achieve an accurate estimation by storing small number of edges/nodes in a limited memory window over one or several scans of the graph data. In contrast, non-streaming methods have more general sampling scheme without streaming limitation and need to have random

access to the nodes/edges of input graph. Similarly, they aim to obtain an accurate estimation using small number of samples.

**A none-streaming model:** a straightforward method in this model is *wedge sampling* [6]. It selects wedges uniformly at random and the fraction of closed ones is used as an unbiased estimation for  $\mathcal{C}$ . This method requires roughly  $4/\epsilon^2\mathcal{C}$  sampled wedges to obtain an estimation in interval  $\mathcal{C} \pm \epsilon\mathcal{C}$  with 95% confidence. To sample a random wedge  $(u, v, w)$ , the center node  $v$  needs to be taken proportional to  $\Lambda_v$ . Recall that  $\Lambda_v$  is the number of wedges for node  $v$ . Then, the other two endpoint nodes  $u$  and  $w$  need to be chosen from the neighborhood of node  $v$  uniformly at random. Unfortunately, this method is not scalable on massive graphs because sampling a random node proportional to the number of its wedges to build a random wedge is an intensive task.

A scalable technique in this context is *edge sampling*. It samples some random edges from input graph  $G$  to build subgraph  $g$ . Then, the number of triangles and wedges in  $g$  are used to estimate  $\mathcal{C}$ . A naive edge sampling technique is to sample edges uniformly at random and create a subgraph  $g$ . Then, use the number of wedges and closed ones in  $g$  to estimate  $\mathcal{C}$ . Let  $p$  be the sampling probability to add edges into  $g$ . Thus, probability of observing a closed wedge in  $g$  is  $p^3$  [7]. Similarly, sampling probability of a wedge in  $g$  is  $p^2$ . Let  $\Lambda_g$  and  $\Delta_g$  be the number of wedges and closed ones in  $g$  respectively. Thus, using the unbiased estimators for  $\Delta$ , i.e.  $\Delta_g/p^3$  [7], and for  $\Lambda$  as  $\Lambda_g/p^2$ , the estimator for  $\mathcal{C}$  is

$$\hat{\mathcal{C}} = \frac{\Delta_g}{p\Lambda_g}. \quad (6)$$

To obtain an estimation with additive error  $\epsilon$  with 95% confidence, the size of subgraph  $g$  needs to be roughly  $1.5M/(\epsilon^2\Delta)^{1/3}$ . Unfortunately, such a technique suffers from the paucity of closed wedges in the sampled graph  $g$  when sampling probability  $p$  is small which is the case for very large graphs. Furthermore, such estimator is biased. When graph data is not entirely available, *indirect sampling* technique can be used to take samples from input graph  $G$ . Authors in [23] used

random walk to sample nodes proportional to their degrees. Then, the number of wedges and closed wedges in the sample are used to estimate  $\mathcal{C}$ . Such an estimator is biased and its bias has not been quantified. Furthermore, the variance of the estimator has not been derived.

**A streaming model:** in this model, edges of the input graph are scanned one by one from a stream of edges of the original graph. The goal is to obtain an accurate estimation by storing sampled edges in a limited memory window over one or several passes of whole graph data. To sample a random wedge in this model, two passes over an edge stream are required. Furthermore, one additional pass over the edge stream is needed to check the closeness of the random wedge [10, 17]. Therefore, sampling a random wedge from a large graph is an intensive task. Akin to non-streaming model, a scalable technique to estimate  $\mathcal{C}$  in this model is *edge sampling* [1, 8, 21, 29].

Authors in [21, 29] increased probability of identifying a triangle in the sample by storing two separate pools—edge and wedge pools. Edges in the stream were preserved in an edge pool with fix-sized  $n$ . By updating the edge pool, new wedges formed in the pool were preserved in a wedge pool with fixed-sized  $k$ . The closenesses of wedges in the wedge pool are also checked for each edge in the stream. Fraction of closed wedges in the wedge pool is used to estimate  $\mathcal{C}$ . Using Chernoff bound the authors showed that the memory usage of the method is at most  $M/(\epsilon^2\Delta^{1/2})$  to have an estimation with additive error  $\epsilon$ . Note that the estimator still is biased as the author mentioned it in [29]. To the best of our knowledge, there is no study for the variance and bias of this estimator.

Next attempt to increase the chance of observing a closed wedge in  $g$  is to sample edges from a stream with different probabilities, i.e.  $p$ ,  $q$ , and 1 where  $q > p$  [8]. The process is as follows. If a new edge  $e$  closes a wedge in  $g$  it is sampled unconditionally. Otherwise if  $e$  connected into some edges in  $g$  it is chosen with probability  $q$ ; otherwise with  $p$ . The number of wedges and closed ones in  $g$  are used to obtain an estimation of  $\mathcal{C}$ . The bias of the estimator noticed but not quantified [8]. Furthermore, the variance of the estimator was provided as a general equation using Delta method and not studied in depth.



More recently, *priority sampling* has been used in [1] to increase the chance of observing a closed wedge in  $g$ . The idea is to give a high priority for edges that close wedges in  $g$ . First technique called GPS-Post estimates wedges and closed wedges at the end of a stream and then uses them to obtain an estimation of  $\mathcal{C}$ . More efficient method called GPS-In estimates wedges and closed wedges in the stream. In contrast to the former method, GPS-In checks the closeness of wedges in  $g$  while scanning the edges. Still such estimators are biased. However, the bias has not been quantified nor corrected.

## 4.3 The bias and variance in a non-streaming model

### 4.3.1 The sampling scheme

Our sampling scheme is based on edge sampling. It selects  $n$  distinct edges from the original graph  $G$  uniformly at random to generate a subgraph  $g$ . When interpreted as a node sampling process, it is the same as PPS sampling, where nodes are sampled with Probability Proportional to Size. In this sense, random walk sampling is an approximation to random edge sampling.

Let  $\Lambda_g$  be the count of wedges in  $g$ , and  $\Delta_g$  denotes the number of closed-wedges restricted to the wedges of  $g$  in which their closenesses are checked based on the original graph  $G$ . The expectations of  $\Lambda_g$  and  $\Delta_g$  are

$$\mathbb{E}(\Lambda_g) = \Lambda p^2, \quad \mathbb{E}(\Delta_g) = \Delta p^2. \quad (7)$$

Hence, the unbiased estimator for  $\Lambda$  and  $\Delta$  [28] are

$$\hat{\Lambda} = \frac{\Lambda_g}{p^2}, \quad \hat{\Delta} = \frac{\Delta_g}{p^2}.$$

Under this sampling scheme, the biased estimator in Eq. 2 is instantiated as

$$\hat{\mathcal{C}} = \frac{\Delta_g}{\Lambda_g}. \quad (8)$$

### 4.3.2 The bias

We quantify the bias using the classic power method [74]. By Taylor expansion, the quadratic approximation of  $x/y$  in the neighbourhood of  $(a, b)$  is

$$\frac{x}{y} \approx \frac{a}{b} + \left( \frac{1}{b}(x-a) - \frac{a}{b^2}(y-b) \right) + \left( \frac{a}{b^3}(y-b)^2 - \frac{1}{b^2}(x-a)(y-b) \right).$$

Applying expectation on both sides of the equation yields:

$$\begin{aligned} \mathbb{E} \left( \frac{x}{y} \right) &\approx \mathbb{E} \left( \frac{a}{b} \right) + \frac{1}{b}(\mathbb{E}(x) - a) - \frac{a}{b^2}(\mathbb{E}(y) - b) \\ &\quad + \frac{a}{b^3}\mathbb{E}(y-b)^2 - \frac{1}{b^2}\mathbb{E}(x-a)(y-b). \end{aligned} \quad (9)$$

Take  $a = \mathbb{E}(\Delta_g)$ ,  $b = \mathbb{E}(\Lambda_g)$ ,  $x = \Delta_g$ , and  $y = \Lambda_g$ . Note that by definition  $cov(\Delta_g, \Lambda_g) = \mathbb{E}((\Delta_g - \mathbb{E}(\Delta_g))(\Lambda_g - \mathbb{E}(\Lambda_g)))$ , and  $var(\Lambda_g) = \mathbb{E}(\Lambda_g - \mathbb{E}(\Lambda_g))^2$ . Eq. 9 can be rewritten as:

$$\mathbb{E} \left( \frac{\Delta_g}{\Lambda_g} \right) \approx \frac{\mathbb{E}(\Delta_g)}{\mathbb{E}(\Lambda_g)} + \frac{\mathbb{E}(\Delta_g)var(\Lambda_g)}{\mathbb{E}(\Lambda_g)^3} - \frac{cov(\Delta_g, \Lambda_g)}{\mathbb{E}(\Lambda_g)^2}. \quad (10)$$

Applying the fact that  $\mathcal{C} = \Delta/\Lambda = \mathbb{E}(\Delta_g)/\mathbb{E}(\Lambda_g)$ , the above equation can be reformulated as

$$\mathbb{E}(\widehat{\mathcal{C}}) \approx \mathcal{C} \left( 1 + \frac{var(\Lambda_g)}{\mathbb{E}(\Lambda_g)^2} - \frac{cov(\Delta_g, \Lambda_g)}{\mathbb{E}(\Delta_g)\mathbb{E}(\Lambda_g)} \right). \quad (11)$$

We can see that the bias hinges on the variance of  $\Lambda_g$  and covariance between  $\Delta_g$  and  $\Lambda_g$ . Before going into the derivation of the variance and covariance, we need to understand the dependency between two wedges, and the dependency between a wedge and a closed-wedge as illustrated in Fig. 4.1. Panel (a) is an example graph. In the graph, two wedges  $(k,j,i)$  and  $(l,j,i)$  share a common edge  $(j,i)$ . We use  $\Psi$  to denote the number of such sharing in a graph. Panel (c) is an example of sharing between a

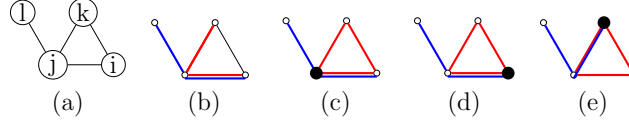


FIGURE 4.1: Illustration of shared wedges and closed-wedges. (a) A sample graph; (b) Wedge  $(l, j, i)$  shares with wedge  $(k, j, i)$ ; (c-d) Wedge  $(l, j, i)$  shares with closed-wedges  $(i, j, k)$ ,  $(k, i, j)$ ; (e) Wedge  $(l, j, k)$  shares with closed-wedge  $(j, k, i)$ . The large node in the plot indicates the centre node of the closed-wedge. E.g., in Panel (c) the closed-wedge is  $(k, j, i)$ .

wedge and a closed-wedge: wedge  $(l, j, i)$  and closed-wedge  $(k, j, i)$  share a common edge  $(j, i)$ . Panels (d) and (e) are similar to (c), except that the center node in the wedges is changed. Here we showcase the following distinction between a closed-wedge and a triangle: a closed-wedge is similar to a triangle except that each triangle has three closed-wedges, and correspondingly, a closed-wedge has a center node. We denote the number of such sharing using  $\Omega$ . In a graph,  $\Psi$  and  $\Omega$  can be very large, much larger than  $\Lambda$  and  $\Delta$ .

Let  $\lambda_i$  be an indicator for the  $i^{\text{th}}$  wedge in the original graph  $G$ . Indicator  $\lambda_i$  is 1 if the wedge is sampled in  $g$ ; otherwise it is 0. Let  $1, 2, \dots, \Lambda$  be the labels for all wedges in  $G$ .  $\Lambda_g = \sum_{i=1}^{\Lambda} \lambda_i$ . By applying  $\text{var}$  on both sides on the equation, we have

$$\begin{aligned} \text{var}(\Lambda_g) &= \text{var}\left(\sum_{i=1}^{\Lambda} \lambda_i\right) = \sum_{i=1}^{\Lambda} \sum_{j=1}^{\Lambda} \text{cov}(\lambda_i, \lambda_j) \\ &= \sum_{i=1}^{\Lambda} \text{var}(\lambda_i) + \sum_{i \neq j} \text{cov}(\lambda_i, \lambda_j). \end{aligned} \quad (12)$$

Variable  $\lambda_i$  follows a Bernoulli distribution with probability  $p^2$  and  $\text{var}(\lambda_i) = p^2 - p^4$ . For the covariance, we need to consider the cases of dependent wedges. Wedges  $\lambda_i$  and  $\lambda_j$  are dependent when they have a shared edge in graph  $G$  as illustrated in Panel (b) of Fig. 4.1. In such a case,  $\mathbb{E}(\lambda_i \lambda_j) = p^3$ , hence  $\text{cov}(\lambda_i, \lambda_j) = \mathbb{E}(\lambda_i \lambda_j) - \mathbb{E}(\lambda_i) \mathbb{E}(\lambda_j) = p^3 - p^4$ . There are  $2\Psi$  cases (each  $\text{cov}(\lambda_i, \lambda_j)$  has an equivalent  $\text{cov}(\lambda_j, \lambda_i)$ ), we have the following by substituting  $\text{var}(\Lambda_i)$  and  $\text{cov}(\lambda_i, \lambda_j)$  in Eq. 12.

$$\text{var}(\Lambda_g) = \Lambda(p^2 - p^4) + 2\Psi(p^3 - p^4). \quad (13)$$

Next we derive  $cov(\Delta_g, \Lambda_g)$ . Let  $\delta_i$  be the indicator for the  $i^{th}$  closed-wedge in graph  $G$ . The covariance between  $\Delta_g$  and  $\Lambda_g$  is

$$\begin{aligned} cov(\Delta_g, \Lambda_g) &= \sum_{i=1}^{\Delta} \sum_{j=1}^{\Lambda} cov(\delta_i, \lambda_j) \\ &= \sum_{i=1}^{\Delta} \sum_{j=1}^{\Lambda} \mathbb{E}(\delta_i \lambda_j) - \mathbb{E}(\delta_i) \mathbb{E}(\lambda_j). \end{aligned}$$

When  $\delta_i$  and  $\lambda_j$  are independent, they share no edges, the covariance between them is  $cov(\delta_i, \lambda_j) = 0$ . There are two cases that  $\delta_i$  and  $\lambda_j$  are dependent: the wedge has either one edge or two edges shared with the closed-wedge. In the first case,  $\mathbb{E}(\delta_i \lambda_j) - \mathbb{E}(\delta_i) \mathbb{E}(\lambda_j) = p^3 - p^4$ . Note that it is not  $p^4 - p^5$  because our sampling method checks the closeness of a wedge whenever a wedge is encountered. Hence the probability of seeing a closed-wedge in a sample is  $p^2$  instead of  $p^3$ . In the second case, the wedge is contained in the closed-wedge, and  $\mathbb{E}(\delta_i \lambda_j) - \mathbb{E}(\delta_i) \mathbb{E}(\lambda_j) = p^2 - p^4$ . Since there are  $\Omega$  number of one-edge sharing, and  $\Delta$  number of two-edge sharing, the covariance is

$$cov(\Delta_g, \Lambda_g) = \Delta(p^2 - p^4) + \Omega(p^3 - p^4). \quad (14)$$

Substitute Eq. 13 and Eq. 14 into 11, and assume that  $1 - p \approx 1 - p^2 \approx 1$  because the sampling probability is very small for large graphs, we obtain

$$\mathbb{E}(\widehat{\mathcal{C}}) \approx \mathcal{C} \left( 1 + \frac{2\mathbb{E}(\Psi_g)}{\mathbb{E}(\Lambda_g)^2} - \frac{\mathbb{E}(\Omega_g)}{\mathbb{E}(\Lambda_g)\mathbb{E}(\Delta_g)} \right). \quad (15)$$

Let relative bias  $RB = \mathbb{E}(\widehat{\mathcal{C}})/\mathcal{C} - 1$ . After rearranging the formula above, and remember that  $\mathbb{E}(\Lambda_g) = \Lambda p^2$ ,  $\mathbb{E}(\Delta_g) = \Delta p^2$ ,  $\mathbb{E}(\Omega_g) = \Omega p^3$ , and  $\mathbb{E}(\Psi_g) = \Psi p^3$ , we quantify the bias as follows.

**Theorem 1.** *The RB of the estimator in Eq. 8 is approximated by*

$$RB \approx \frac{1}{p} \left( \frac{2\Psi}{\Lambda^2} - \frac{\Omega}{\Lambda\Delta} \right), \quad (16)$$

and it can be estimated by

$$\widehat{RB} = \frac{2\Psi_g}{\Lambda_g^2} - \frac{\Omega_g}{\Lambda_g\Delta_g}. \quad (17)$$

Correspondingly, we have the following bias-corrected estimators:

$$\widehat{\mathcal{C}}^* = \frac{\Delta_g}{\Lambda_g} \left[ 1 + \frac{1}{p} \left( \frac{2\Psi}{\Lambda^2} - \frac{\Omega}{\Lambda\Delta} \right) \right]^{-1}, \quad (18)$$

where  $p$  is the sampling probability,  $\Psi$  and  $\Omega$  are the counts for shared wedges and shared wedges and closed-wedges, respectively. The practical estimator based on subgraph only is

$$\widehat{\mathcal{C}}^+ = \frac{\Delta_g}{\Lambda_g} \left[ 1 + \frac{2\Psi_g}{\Lambda_g^2} - \frac{\Omega_g}{\Lambda_g\Delta_g} \right]^{-1}, \quad (19)$$

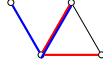
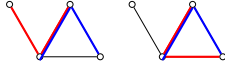
where  $\Psi_g$  and  $\Omega_g$  are the corresponding counts for  $\Psi$  and  $\Omega$  in subgraph  $g$ .

### 4.3.3 Counting $\Psi$ and $\Omega$


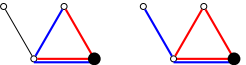
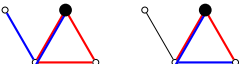
The estimator  $\widehat{\mathcal{C}}^+$  relies on  $\Psi_g$  and  $\Omega_g$ .  $\Psi$  and  $\Omega$  can be computed efficiently, especially for sample graphs that are typically not very large.  $\Psi$  is counted by iterating through all the edges using the following equation:

$$\Psi = \sum_{e(j,k) \in \mathcal{E}} \left[ \binom{d_j-1}{2} + \binom{d_k-1}{2} + (d_j-1)(d_k-1) \right], \quad (20)$$

where  $j$  and  $k$  are end nodes of edge  $e$  and  $d_x$  is the degree of node  $x$  for  $x = j, k$ . This can be verified by looking at the three cases of shared wedges in Panel (a) of Fig. 4.2. For two first cases, there are  $\binom{d_j-1}{2} + \binom{d_k-1}{2}$  shared wedges. For last case,

Cases	Overlap cases	$\Psi$
For node $j$		$\binom{d_j-1}{2} = 1$
For node $k$	Empty	$\binom{d_k-1}{2} = 0$
Between $j$ and $k$		$(d_j - 1)(d_k - 1) = 2$

(a) Example for counting  $\Psi$  by checking edge  $(j, k)$ . Need to repeat the process for every edge in the graph. For edge  $(j, k)$ ,  $\Psi = \binom{d_j-1}{2} + \binom{d_k-1}{2} + (d_j - 1)(d_k - 1) = 3$ .

Closed-wedge	Overlap cases	$\Phi$
$(i, j, k)$		$d_j - 2$ for $(j, k)$ $d_j - 2$ for $(j, i)$
$(j, i, k)$		$d_j - 1$
$(j, k, i)$		$d_j - 1$

(b) An example of counting  $\Omega$  by checking node  $j$ . Need to repeat the process for every node in triangles in the graph. Large nodes indicate the center node of a closed-wedge. For node  $j$ , there are  $2 \times (d_j - 2) + d_j - 1 + d_j - 1 = 4d_j - 6 = 6$  cases of overlaps between a wedge and a closed-wedge.

FIGURE 4.2: An example for computing  $\Psi$  and  $\Omega$  in the sample graph in Fig. 4.1 Panel (a).

there are  $(d_j - 1)(d_k - 1)$  shared wedges.

Metric  $\Omega$  is computed by summarizing the overlaps for each node in all the triangles using the following equation:

$$\Omega = \sum_{(i,j,k) \in \Delta} (4d_j - 6), \quad (21)$$

where  $i, j, k$  are nodes in graph  $G$  and  $(i, j, k)$  is a closed-wedge with center node  $j$ . Note that  $d_j \geq 2$ , hence  $\Omega$  is always a positive value.

The equation can be explained using the example depicted in Panel (b) of Fig. 4.2. For node  $j$ , there are  $d_j - 2$  number of wedges that share edge  $(j,k)$  for the closed-wedge  $(i,j,k)$ . Similarly, there are  $d_j - 2$  number of shared pairs with common edge  $(j,i)$ . Now, we need to look at other two closed-wedges  $(j,i,k)$  and  $(j,k,i)$ . For closed-wedge  $(j,i,k)$ , there are  $d_j - 1$  number of wedges emanating from node  $j$  that share one edge with the closed wedge. Similarly, for closed-wedge  $(j,k,l)$ , there are also  $d_j - 1$  shares. Hence, overall for each node  $j$ , there are  $4d_j - 6$  shared pairs.

### 4.3.4 The variance

Next, we derive the variance of the estimator. To do so, we use the Delta method as follows. Apply  $var$  on the estimator, we get:

$$var(\widehat{\mathcal{C}}) = var\left(\frac{\Delta_g}{\Lambda_g}\right). \quad (22)$$

The approximation of the variance of term  $\Delta_g/\Lambda_g$  in the neighborhood of  $(a, b)$  where  $a = \mathbb{E}[\Delta_g]$ ,  $b = \mathbb{E}[\Lambda_g]$  using Delta method is:

$$var\left(\frac{\Delta_g}{\Lambda_g}\right) \approx \frac{var(\Delta_g)}{b^2} + \frac{a^2 var(\Lambda_g)}{b^4} - \frac{2a cov(\Delta_g, \Lambda_g)}{b^3}. \quad (23)$$

In our sampling scheme  $\mathbb{E}(\Delta_g) = \Delta p^2$ , and  $\mathbb{E}(\Lambda_g) = \Lambda p^2$ ; replace them in Eq. 23, and we get:

$$\begin{aligned} var(\widehat{\mathcal{C}}) &\approx \frac{var(\Delta_g)}{\Lambda^2 p^4} + \frac{\Delta^2 p^4 var(\Lambda_g)}{\Lambda^4 p^8} - \frac{2\Delta p^2 cov(\Delta_g, \Lambda_g)}{\Lambda^3 p^6} \\ &= \frac{var(\Delta_g)}{\Lambda^2 p^4} + \frac{\Delta^2 var(\Lambda_g)}{\Lambda^4 p^4} - \frac{2\Delta cov(\Delta_g, \Lambda_g)}{\Lambda^3 p^4}. \end{aligned} \quad (24)$$

The variance formula in Eq. 24 is complicated and depends on several parameters. To have better insight into it, we present the variance in the form of relative standard

error ( $RSE = \sqrt{\text{var}}/\mathcal{C}$ ). Replace  $\mathcal{C} = \Delta/\Lambda$ ; we get the RSE as:

$$\begin{aligned} RSE(\widehat{\mathcal{C}}) &\approx \left[ \frac{\Lambda^2 \text{var}(\Delta_g)}{\Delta^2 \Lambda^2 p^4} + \frac{\Lambda^2 \Delta^2 \text{var}(\Lambda_g)}{\Delta^2 \Lambda^4 p^4} - \frac{2\Lambda^2 \Delta \text{cov}(\Delta_g, \Lambda_g)}{\Delta^2 \Lambda^3 p^4} \right]^{1/2} \\ &= \left[ \frac{\text{var}(\Delta_g)}{\Delta^2 p^4} + \frac{\text{var}(\Lambda_g)}{\Lambda^2 p^4} - \frac{2\text{cov}(\Delta_g, \Lambda_g)}{\Delta \Lambda p^4} \right]^{1/2}. \end{aligned} \quad (25)$$

The RSE relies on the variance of  $\Delta_g$  and  $\Lambda_g$ , and the covariance between them. We derived terms  $\text{var}(\Lambda_g)$  and  $\text{cov}(\Delta_g, \Lambda_g)$  in section 4.3.2. We also use the result in [28], to rewrite the variance of  $\text{var}(\Delta_g)$  as:

$$\text{var}(\Delta_g) = \Delta(p^2 - p^4) + 8\Phi(p^3 - p^4). \quad (26)$$

Variable  $\Phi$  stands for the number of distinct pairs of dependent triangles in the original graph. We refer to [28] for more details.

When the input graph is large, sampling probability  $p$  is very small. Thus,  $1 - p$  and  $1 - p^2$  is approximated by 1. Therefore,  $\text{var}(\Delta_g)$ ,  $\text{var}(\Lambda_g)$  and  $\text{cov}(\Delta_g, \Lambda_g)$  are approximated as follows.

$$\text{var}(\Delta_g) \approx \Delta p^2 + 8\Phi p^3, \quad (27)$$

$$\text{var}(\Lambda_g) \approx \Lambda p^2 + 2\Psi p^3, \quad (28)$$

$$\text{cov}(\Delta_g, \Lambda_g) \approx \Delta p^2 + \Omega p^3. \quad (29)$$

Substitute Eq. 27, Eq. 28, and Eq. 29 in the RSE formula we get:

$$RSE(\widehat{\mathcal{C}}) \approx \left[ \frac{1}{\Delta p^2} + \frac{8\Phi p^3}{\Delta^2 p^4} + \frac{2\Psi p^3}{\Lambda^2 p^4} - \frac{1}{\Lambda p^2} - \frac{2\Omega p^3}{\Delta \Lambda p^4} \right]^{1/2}. \quad (30)$$

The RSE formula in Eq. 30 depends on the properties of the original graph  $G$ . Therefore, practitioners need to use the properties of sampled graph  $g$  to get idea about the error bound of the estimator. To do so, substitute  $\mathbb{E}(\Delta_g) = \Delta p^2$ ,  $\mathbb{E}(\Lambda_g) = \Lambda p^2$ ,  $\mathbb{E}(\Phi_g) = 4\Phi p^3$ ,  $\mathbb{E}(\Psi_g) = \Psi p^3$ , and  $\mathbb{E}(\Omega_g) = \Omega p^3$ ; after simplifying



**Algorithm 3:** Naive edge streaming (NES)

---

```

Input:  $p$ 
Output:  $\widehat{\Delta}$ ,  $RSE(\widehat{\Delta})$ 
1 begin
2    $\Delta_g = 0, \Lambda_g = 0, g = \phi.$ 
3   while new edge  $e$  do
4     Add  $e$  into  $g$  with probability  $p.$ 
5     foreach wedge  $w$  formed using  $e$  and edges in  $g$  do
6        $\Lambda_{g^+} = 1.$ 
7     end
8     foreach wedge  $w \in g$  closed by  $e$  do
9        $\Delta_{g^+} = 1.$ 
10    end
11  end
12   $\widehat{C} = 3\Delta_g(p\Lambda_g)^{-1}; \widehat{RSE}(\widehat{C}) \approx \Delta_g^{-1/2}.$ 
13 end

```

---

the result; we obtain:

$$RSE(\widehat{C}) \approx \left[ \frac{1}{\Delta_g} + \frac{2\Phi_g}{\Delta_g^2} + \frac{2\Psi_g}{\Lambda_g^2} - \frac{1}{\Lambda_g} - \frac{2\Omega_g}{\Delta_g\Lambda_g} \right]^{1/2}. \quad (31)$$

Obviously, the first term in above equation is dominant. Thus, we have:

**Theorem 2.** *The RSE of  $\widehat{C}$  is obtained by*

$$\widehat{RSE}(\widehat{C}) \approx \Delta_g^{-1/2}. \quad (32)$$

## 4.4 The bias and variance in a streaming model

### 4.4.1 Naive edge streaming (NES)

Having obtained the bias and variance estimators for random edge sampling, we ponder whether the results can be extended to other sampling methods. The answer is positive– the results are generic with small changes. As an example, we expand the random edge sampling slightly to its streaming counterpart, which we call Naive Edge Streaming (NES) as described in Alg. 3. It is also a special case of [1], where

all edges are sampled with the same priority.

In NES, edges arrive in an arbitrary sequence over an edge stream of the original graph  $G$ . Over the stream, NES samples some edges uniformly at random with an equal probability  $p$ , and adds them into the subgraph  $g$ . Once a new edge  $e$  arrives, the number of wedges closed by  $e$  in  $g$  is counted. Let  $\Delta_g$  denote the number of such closed wedges observed in the stream. At the same time, the number of wedges formed using  $e$  and edges in  $g$  is enumerated. Let  $\Lambda_g$  be the total number of such wedges. Then,  $\mathcal{C}$  is estimated using  $\Delta_g$  and  $\Lambda_g$ . A key difference with our non-streaming method is that  $\Delta_g$  and  $\Lambda_g$  are counted along the streaming process, not after the sampling.

Given a subgraph  $g$  of  $G$ , for every wedge  $(u, v, w)$  in  $g$ , we check whether or not  $(u, w)$  is in the rest of the stream. For every closed wedge in  $G$ , if its two edges are sampled in  $g$ , the probability of observing the third edge in the rest of the stream is  $\frac{1}{3}$ . Thus, the probability of identifying a closed wedges by NES is  $\frac{1}{3}p^2$ . Let  $\delta_i$  be an indicator for the  $i^{th}$  closed wedge in  $G$ . Variable  $\delta_i$  is 1 when two edges of the  $i^{th}$  closed wedge are sampled and the third edge is observed in the rest of the stream; otherwise it is 0. Recall that  $\Delta_g$  is the number of closed wedges identified by NES over a single pass on an edge stream of  $G$ . The expectation of  $\Delta_g$  is:  $\mathbb{E}(\Delta_g) = \mathbb{E}(\sum_{i=1}^{\Delta} \delta_i) = \sum_{i=1}^{\Delta} \mathbb{E}(\delta_i) = \sum_{i=1}^{\Delta} \frac{1}{3}p^2 = \frac{1}{3}p^2\Delta$ . Thus, an unbiased estimator for  $\Delta$  is given by  $\hat{\Delta} = \frac{3\Delta_g}{p^2}$ . Similar estimators for  $\Delta$  have also been proposed in [19, 20, 28].

Next, we give an unbiased estimator for the number of wedges in  $G$ . To sample a wedge by NES, one of its two edges needs to be added into  $g$  with probability  $p$ , and its second edge is required to be observed in the rest of the stream. Thus, the probability of identifying a wedge based on  $g$  is  $p$ . Suppose  $\lambda_i$  is an indicator for the  $i^{th}$  wedge in the input graph. Clearly,  $\lambda_i$  is 1 when its two edges are observed; otherwise it is 0. Recall that  $\Lambda_g$  is the number of wedges identified based on  $g$  by NES. Its expectation is  $\mathbb{E}(\Lambda_g) = \mathbb{E}(\sum_{i=1}^{\Lambda} \lambda_i) = \sum_{i=1}^{\Lambda} \mathbb{E}(\lambda_i) = \sum_{i=1}^{\Lambda} p = p\Lambda$ . Thus, an unbiased estimation for  $\Lambda$  is given by  $\hat{\Lambda} = \frac{\Lambda_g}{p}$ .

Now we can use the unbiased estimators for  $\Delta$  and  $\Lambda$  to estimate  $\mathcal{C}$ . Although

both  $\widehat{\Delta}$  and  $\widehat{\Lambda}$  are unbiased, the following estimator is **biased** as we will correct it later in this chapter.

$$\widehat{\mathcal{C}} = \frac{\widehat{\Delta}}{\widehat{\Lambda}} = \frac{3p\Delta_g}{p^2\Lambda_g} = \frac{3\Delta_g}{p\Lambda_g}. \quad (33)$$

#### 4.4.2 Estimator of the variance

We derive the variance of the estimator using the Delta method. Applying *var* on Eq. 33, we get:

$$\text{var}(\widehat{\mathcal{C}}) = \text{var}\left(\frac{3\Delta_g}{p\Lambda_g}\right) = \frac{9}{p^2} \text{var}\left(\frac{\Delta_g}{\Lambda_g}\right). \quad (34)$$

Applying Taylor expansion in the neighbourhood of  $(a, b)$ , we have:

$$\text{var}\left(\frac{\Delta_g}{\Lambda_g}\right) \approx \frac{1}{b^2} \text{var}(\Delta_g) + \frac{a^2}{b^4} \text{var}(\Lambda_g) - \frac{2a}{b^3} \text{cov}(\Delta_g, \Lambda_g).$$

Let  $a = \mathbb{E}(\Delta_g)$  and  $b = \mathbb{E}(\Lambda_g)$ , and using the fact that  $\mathbb{E}(\Delta_g) = \frac{1}{3}\Delta p^2$ , and  $\mathbb{E}(\Lambda_g) = \Lambda p$ , we obtain the variance and present it in the form of relative standard error ( $\text{RSE} = \sqrt{\text{var}}/\mathcal{C}$ ) as follows:

$$\text{RSE}(\widehat{\mathcal{C}}) \approx \left[ \frac{9 \text{var}(\Delta_g)}{\Delta^2 p^4} + \frac{\text{var}(\Lambda_g)}{\Lambda^2 p^2} - \frac{6 \text{cov}(\Delta_g, \Lambda_g)}{\Delta \Lambda p^3} \right]^{1/2}.$$

The RSE depends on the variance of  $\Delta_g$  and  $\Lambda_g$ , and the covariance between them. Note that this is where [1, 8] stops. We continue the derivation of the variances and covariances in Appendix. When the networks are large,  $p$  is a very small value. Hence, we can assume that  $1 - p \approx 1 - p^2 \approx 1$ , and the results of the Lemmas 1, 2, and 3 are simplified as follows:

$$\text{var}(\Delta_g) \approx \frac{1}{3}\Delta p^2 + \frac{16}{15}\Phi p^3, \quad (35)$$

$$\text{var}(\Lambda_g) \approx \frac{2}{3}\Psi p, \quad (36)$$

$$\text{cov}(\Delta_g, \Lambda_g) \approx \frac{5}{12}\Omega' p^2. \quad (37)$$

Here  $\Phi$  is the number of pairs of dependent triangles,  $\Psi$  is the number of pairs of shared wedges, and  $\Omega'$  is the number of pairs of wedges and triangles with one common edge in the original graph  $G$ . Replace Eq. 35, Eq. 36, and Eq. 37 in the RSE above; we obtain the RSE as:

$$\begin{aligned} RSE(\widehat{\mathcal{C}}) &\approx \left[ \frac{9 \left( \frac{1}{3} \Delta p^2 + \frac{16}{15} \Phi p^3 \right)}{\Delta^2 p^4} + \frac{\frac{2}{3} \Psi p}{\Lambda^2 p^2} - \frac{6 \left( \frac{5}{12} \Omega' p^2 \right)}{\Delta \Lambda p^3} \right]^{1/2} \\ &= \left[ \frac{3}{\Delta p^2} + \frac{48 \Phi p^3}{5 \Delta^2 p^4} + \frac{2 \Psi p}{3 \Lambda^2 p^2} - \frac{5 \Omega' p^2}{2 \Delta \Lambda p^3} \right]^{1/2}. \end{aligned} \quad (38)$$

The RSE in Eq. 38 depends on the properties of the original graph  $G$ , i.e.  $\Delta$ ,  $\Lambda$ ,  $\Phi$ ,  $\Psi$ , and  $\Omega'$ . Note that those properties are unknown for the third party. However, practitioners need to know the error bound of the estimator using the properties in the sample. To do so, we give the estimation of the variance as follows.

Based on NES sampling scheme we have:  $\mathbb{E}(\Delta_g) = \frac{1}{3} \Delta p^2$ ,  $\mathbb{E}(\Lambda_g) = \Lambda p$ ,  $\mathbb{E}(\Phi_g) = \frac{8}{15} \Phi p^3$ ,  $\mathbb{E}(\Psi_g) = \frac{2}{6} \Psi p$ , and  $\mathbb{E}(\Omega'_g) = \frac{5}{12} \Omega' p^2$ . Thus, substitute  $\Delta$ ,  $\Lambda$ ,  $\Phi$ ,  $\Psi$ , and  $\Omega'$  in Eq. 38 by their estimations, we obtain the estimator of the RSE as:

$$\widehat{RSE}(\widehat{\mathcal{C}}) \approx \left[ \frac{1}{\Delta_g} + \frac{2 \Phi_g}{\Delta_g^2} + \frac{2 \Psi_g}{\Lambda_g^2} - \frac{2 \Omega'_g}{\Delta_g \Lambda_g} \right]^{1/2}. \quad (39)$$

where  $\Phi_g$  is the number of pairs of dependent triangles,  $\Psi_g$  is the number of pairs of shared wedges, and  $\Omega'_g$  is the number of pairs of wedges and triangles with one common edge observed based on subgraph  $g$ . The RSE in Eq. 39 hangs on several variables in the sample, i.e.  $\Delta_g$ ,  $\Lambda_g$ ,  $\Phi_g$ ,  $\Psi_g$ , and  $\Omega'_g$ . To have better understating the RSE of the estimator, we need to simplify it further. We claim that when sampling probability  $p$  is small, the first term in Eq. 39, i.e.  $\Delta_g^{-1}$ , is dominant. Therefore, we give the following Theorem as a simplified estimator for the RSE of  $\mathcal{C}$ .

**Theorem 3.** *When  $p$  is small, the RSE of  $\widehat{\mathcal{C}}$  is approximated by*

$$\widehat{RSE}(\widehat{\mathcal{C}}) \approx \Delta_g^{-1/2}. \quad (40)$$

### 4.4.3 The bias-corrected estimator

To quantify the bias, we apply the expectation on  $\widehat{\mathcal{C}}$ :

$$\mathbb{E}(\widehat{\mathcal{C}}) = \mathbb{E}\left(\frac{3\Delta_g}{p\Lambda_g}\right) = \frac{3}{p}\mathbb{E}\left(\frac{\Delta_g}{\Lambda_g}\right). \quad (41)$$

The approximation of  $\mathbb{E}(\Delta_g/\Lambda_g)$  using the quadratic Taylor expansion of  $\Delta_g/\Lambda_g$  in the neighborhood of  $(a, b)$  is:

$$\mathbb{E}\left(\frac{\Delta_g}{\Lambda_g}\right) \approx \frac{a}{b} + \frac{a}{b^3}\text{var}(\Lambda_g) - \frac{1}{b^2}\text{cov}(\Delta_g, \Lambda_g). \quad (42)$$

Replace  $a = \mathbb{E}(\Delta_g)$  and  $b = \mathbb{E}(\Lambda_g)$  and take  $\mathbb{E}(\Delta_g) = \Delta p^2/3$  and  $\mathbb{E}(\Lambda_g) = \Lambda p$ ; substitute Eq. 42 in Eq. 41, and use  $\mathcal{C} = \Delta/\Lambda$  we obtain:

$$\mathbb{E}(\widehat{\mathcal{C}}) \approx \mathcal{C}\left(1 + \frac{\text{var}(\Lambda_g)}{\Lambda^2 p^2} - \frac{3\text{cov}(\Delta_g, \Lambda_g)}{\Delta \Lambda p^3}\right). \quad (43)$$

To quantify the bias of  $\widehat{\mathcal{C}}$ , the relative bias, i.e.  $RB = (\mathbb{E}(\widehat{\mathcal{C}}) - \mathcal{C})/\mathcal{C}$ , is used. By substituting Eq. 43 in RB, and simplifying the result we obtained:

$$RB \approx \frac{\text{var}(\Lambda_g)}{\Lambda^2 p^2} - \frac{3\text{cov}(\Delta_g, \Lambda_g)}{\Delta \Lambda p^3}. \quad (44)$$

The bias depends on the variance of  $\Lambda_g$  and the covariance between  $\Delta_g$  and  $\Lambda_g$ . Using the same treatment in the variance of  $\widehat{\mathcal{C}}$  for the  $\text{var}(\Lambda_g)$  and  $\text{cov}(\Delta_g, \Lambda_g)$ , i.e. Eq. 36 and Eq. 37, and replacing them in Eq. 44 we have:

$$RB \approx \frac{2\Psi p}{3\Lambda^2 p^2} - \frac{5\Omega' p^2}{4\Delta \Lambda p^3}. \quad (45)$$

Thus, we summarize our result in the following theorem.

**Theorem 4.** *The RB of the estimator in Eq. 33 is approximated by*

$$RB \approx \frac{1}{p} \left[ \frac{2\Psi}{3\Lambda^2} - \frac{5\Omega'}{4\Delta\Lambda} \right]. \quad (46)$$

and its estimation is:

$$\widehat{RB} \approx \frac{2\Psi_g}{\Lambda_g^2} - \frac{\Omega'_g}{\Delta_g\Lambda_g}. \quad (47)$$

Using  $\widehat{RB}$ , a bias-corrected estimator of  $\widehat{\mathcal{C}}$  is

$$\widehat{\mathcal{C}}^+ = \frac{\widehat{\mathcal{C}}}{1 + \widehat{RB}}. \quad (48)$$

The bias-corrected estimator depends on the structure of sampled graph  $g$ . Therefore, using a single pass over an edge stream of the input graph we need to count  $\Delta_g$ ,  $\Lambda_g$ ,  $\Psi_g$ , and  $\Omega'_g$ . Recall that  $\Psi_g$  is the number of pairs of shared wedges and  $\Omega'_g$  is the number of pairs of wedges and triangles with one common edge observed based on subgraph  $g$ . Note that  $\Omega' = (\Omega - 6\Delta)/2$ .

#### 4.4.4 The variances of $\Delta_g$ and $\Lambda_g$ and their covariance

This section presents the derivations of the variances of  $\Delta_g$  and  $\Lambda_g$  and their covariance used to estimate the variance of  $\widehat{\mathcal{C}}$  and its bias.

**Lemma 1.** *Let  $\Delta_g$  be the number of closed wedges identified based on  $g$  using NES. The variance of  $\Delta_g$  is*

$$\text{var}(\Delta_g) = \frac{1}{3} \left( \Delta(p^2 - \frac{1}{3}p^4) + 8\Phi(\frac{2}{5}p^3 - \frac{1}{3}p^4) \right). \quad (49)$$

*Proof.*  $\text{var}(\Delta_g) = \text{var}(\sum_{i=1}^{\Delta} \delta_i) = \sum_{i=1}^{\Delta} \text{var}(\delta_i) + \sum_{i \neq j} \text{cov}(\delta_i, \delta_j)$ . The probability of identifying a closed wedge by NES is  $\frac{1}{3}p^2$ . Hence, the cost of the variance term is  $\Delta(\frac{1}{3}p^2 - \frac{1}{9}p^4)$ . The covariance between two independent closed wedges is zero. Thus, we need to find the covariance between dependent closed wedges. The probability of identifying such

a dependent case is  $\frac{2}{15}p^3$ . Recall that the total number of pairs of shared triangles is denoted by  $\Phi$ . For each pair of shared triangles there are four dependent closed wedges. Thus, in the summation above, there are  $8\Phi$  pairs of shared closed wedges. Thus, the covariance is  $8\Phi(\frac{2}{15}p^3 - \frac{1}{9}p^4)$ . Adding the costs, we obtain the Lemma.

**Lemma 2.** *Let  $\Lambda_g$  be the number of wedges identified based on  $g$  using NES. The variance of  $\Lambda_g$  is obtained by*

$$\text{var}(\Lambda_g) = \Lambda(p - p^2) + \frac{2}{3}\Psi(p - p^2). \quad (50)$$

*Proof.*  $\text{var}(\Lambda_g) = \text{var}(\sum_{i=1}^{\Lambda} \lambda_i) = \sum_{i=1}^{\Lambda} \sum_{j=1}^{\Lambda} \text{cov}(\lambda_i, \lambda_j)$  The probability of identifying a wedge by NES is  $p$ . When  $i = j$  holds, the covariance term is equal to the variance of  $\lambda_i$ , which is  $p - p^2$ . Thus, we get  $\text{var}(\hat{\Lambda}) = \Lambda(p - p^2) + \sum_{i \neq j} \text{cov}(\lambda_i, \lambda_j)$ . Next, we need to understand the covariance between two wedges. When the two wedges  $w_i$  and  $w_j$  are independent the covariance between them is zero. Hence, we need to consider the covariance between dependent wedges. The probability of identifying a pair of dependent wedges is  $p$ . Thus, the covariance between two shared closed wedges is  $(p - p^2)$ . Suppose  $\Psi$  is the number of pairs of dependent wedges in  $G$ . The chance to identify such a dependent case is  $1/3$ . Because  $\text{cov}(\lambda_i, \lambda_j) = \text{cov}(\lambda_j, \lambda_i)$ , we need to multiply the covariance term by 2. Thus, the lemma is proved.

**Lemma 3.** *Covariance between  $\Delta_g$  and  $\Lambda_g$  is given by*

$$\text{cov}(\Delta_g, \Lambda_g) = 2\Delta(p^2 - p^3) + \frac{5}{12}\Omega'(p^2 - p^3). \quad (51)$$

*Proof.*  $\text{cov}(\Delta_g, \Lambda_g) = \sum_{i=1}^{\Delta} \sum_{j=1}^{\Lambda} \text{cov}(\delta_i, \lambda_j)$ . Suppose  $\Omega'$  is the exact number of pairs of wedges and triangles with one common edge in  $G$ . The sampling probabilities of such a pair is  $p^2$ . Moreover, the chance to identify such a pair in the streaming model by NES is  $5/12$ . Therefore, the total cost is  $\frac{5}{12}\Omega'(p^2 - p^3)$ . In addition, each closed wedge of a triangle shares an edge with the other two wedges in the triangle. The total cost for such cases is  $6\Delta(\frac{1}{3}p^2 + \frac{1}{3}p^3)$ . Therefore, the lemma thus follows.

## 4.5 Experiments

### 4.5.1 Datasets

The bias and variance phenomenon varies greatly from graph to graph. To find out the patterns behind, we need to experiment extensively with many different kinds of graphs. In total, we use 56 real graphs from a variety of areas such as online social networks, web graphs, Co-authorship, and citation networks. The graph size also varies from about  $4 \times 10^3$  (very small) to  $65 \times 10^6$  (very large). The directionality of directed graphs is ignored and self-edges are removed. The properties of the graphs are summarized in Table 4.2. The codes along with the intermediate data are available on the website <http://cs.uwindsor.ca/~etemadir/cc>. We used two servers each with 24 cores and 256 GB RAM to calculate ground truth for weeks for large graphs.

### 4.5.2 The bias in the non-streaming model

First, we demonstrate the existence of bias using Fig. 4.3. In the plot, the observed bias is obtained by repeating the estimation for  $5 \times 10^4$  times except for the very large datasets with  $10^4$  repetitions. X-axes are sampling probability  $p$ . We can see that the range of  $p$  varies from data to data. We do not set a fixed range of  $p$  because, for different data sizes, the required sampling probability is different to achieve the same accuracy. Larger data normally requires smaller sampling probability. Hence, we fix the RSE (Relative Standard Error) to be within the range of 0.1 to 0.4. Then,  $p$  is derived from RSE using the formula provided in [28].

We can use Eq. 16 to interpret our experiments. From the equation, we can tell that the bias depends on the sampling probability  $p$ . Thus, we can expect that the bias diminishes with the increase of sample size, as verified by all the datasets. When the graph is very large,  $p$  could be very small to achieve accurate estimation. For instance, in WebArabic,  $p$  is in the order of  $10^{-5}$  to achieve 95% confidence interval  $\mathcal{C} \pm 0.1\mathcal{C}$ . Secondly, the bias depends on the structure of the graph characterized by  $r = \frac{2\Psi}{\Lambda^2} - \frac{\Omega}{\Lambda\Delta}$ . Empirically,  $r$  is a small value that ranges from  $10^{-5}$  to  $10^{-9}$  among 56



TABLE 4.2: Properties of the networks in our experiments, sorted by graph size  $N$ .

Dataset	$N(\times 10^6)$	$\langle d \rangle$	$\mathcal{C}$	$\Delta(\times 10^6)$	$\Lambda(\times 10^9)$	$\langle d^3 \rangle(\times 10^6)$	$\langle d^2 \rangle(\times 10^3)$	$\frac{2\psi}{\Lambda^2} - \frac{\Omega}{\Delta\Lambda}$	$\frac{2\psi}{\Lambda^2}$	$\frac{\Omega}{\Delta\Lambda}$	Description
Ego-facebook [61]	0.004	43.69	0.519	4.8	0.009	1.09	4.6	1.2e-5	7.4e-5	6.2e-5	OSN
CA-GrQc [61]	0.005	5.52	0.629	0.1	0.0002	0.003	0.09	-7.0e-5	5.2e-4	5.9e-4	Collaboration
Wiki-vote [61]	0.007	28.32	0.125	1.8	0.014	1.2	4.1	-3.7e-6	5.9e-5	6.3e-5	OSN
AstroPh [62]	0.01	21.10	0.31	4.0	0.012	0.17	1.3	4.8e-6	3.2e-5	2.7e-5	Citation
CA-CondMat [61]	0.02	8.08	0.264	0.5	2	0.01	0.1	2.5e-5	8.3e-5	5.8e-5	Coauthorship
HepPh [62]	0.02	224.14	0.279	587	2	184	149	1.0e-7	1.8e-6	1.7e-6	Coauthorship
Enron-email [62]	0.03	10.02	0.085	2	0.025	0.8	1.4	1.6e-5	5.2e-5	3.5e-5	E-communication
Brightkite [61]	0.05	7.35	0.110	1.4	0.013	0.15	0.4	1.8e-5	5.8e-5	3.9e-5	OSN
Facebook [62]	0.06	25.64	0.147	10.5	0.07	0.43	2.2	1.0e-6	8.3e-6	7.2e-6	OSN
Epinions [62]	0.07	10.69	0.065	4.8	0.07	1.3	1.9	3.5e-6	2.3e-5	1.9e-5	OSN
Slashdot-Zoo [62]	0.07	11.82	0.023	1.6	0.06	1.1	1.7	5.2e-6	2.2e-5	1.7e-5	OSN
Prosper [62]	0.08	74.60	0.003	3.4	1.1	30	24	-1.9e-7	2.6e-6	2.8e-6	Interaction
Livemocha [62]	0.1	42.13	0.014	10.0	0.716	12	13	-2.5e-7	3.2e-6	3.5e-6	OSN
Douban [62]	0.1	4.22	0.01	0.1	0.011	0.01	0.1	-4.7e-6	1.6e-5	2.1e-5	OSN
Gowalla [61]	0.1	9.66	0.023	6.8	0.290	24	2.9	4.9e-5	5.6e-5	7.1e-6	OSN
Libimseti [62]	0.2	155.97	0.007	207	28	2,203	255	1.3e-7	6.5e-7	5.1e-7	OSN
Digg [62]	0.2	11.07	0.061	42	0.69	15	4.9	5.1e-6	9.7e-6	4.6e-6	OSN
Web-Stanford [62]	0.2	14.13	0.008	33	3.94	536	27	7.1e-6	9.7e-6	2.5e-6	Web graph
Dblp-Coau [61]	0.3	6.62	0.306	6	0.021	0.008	0.1	-4.5e-7	8.3e-6	8.7e-6	Coauthorship
Web-NotreDame [61]	0.3	6.69	0.087	26	0.304	8.3	1.8	2.5e-5	2.9e-5	4.5e-6	Web graph
Amazon [61]	0.3	5.53	0.205	2	0.009	0.002	0.06	5.5e-6	1.0e-5	5.2e-6	Co-purchasing
Actor [62]	0.3	78.68	0.166	1,040	6.26	36	32	5.1e-8	5.3e-7	4.8e-7	Collaboration
Citeseer [62]	0.3	9.03	0.049	4	0.081	0.14	0.4	5.1e-6	8.8e-6	3.6e-6	Citation
Dogster [62]	0.4	40.03	0.014	250	17	1,742	82	1.5e-6	2.4e-6	8.7e-7	OSN
Catster [62]	0.6	50.32	0.028	1,969	69	11,637	222	1.2e-6	1.5e-6	2.2e-7	OSN
Web-Berkeley [62]	0.6	19.40	0.0069	194	27.9	3,348	81	2.2e-6	2.9e-6	6.5e-7	Web graph
Web-Google [62]	0.8	9.87	0.055	40	0.727	4.5	1.6	6.3e-6	7.5e-6	1.1e-6	Web graph
Youtube [61]	1.1	5.27	0.006	9	1	30	2.6	1.2e-5	1.5e-5	3.9e-6	OSN
Dblp [62]	1.3	8.16	0.170	36	0.214	0.067	0.3	1.2e-6	2.4e-6	1.1e-6	Coauthorship
Hypes [62]	1.4	3.96	0.001	2	1.4	45	2	2.9e-5	3.0e-5	9.5e-7	OSN
Wiki-Polish [62]	1.5	55.17	0.01	3,402	308	81,387	404	1.2e-6	1.3e-6	6.5e-8	Web graph
Trec-wt10g [62]	1.6	8.33	0.014	63	4.3	63	5.4	4.3e-6	5.3e-6	1.0e-6	Web graph
Wiki-Portuguese [62]	1.6	48.19	0.022	3,798	170	17,635	213	9.1e-7	9.7e-7	5.5e-8	Web graph
Wiki-Japanese [62]	1.6	69.82	0.021	3,863	180	15,595	223	6.9e-7	7.7e-7	8.0e-8	Web graph
Pokec [62]	1.6	27.31	0.046	97	2.08	3.8	2.5	1.2e-6	1.5e-6	2.3e-7	OSN
As-skitter [61]	1.6	13.08	0.005	86	16	341	18	1.5e-6	2.2e-6	6.9e-7	Internet topology
Wiki-Italian [62]	1.8	72.90	0.024	9,419	388	47,127	416	5.3e-7	5.8e-7	4.6e-8	Web graph
Wiki-En [62]	1.8	39.05	0.003	379	122.9	10,112	131	9.6e-7	1.2e-6	2.9e-7	Web graph
Hudong [62]	1.9	14.54	0.003	64	18.7	358	18	1.7e-6	2.0e-6	3.3e-7	Web graph
Hollywood [63, 64]	1.9	24.51	0.152	614	4	1.5	4	-4.4e-9	3.01e-7	3.06e-7	OSN
Baidu [62]	2.1	15.89	0.002	75	30.8	1,600	28	3.0e-6	3.6e-6	5.9e-7	Web graph
Flicker [62]	2.3	19.83	0.107	2,512	23	84	20	7.5e-8	4.4e-7	3.6e-7	OSN
Flixster [62]	2.5	6.27	0.013	23	1.7	0.88	1.3	6.0e-8	8.3e-7	7.7e-7	OSN
Wiki-Russian [62]	2.8	44.20	0.015	5,697	370	39,457	259	7.7e-7	8.2e-7	5.1e-8	Web graph
Wiki-French [62]	3.0	55.21	0.015	6,843	455	35,771	301	4.7e-7	5.2e-7	4.9e-8	Web graph
Orkut [62]	3.0	76.28	0.041	1,882	45	194	29	2.2e-7	3.0e-7	8.1e-8	OSN
Wiki-German [62]	3.2	40.77	0.0088	2,899	328	40,234	203	1.1e-6	1.2e-6	5.3e-8	Web graph
USpatent [62]	3.7	8.75	0.067	22	0.33	0.011	0.1	7.2e-8	5.2e-7	4.5e-7	Citation
LiveJournal [61]	3.9	17.35	0.125	533	4	3.1	2.1	5.0e-7	7.7e-7	2.7e-7	OSN
Orkut2 [63, 64]	11	56.80	0.0002	669	2,543	36,715	441	2.3e-8	6.6e-8	4.3e-8	OSN
DBpedia [62]	18	13.89	0.0016	986	583	19,199	63	9.7e-7	1.0e-6	5.6e-8	Web graph
Web-Arabic [63, 64]	22	48.70	0.031	110,686	3,531	86,644	310	1.5e-7	1.5e-7	4.7e-9	Web graph
Gsh-2015 [63, 64]	29	9.18	0.007	1,169	164	1,000	11	9.9e-7	1.1e-6	1.1e-7	Web graph
Twitter [62]	41	57.74	0.0008	104,474	123,435	5,659,930	5,927	1.8e-8	2.0e-8	1.4e-9	OSN
MicrosoftAc.G. [65]	46	22.61	0.015	1,734	115	203	4.9	7.1e-7	7.2e-7	1.07e-8	Citation
Friendster [62]	65	55.06	0.017	12,521	720	23	22	1.5e-9	4.3e-9	2.8e-9	OSN

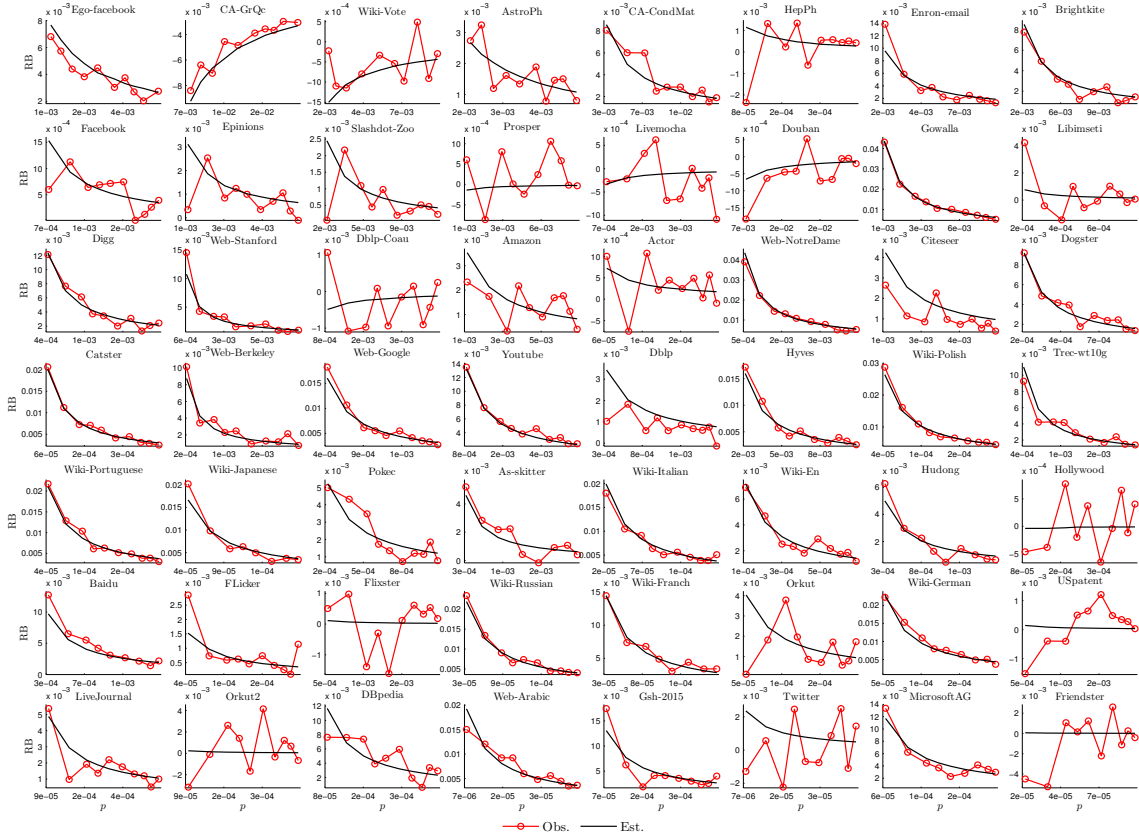


FIGURE 4.3: The observed vs. estimated RB of  $\hat{\mathcal{C}}$ . The results were obtained over  $5 \times 10^4$  independent runs for all graphs except for the large graphs in the last row with  $10^4$  independent repetitions. The estimated RBs are obtained using Eq. 17.

graphs we explored.

To verify our estimated bias, we plot the estimated bias given in Eq. 17 side by side with the observed bias. Overall, the observed and estimated biases fit well. Observed bias fluctuates for some data sets, because of the low bias (hence high variation) of the estimations. For graphs with larger bias (e.g.,  $RB > 1\%$ ), our equation fits the observed RB smoothly. This confirms that two approximations made during the derivation are valid, i.e. 1) it is good enough to take the quadratic expansion of the Taylor expansion; 2) It is valid to assume that  $1 - p \approx 1$ .

The most important result of this research work is the bias-corrected estimator  $\hat{\mathcal{C}}^+$ . Fig. 4.4 compares the RB of  $\hat{\mathcal{C}}^+$  and  $\hat{\mathcal{C}}$ . We can see that  $\hat{\mathcal{C}}^+$  corrects the bias consistently for all the datasets. For the same reason explained above, RBs fluctuates because the bias is very small, hence we see the large variation. For data sets where

bias is large (above 1%), such as Gowalla, Web-Stanford, Web-NotreDame, Web-Google, and all the graphs from Wiki, RBs are more smooth.

Fig. 4.5 gives another perspective for understanding Eq. 16. This time we put 56 data sets in one plot, and demonstrate how good Eq. 16 is to quantify the bias. Panel (A) plots observed RB against the estimated ones. The observed RB is taken for anticipated RSE=0.2. We can see that observed RBs fit Eq. 16 well. It is not a perfectly straight line because the estimation varies for each run. There are a few data sets that have their relative biases larger than 1%. In most cases, the RB is very small value that is close to zero. In some cases, the bias is negative.

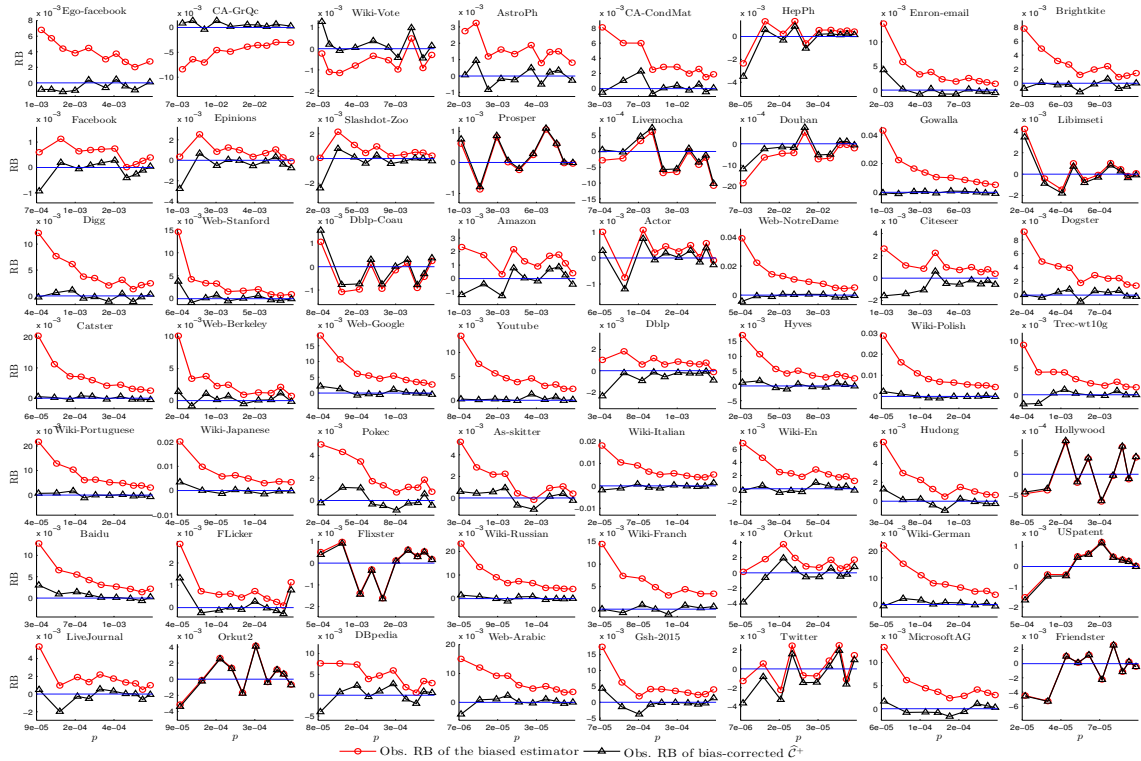


FIGURE 4.4: The bias-corrected estimator  $\hat{C}^+$  vs. biased estimator. The observed RBs were obtained over  $5 \times 10^4$  independent runs for all graphs except for the large graphs in the last row with  $10^4$  independent repetitions.

### 4.5.3 Positive and negative bias

We observed that there are both positive and negative biases, although most of the datasets demonstrate positive bias. We should note that by Jensen's inequality,

$\mathbb{E}(1/X) \geq 1/\mathbb{E}(X)$ , thus we may have the wrong impression that there is positive bias only. However, Jensen's inequality can not be applied to our result because we are looking at the bias of  $Y/X$ , not  $1/X$ . Therefore, in our case, we can have both positive and negative biases.

Next, let's check when negative bias can occur. According to Eq. 16, the negative bias occurs when  $\frac{2\Psi}{\Lambda^2} < \frac{\Omega}{\Lambda\Delta}$ . In other words, it happens when  $\Omega$  is large compared with other metrics. Empirically, it happens only for seven graphs among 56. These graphs are CA-GrQc, Wiki-vote, Prosper, Livemocha, Douban, DBLP-coau, Hollywood. All these graphs are (online) social networks with relatively high clustering coefficients.

Recall that terms  $\frac{2\Psi}{\Lambda^2}$  and  $\frac{\Omega}{\Lambda\Delta}$  are resulted from the first two terms of Taylor expansion. The relationship between these two terms decides not only the positive and negative bias, but also the size of the bias. When the two terms are close, the overall bias would be small. Hence, we use Fig. 4.5 Panel (B) to show the role of the first term. The figure plots observed bias against the first term. We can see that the first term can describe the RB for most data sets. However, there are some outliers. We plotted the labels for the first 12 graphs describe in Table 4.2. Recall that we sort the graphs by their (node) size. Overall, we can see that the smaller the graph is, the farther away it deviates from the linear fit. In other words, for large graphs, we can use the first term  $\frac{2\Psi}{\Lambda^2}$  to approximate the bias.

The next question is, if we focus on large graphs only, can we simplify  $\frac{2\Psi}{\Lambda^2}$  further? Estimation is needed only for very large graphs. Hence the assumption on large graphs is valid. The values of  $\Psi$  and  $\Lambda$  lack intuitive interpretation. Even though it is easy to estimate them from a sample graph, it would be helpful to give a more intuitive understanding of the values as described in the next subsection.

#### 4.5.4 Characterizing bias using second and third moments

When the graph is large, interestingly  $\Lambda$  and  $\Psi$  can be approximated by the second and third moments of the degrees of the graph. Recall that

$$\Lambda = \sum_{i=1}^N \Lambda_i = \sum_{i=1}^N \binom{d_i}{2}.$$

When the graph is large,  $d_i^2$  dominates, and the above can be simplified as

$$\Lambda \approx 0.5N \langle d^2 \rangle. \quad (52)$$

Similarly,

$$\begin{aligned} \Psi &= \sum_{(i,j) \in \mathcal{E}} \left[ \binom{d_i-1}{2} + \binom{d_j-1}{2} + (d_i-1)(d_j-1) \right] \\ &= \sum_{i=1}^N d_i \binom{d_i-1}{2} + \sum_{(i,j) \in \mathcal{E}} (d_i-1)(d_j-1) \approx 0.5N \langle d^3 \rangle. \end{aligned}$$

Therefore, we can approximate the bias by ignoring the second term:

$$RB \approx \frac{4 \langle d^3 \rangle}{pN \langle d^2 \rangle^2}, \quad (53)$$

where  $\langle d^2 \rangle = \sum_{i=1}^N d_i^2 / N$ , and  $\langle d^3 \rangle = \sum_{i=1}^N d_i^3 / N$ .

Although this result is not rigorous, we can demonstrate that  $\frac{2 \langle d^3 \rangle}{N \langle d^2 \rangle^2}$  can approximate  $\frac{\Psi}{\Lambda^2}$  well using Fig. 4.6. A visual inspection reveals that, among all 56 graphs including those small ones, all the data points are aligned well along the line. The Pearson correlation coefficient between those two is 0.99 for both logged and unlogged data points. We show the log-log plot only here. The unlogged version will have most data points cramped on the left lower corner due to the uneven distribution of those values.

This result tells us that the bias can be mostly determined by the third and second moments of degrees of the graph when the graph is large. We want to emphasize that when estimating clustering coefficient using  $\widehat{\mathcal{C}}^+$ , we only need to know the  $\Psi$  and  $\Omega$  in the sample graph. There is no need to calculate the second and third moments for

the entire graph. Eq. 53 is used only to understand the nature of the bias– the bias is large only when the third moment of the degree is large.

### 4.5.5 When the bias is large

Practitioners need to know what kind of graphs may have a large bias. Our results point out that when the bias is large,  $\Psi$  is relatively large while  $\Omega$  is relatively small. This happens for web graphs, such as a Stanford Web depicted in Fig. 4.7. The figure shows only a sampled graph obtained from a random walk. Yet it can reveal the overall structure of the graph: It contains a ball (s) that has a very high degree. Therefore,  $\Psi$  (or, equivalently,  $\langle d^3 \rangle$ ) is large. At the same, there are no triangles in the ball structure,  $\Omega$  will not increase for this node. This explains why the web graphs often have higher bias, as indicated in Fig. 4.3.

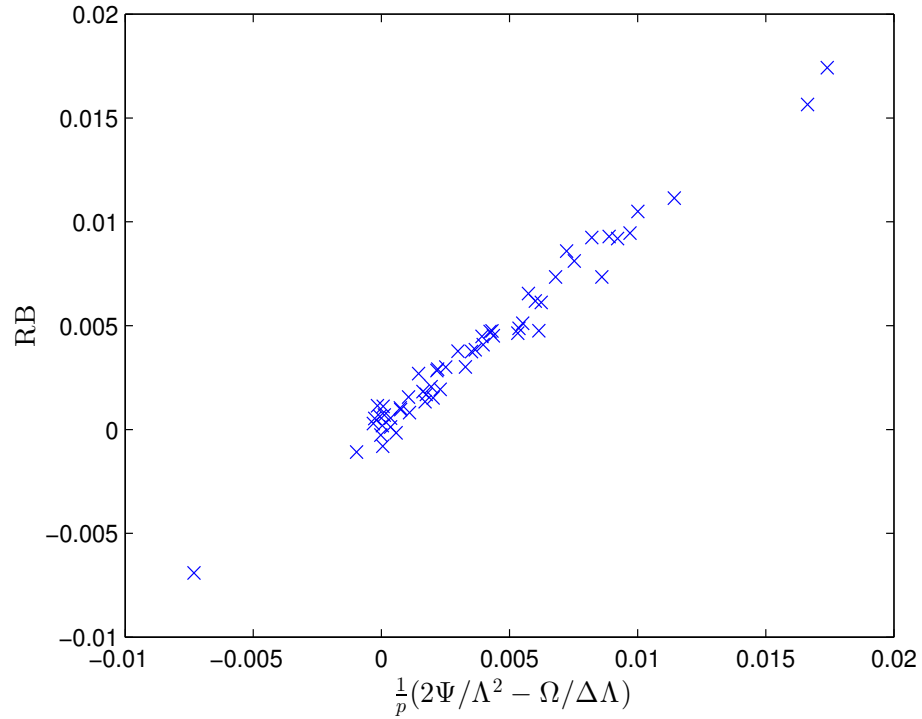
### 4.5.6 Verification of Theorem 2

To evaluate the RSE of the estimator, we tuned  $p$  to obtain the RSE between 0.1 and 0.4. The estimator repeated 100 times and the observed RSEs are reported in Fig. 4.8. The estimated RSEs in the plots are obtained using Eq. 32. Our estimation for the RSE based on Eq. 32 fits perfectly the observed RSE not only on the large graphs but also on the small ones, as shown in the plots.

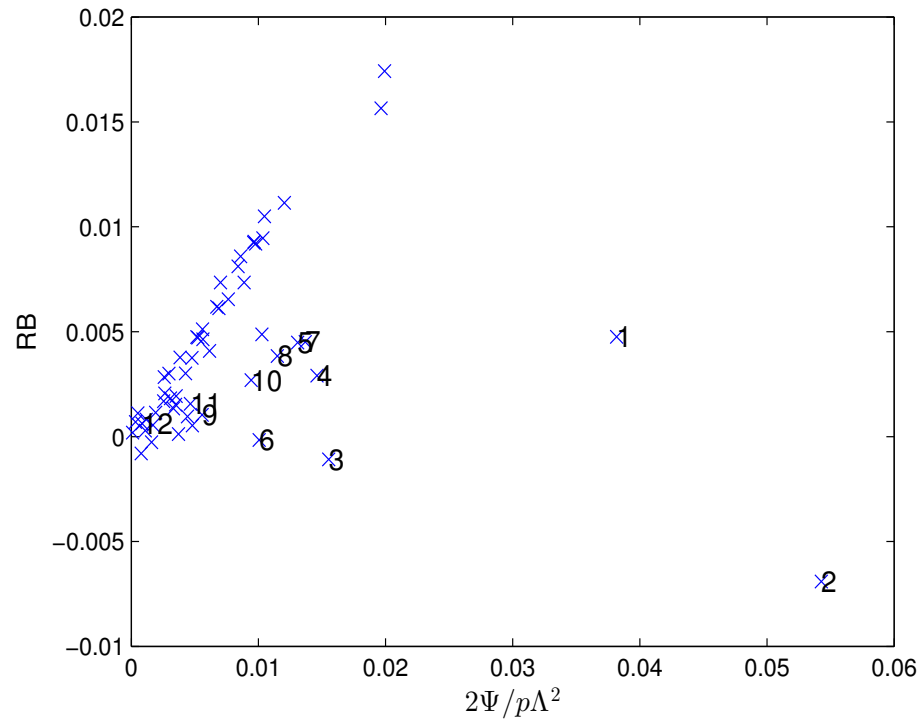
### 4.5.7 Verification of Theorem 3

To verify the approximations made in the derivation of the RSE of the estimator in Theorem 3, the parameters of the estimator were set up to obtain the RSEs between 0.1 and 0.4. First, we report the observed RSEs along with its estimations based on Eq. 39 in Fig. 4.9. Only 4 representative graphs are reported for lack of space. Similar pattern are observed for other graphs. We make several observations as follows.

- The estimated RSEs based on Eq. 39 fit perfectly the observed RSEs not only for large graphs when  $p$  is small but also for small ones. Take Ego-facbook, for



(A) RB vs. Two terms of Taylor expansion



(B) RB vs. First term of Taylor expansion

FIGURE 4.5: RB depends on the first term and second term of the Taylor expansion. The outliers are the 10 smallest graphs. Observed RBs are taken when RSE=0.2 over  $10^5$  independent runs.

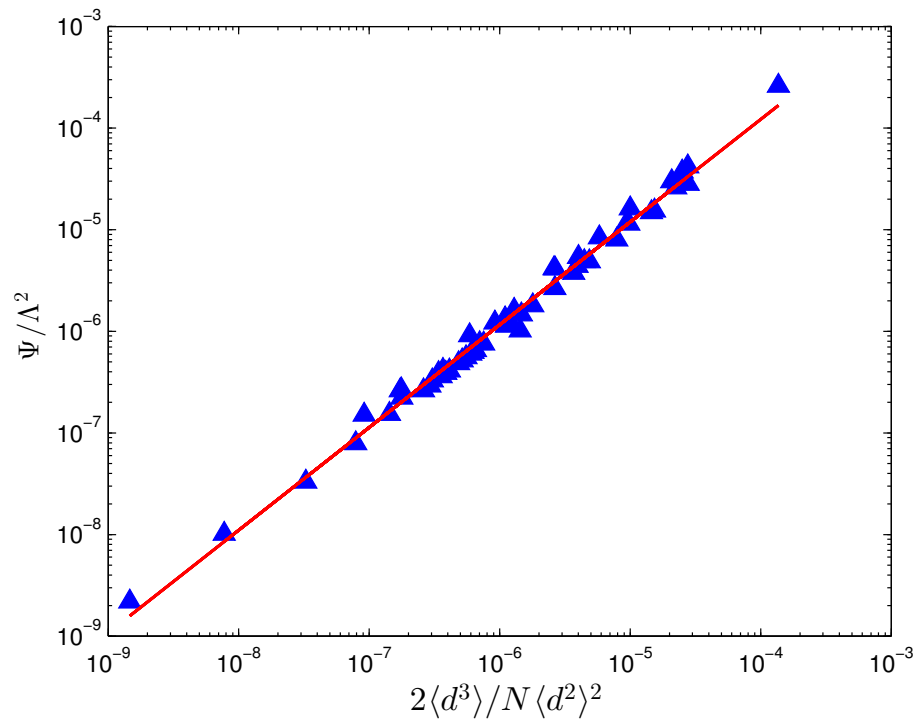


FIGURE 4.6:  $\frac{\Psi}{\Lambda^2}$  against  $\frac{2\langle d^3 \rangle}{N\langle d^2 \rangle^2}$  for 56 graphs.



FIGURE 4.7: A sample of the Stanford Web network.



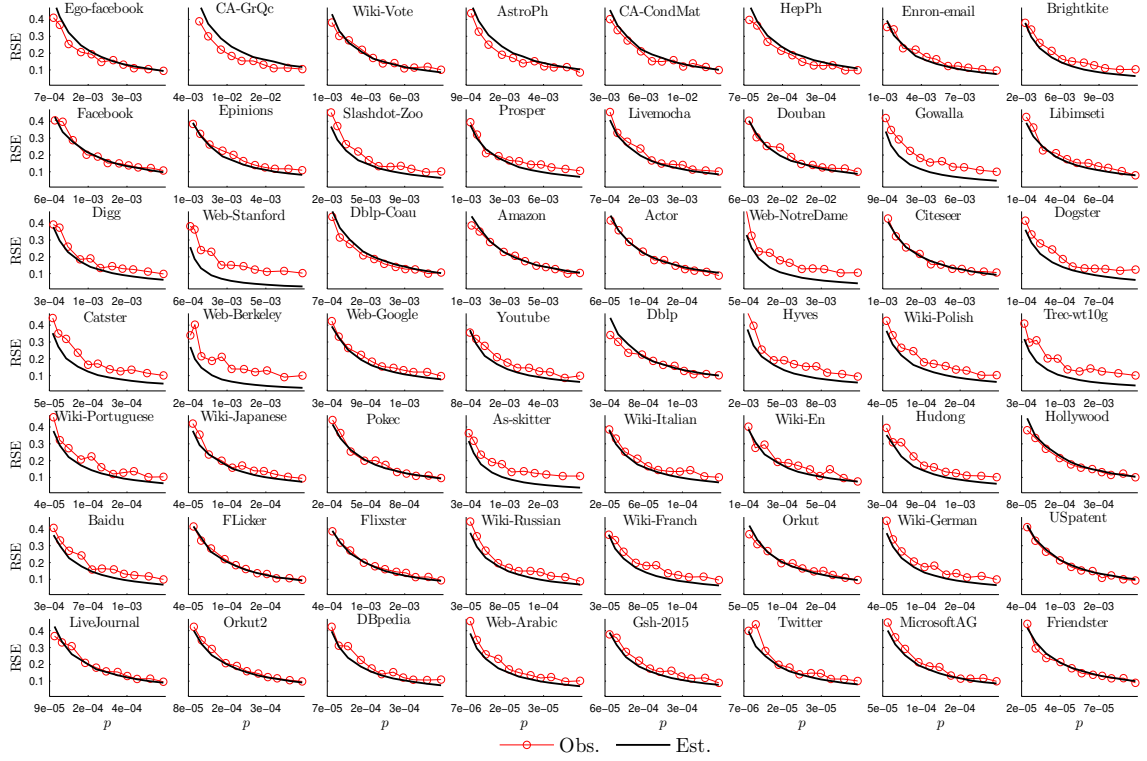


FIGURE 4.8: The observed vs. the estimated RSEs. Estimated RSEs are obtained based on Eq. 32 over 100 independent runs.

example, the smallest graph in our datasets, the observed RSEs (red circles) fit very well our estimations based on Eq. 39.

- When  $p$  is small the first term in Eq. 39, i.e.  $1/\Delta_g$ , is a dominant term compared to the remaining three. The reason is as follows. Firstly, when  $p$  is small, the probability of identifying shared triangles, i.e.  $\Phi_g$ , and pairs of triangle and wedge with one common edge ( $\Omega'_g$ ) based on the sampled  $g$  is very small. Note that  $\mathbb{E}(\Phi_g) = \frac{8}{15}\Phi p^3$ , and  $\mathbb{E}(\Omega'_g) = \frac{5}{12}\Omega' p^2$ . Moreover, the term  $2\Psi_g/\Lambda_g^2$  is neglectable due to the fact that their expectations depends on the sampling probability  $p$ . Therefore, in the sample  $\Psi_g \ll \Lambda_g^2$ .
- By increasing sampling probability  $p$ , terms  $+2\Phi_g/\Delta_g^2$  and  $-2\Omega'_g/\Delta_g\Lambda_g$  are increasing in the same order. Surprisingly that the two terms neutralize each other. Furthermore, term  $2\Psi_g/\Lambda_g^2$  remains neglectable compared to the other terms.

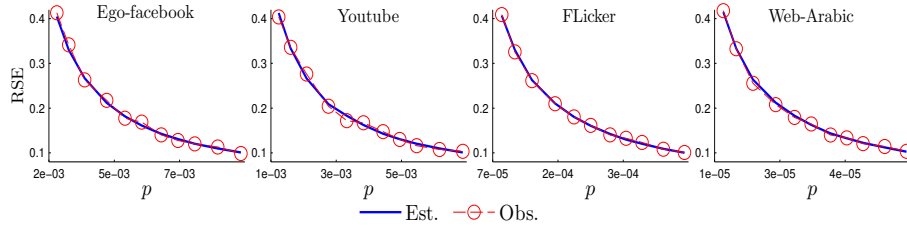


FIGURE 4.9: The observed RSEs of  $\hat{\mathcal{C}}$  fit perfectly the estimated RSEs based on Eq. 39 in representative graphs.

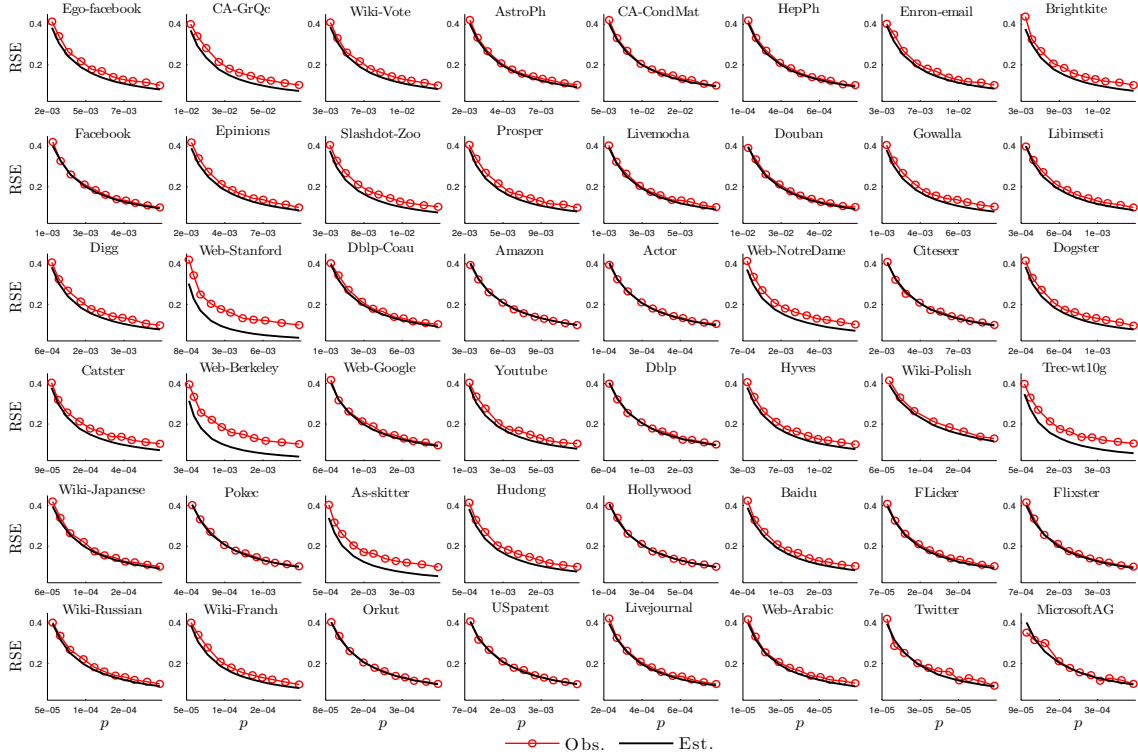


FIGURE 4.10: The estimated RSEs obtained based on Eq. 40 are apt estimations for the observed RSEs of  $\hat{\mathcal{C}}$ .

The observations above support our claim in Theorem 3 to simplify the RSE of the estimator. To verify the result in Theorem 3, we report the observed RSEs and their estimations based on Eq. 40 in Fig. 4.10. It can be seen that the observed RSEs support our estimations not only for large graphs but also for small ones. In a few small graphs, i.e. Web-Stanford, Web-Berkeley, and As-skitter, there are small gaps between the observed and estimated RSEs. However, by increasing the size of networks (see the last row in the figure) the estimated RSEs match perfectly the observed ones. Thus, we believe that our result in Theorem 3 can be used in practice

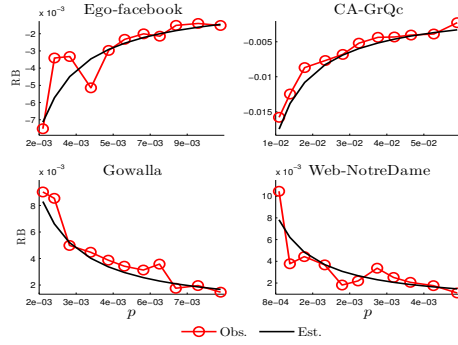


FIGURE 4.11: The observed RBs of  $\hat{\mathcal{C}}$  (biased estimator) support our estimations of RB based on Eq. 47.

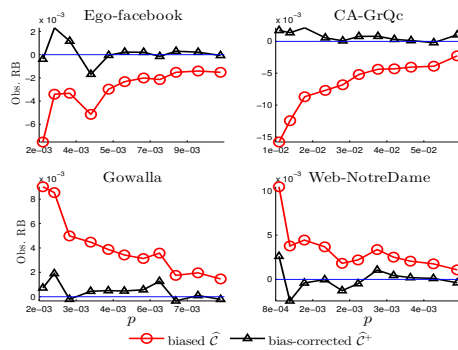


FIGURE 4.12: Our biased-corrected  $\hat{\mathcal{C}}^+$  removes the bias perfectly.

to determine the size of samples to achieve a given accuracy level of the estimation.

#### 4.5.8 The bias in the streaming model

To understand the bias, we set the parameters of the estimator to achieve the RSEs between 0.1 and 0.4. The estimator run on the graphs and the observed RBs (relative bias) and the estimated ones based on Eq. 47 were computed. The results were reported in Fig. 4.11 for the graphs with the RB more than 0.7%. We make several observations as follows: 1) The observed RBs support the estimated RBs based on Eq. 47 for all graphs; 2) Both negative and positive biases are observed; 3) The largest RBs were observed on small graphs and it can be as high as 2%;

We also report the observed RBs of our biased estimator and the bias-corrected one in Fig. 4.12. It can be seen that the bias of  $\hat{\mathcal{C}}^+$  was removed in all the plots.

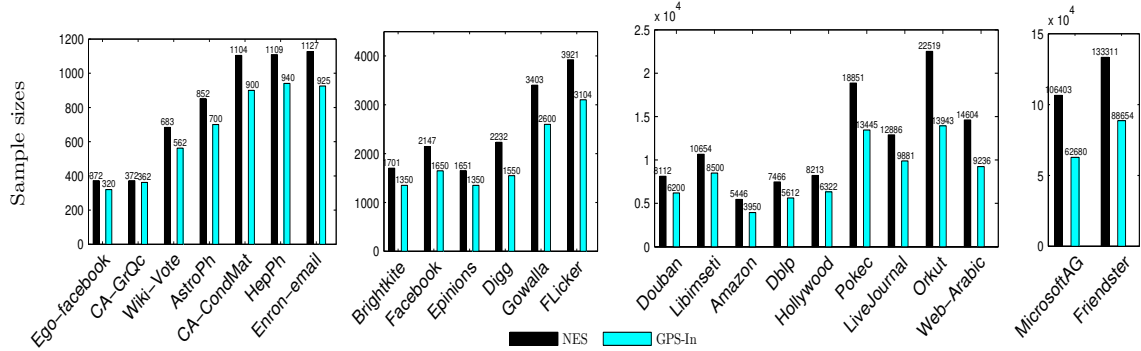


FIGURE 4.13: Comparison of sample sizes, i.e. number of sampled edges, of the methods when  $RSE=0.2$ . The sample size of NES is comparable with GPS-In [1].

## 4.6 Discussions and conclusions

We address the estimation of the bias and variance in both non-streaming and streaming algorithms for estimating clustering coefficient. Essentially it is about estimating the properties of an estimator. It is important since it is the only way to know how good an estimate is during the estimation process.

Our result is obtained based on two simplifications. One is the Taylor expansion—we take only the first two terms. The other is  $1 - p \approx 1 - p^2 \approx 1$ , assuming that sampling probability is very small if the data is very large. With such simplifications, we can characterize the variance in both non-streaming and streaming models with a single variable, i.e.,  $\Delta_g$ . Although in theory, variances depends on graph structures characterized by  $\Phi$ ,  $\Psi$ , and  $\Omega$ , all these variables can be neglected when estimations are performed on very large graph. This simple yet powerful result is very useful in practice – we can give a confidence interval when an estimate is given.

Bias is a perplexing problem in estimating graph properties in general [11] and clustering coefficient in particular [21]. It is difficult to observe because, for many graphs, especially small ones, the bias is almost negligible. Therefore, it has been taken for granted to use biased estimators by practitioners as well as researchers [23, 32]. It only became a more prominent problem recently, when people started to estimate very large graphs.

Bias for clustering coefficient estimation is difficult to quantify and correct [8, 29].

It involves two variables, i.e.,  $\Lambda_g$  and  $\Delta_g$ . We use Taylor expansion to approximate the bias, and show that quadratic expansion is good enough for the approximation. The quadratic expansion involves the variance and covariance of wedges and closed-wedges in the sample graph. We show that they can be quantified by  $\Psi$  and  $\Omega$  and estimated by  $\Psi_g$  and  $\Omega_g$ . Based on this result, we propose a bias-corrected estimator  $\widehat{C}^+$  in non-streaming model.

Bias for clustering coefficient estimation is difficult to understand. We observe positive and negative biases for different graphs. It can be as large as 4% for some graphs. On the other hand, it can be small even if the graph is very large. To understand negative bias, we find that the second term dominates the equation only when the network is relatively small, and they are (online) social networks. In other words, their clustering coefficient is high.

Our result is simple and elegant: first we quantify the bias as  $RB \approx \frac{1}{p} \left( \frac{2\Psi}{\Lambda^2} - \frac{\Omega}{\Lambda\Delta} \right)$ . This is much simpler than the original Taylor expansion because of our assumption that  $1 - p \approx 1$ . The assumption is valid when the graph is large. In most of the data sets we experimented with,  $p$  is typically in the order of  $10^{-4}$  to achieve reasonable accuracy. Furthermore, we demonstrate that RB can be simplified further by ignoring the second term when the graph is large, i.e.,  $RB \approx \frac{2\Psi}{p\Lambda^2}$ . Based on this, we can simplify the result further by approximate the bias using the second and third moments of the degrees of the graph, i.e.,  $RB \approx \frac{4\langle d^3 \rangle}{pN\langle d^2 \rangle^2}$ . This is instrumental in helping us identify the type of graphs that have high bias.

For the bias part in streaming model, we conclude that it is small overall, and it is more observable for smaller graphs. Interestingly, the bias can be either positive or negative, depending on the graph structure.

Although our result is developed on NES algorithm, the same method can be extended to numerous other streaming algorithms. Besides, NES itself is a very powerful algorithm. Despite its simplicity, its performance is comparable to the state-of-the-art algorithm GPS-In as illustrated in Fig.4.13. Hence, the variance estimator and NES are a good combination to be used in practice.

# Conclusions and Future Directions

---

---

## 5.1 Introduction

The estimation of metrics  $\Delta$  and  $\mathcal{C}$  in different sampling scenarios have been used as case studies to address the challenges of sampling methods to analyze real-world network graphs in the previous chapters. This chapter will summarize what has been accomplished in this dissertation. It will also contain the conclusions of the dissertation. Furthermore, it discusses future work in this direction.

## 5.2 Discussions and conclusions

We addressed the CI and bias problems of sampling methods to analyze massive real-life networks. Reviewing existing methods revealed that the CI of estimators were constructed using properties of original network data which are unknown in sampling scenarios [1, 7, 8, 10, 18–21, 23]. Furthermore, we found that efficient estimators for  $\mathcal{C}$  suffer from a bias problem [1, 8, 18, 21, 29]. Thus, motivated by those observations we used the estimation of  $\Delta$  and  $\mathcal{C}$  as case studies to address such problems. We believe that our techniques to construct the CI, and to quantify and correct the bias based on sampled data can be used in other sampling domains.

First, an edge sampling-based method was proposed to estimate  $\Delta$  in a non-streaming model. The variance of the estimator was derived as well. To have a better understanding of the performance of the estimator we simplified the variance using big data assumption. We also derived the estimator of the variance of the estimator. We also used the same treatment to get the estimator of the variance of the existing method. Our analytical results show that the variance of the estimators

can be estimated using the number of triangles in the sample. Furthermore, the performance ratio between the methods was quantified for the first time. To verify our assumptions in the analytical results, extensive experiments on 56 real-world graphs were conducted.

Second, the estimation of  $\Delta$  was addressed in a streaming model. The combination of uniform random edge sampling and reservoir sampling were used to estimate  $\Delta$  in such a model. Moreover, the variance of the estimator was derived. The same treatment in our first study was used to derive the estimator of the variance of the estimators. The variance estimators were utilized to construct the confidence interval of the estimators. Moreover, they were used to quantify the performance ratio of the methods. To verify our analytical results, extensive experiments on real-life networks from varieties of domains with different sizes were conducted.

Then, the estimation of  $\mathcal{C}$  in both streaming and non-streaming models was studied. Edge sampling-based methods have been used to estimate  $\mathcal{C}$ . However, we found that such estimators suffer from a bias problem which is noticed in several works but not quantified. Therefore, we quantified the bias in both streaming and non-streaming models using the Taylor extension method. Then, the biased-corrected estimators were proposed based on sampled data. Furthermore, the variances of the estimators were derived.

Last but not least, we proposed a new technique to obtain analytical comparisons of the methods in this context. By investigating the literature in this direction, we found that the sampling methods were compared using experimental results. The main disadvantage of such an experimental comparison is that the results depend on the network data and vary from data to data. Thus, we were motivated to quantify the performance ratio between the methods. To do so, we used big data assumption to derive the estimator of the variance for the estimators. Our estimator of the variance of the estimators has two main implications. First, they are useful to derive the analytical comparisons of sampling-based methods. Second, they can be used to construct the confidence interval of the estimators using the properties of sampled data.

### 5.3 Impact of our dissertation research

This dissertation proposed efficient methods to estimate the number of triangles and clustering coefficient in network graphs. Counting the number of triangles and computing clustering coefficient are fundamental problems in graph theory with many applications. Metrics  $\Delta$  and  $\mathcal{C}$  are also most useful properties of real-life network graphs. For example,  $\Delta$  and  $\mathcal{C}$  can help practitioners to determine whether a graph looks like a social network or not. A social-network graph is expected to have  $\Delta$  much greater than the value for random graphs [38]. Moreover, they can be used to study the properties of communities in a social network. It is directly correlated with the age of a community. Thus, our estimators to  $\Delta$  and  $\mathcal{C}$  can be used to develop scalable tools to analyze massive real-world networks.

Metrics  $\Delta$  and  $\mathcal{C}$  have been widely used for data mining tasks such as link prediction. One example of such a task is friend suggestion in online social networks such as Facebook, Orkut, and LinkedIn [35, 75, 76]. Another example is protein-protein interactions prediction for drug design in bioinformatics [42, 54]. Obviously, the exact values of measures  $\Delta$  and  $\mathcal{C}$  are not required in those tasks. Thus, our methods to estimate  $\Delta$  and  $\mathcal{C}$  can be used in such applications.

Other applications of our methods are graph clustering in data mining, spam detection in computer networks, wireless and ad hoc networks analysis in computer networks, blog analysis and finding interesting individuals in online social networks, DAN sequence analysis, prediction of essential proteins, microarray data analysis for finding cancer genes, and identifying modular formations in protein-protein interaction network in biology, risk analysis in economy, word-learning in education, and many more.

### 5.4 Looking into the future

Our approaches in this dissertation can be extended in the following directions:

- **Counting  $\Delta$  in directed networks:** This dissertation studied the estimation



of  $\Delta$  and  $\mathcal{C}$  in simple undirected real-world networks in depth. Estimation of structural metrics of directed network graphs can be a future study in this direction.

- **Counting motifs and cliques:** A triangle is a complete graph with size three ( $K_3$ ) or a 3-clique. Our methods to estimate  $\Delta$  can be easily applied to count larger useful structures such as  $k$ -clique ( $k=4,5,..$ ), or other pattern subgraphs.
- **Estimate other structural metrics:** The average shortest path length is an important metric to understand the information flow in a network. However, some research has been conducted to estimate such a metric on large networks. Sampling methods can be used to design efficient estimators in this context as well.
- **Graph clustering and community detection:** Metric  $\Delta$  has been used to improve the results of graph clustering and community detection tasks [33, 77, 78]. Metrics  $\Delta$  and  $\mathcal{C}$  can be used as heuristics to design sampling methods to preserve community structure of massive networks in a sampled data.

## REFERENCES

- [1] Nesreen K Ahmed, Nick Duffield, Theodore L Willke, and Ryan A Rossi. On sampling from massive graph streams. *Proceedings of the VLDB Endowment*, 10(11):1430–1441, 2017.
- [2] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.
- [3] Matthieu Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical Computer Science*, 407(1):458–473, 2008.
- [4] Liran Katzir and Stephen J Hardiman. Estimating clustering coefficients and size of social networks via random walk. *ACM Transactions on the Web (TWEB)*, 9(4):19, 2015.
- [5] Jianguo Lu and Dingding Li. Sampling online social networks by random walk. In *ACM SIGKDD Workshop on Hot Topics in Online Social Networks*, pages 33–40. ACM, 2012.
- [6] Thomas Schank and Dorothea Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005.
- [7] Charalampos E Tsourakakis, U Kang, Gary L Miller, and Christos Faloutsos. Doulion: counting triangles in massive graphs with a coin. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 837-846. ACM, 2009.
- [8] Nesreen K Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. Graph sample and hold: A framework for big-graph analytics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1446–1455. ACM, 2014.
- [9] Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, 623-632. Society for Industrial and Applied Mathematics, 2002.
- [10] Luciana S Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 253-262. ACM, 2006.

- [11] Jianguo Lu and Dingding Li. Bias correction in a small sample from big data. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2658–2663, 2013.
- [12] Jianguo Lu and Hao Wang. Uniform random sampling not recommended for size estimation for large graph. 2014.
- [13] Tianyi Wang, Yang Chen, Zengbin Zhang, Tianyin Xu, Long Jin, Pan Hui, Beixing Deng, and Xing Li. Understanding graph sampling algorithms for social network analysis. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 123–128. IEEE, 2011.
- [14] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.
- [15] Pafnutii Lvovich Chebyshev. Des valeurs moyennes, liouville’s. *J. Math. Pures Appl.*, 12:177–184, 1867.
- [16] C Seshadhri, Ali Pinar, and Tamara G Kolda. Fast triangle counting through wedge sampling. In *Proceedings of the SIAM Conference on Data Mining*, volume 4, page 5. Citeseer, 2013.
- [17] Hossein Jowhari and Mohammad Ghodsi. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics*, 710-716. Springer, 2005.
- [18] Aduri Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Counting and sampling triangles from a graph stream. *Proceedings of the VLDB Endowment*, 6(14):1870–1881, 2013.
- [19] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. *triest*: Counting local and global triangles in fully-dynamic streams with fixed memory size. In *Proceedings of the 22th ACM KDD international conference on Knowledge discovery and data mining*. ACM, 2016.
- [20] Yongsub Lim and U Kang. Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 685–694. ACM, 2015.
- [21] Madhav Jha, C. Seshadhri, and Ali Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’13, pages 589–597, New York, USA, 2013. ACM.
- [22] Liran Katzir, Edo Liberty, and Oren Somekh. Estimating sizes of social networks via biased sampling. In *Proceedings of the 20th international conference on World wide web*, pages 597–606. ACM, 2011.

- [23] Stephen J Hardiman and Liran Katzir. Estimating clustering coefficients and size of social networks via random walk. In *Proceedings of the 22nd international conference on World Wide Web*, pages 539–550. ACM, 2013.
- [24] Jianguo Lu and Hao Wang. Uniform random sampling not recommended for large graph size estimation. *Information Sciences*, 421(1):136–153, 2017.
- [25] Guoyong Mao and Ning Zhang. Analysis of average shortest-path length of scale-free network. *Journal of Applied Mathematics*, 2013, 2013.
- [26] Guoyong Mao and Ning Zhang. Fast approximation of average shortest path length of directed ba networks. *Physica A: Statistical Mechanics and its Applications*, 466:243–248, 2017.
- [27] Mahmudur Rahman and Mohammad Al Hasan. Sampling triples from restricted networks using mcmc strategy. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM '14*, pages 1519–1528, New York, NY, USA, 2014. ACM.
- [28] Roohollah Etemadi, Jianguo Lu, and Yung H. Tsin. Efficient estimation of triangles in very large graphs. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16*, pages 1251–1260. ACM, 2016.
- [29] Madhav Jha, C Seshadhri, and Ali Pinar. A space-efficient streaming algorithm for estimating transitivity and triangle counts using the birthday paradox. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(3):15, 2015.
- [30] Roohollah Etemadi and Jianguo Lu. Bias correction in clustering coefficient estimation. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 606–615, Dec 2017.
- [31] Comandur Seshadhri, Ali Pinar, and Tamara G Kolda. Triadic measures on graphs: The power of wedge sampling. In *SIAM International Conference on Data Mining (SDM)*, pages 10–18. SIAM, 2013.
- [32] Liran Katzir and Stephen J Hardiman. Estimating clustering coefficients and size of social networks via random walk. *ACM Transactions on the Web (TWEB)*, 9(4):19, 2015.
- [33] Mariá CV Nascimento. Community detection in networks via a spectral heuristic based on the clustering coefficient. *Discrete Applied Mathematics*, 176:89–99, 2014.
- [34] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564. ACM, 2017.

- [35] Yangyang Liu, Chengli Zhao, Xiaojie Wang, Qiangjuan Huang, Xue Zhang, and Dongyun Yi. The degree-related clustering coefficient and its application to link prediction. *Physica A: Statistical Mechanics and Its Applications*, 454:24–33, 2016.
- [36] Ho-Yu Lam and Dit-Yan Yeung. A learning approach to spam detection based on social networks. In *4th Conference on Email and Anti-Spam (CEAS)*, 2007.
- [37] Michael Brautbar and Michael J Kearns. Local algorithms for finding interesting individuals in large networks. 2010.
- [38] David Easley, Jon Kleinberg, et al. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010.
- [39] Matthias R Brust, Damla Turgut, Carlos HC Ribeiro, and Marcus Kaiser. Is the clustering coefficient a measure for fault tolerance in wireless sensor networks? In *2012 IEEE International Conference on Communications (ICC)*, pages 183–187. IEEE, 2012.
- [40] Günther JL Gerhardt, Ney Lemke, and Gilberto Corso. Network clustering coefficient approach to DNA sequence analysis. *Chaos, Solitons & Fractals*, 28(4):1037–1045, 2006.
- [41] Jianxin Wang, Min Li, Huan Wang, and Yi Pan. Identification of essential proteins based on edge clustering coefficient. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):1070–1080, 2012.
- [42] Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell’s functional organization. *Nature reviews genetics*, 5(2):101, 2004.
- [43] Gabriela Kalna and Desmond J Higham. A clustering coefficient for weighted networks, with application to gene expression data. *Ai Communications*, 20(4):263–271, 2007.
- [44] Zelmina Lubovac, Björn Olsson, and Jonas Gamalielsson. Weighted clustering coefficient for identifying modular formations in protein-protein interaction networks. In *Proc. 3: rd International Conference on Bioinformatics and Computational and Systems Biology, Prague, Czech republic*, 2006.
- [45] Benjamin M Tabak, Marcelo Takami, Jadson MC Rocha, Daniel O Cajueiro, and Sergio RS Souza. Directed clustering coefficient as a measure of systemic risk in complex banking networks. *Physica A: Statistical Mechanics and its Applications*, 394:211–216, 2014.
- [46] Rutherford Goldstein and Michael S Vitevitch. The influence of clustering coefficient on word-learning: how groups of similar sounding words facilitate acquisition. *Frontiers in psychology*, 5:1307, 2014.

- [47] Duru Türkoglu and Ata Turk. Edge-based wedge sampling to estimate triangle counts in very large graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 455–464. IEEE, 2017.
- [48] Roohollah Etemadi and Jianguo Lu. Pes: Priority edge sampling in streaming triangle estimation. *IEEE Transactions on Big Data (Under revision)*, 2019.
- [49] Roohollah Etemadi and Jianguo Lu. Pes: Priority edge sampling in streaming triangle estimation. *arXiv preprint arXiv:1812.01200*, 2018.
- [50] Roohollah Etemadi and Jianguo Lu. Bias correction in streaming and non-streaming algorithms in clustering coefficient estimation. *IEEE Transactions on Knowledge and Data Engineering (Under review)*, 2019.
- [51] Roohollah Etemadi and Jianguo Lu. The estimation of bias and variance in clustering coefficient streaming algorithms. *arXiv preprint arXiv:1811.01109*, 2018.
- [52] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.
- [53] Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572, 2002.
- [54] Jianxin Wang, Min Li, Huan Wang, and Yi Pan. Identification of essential proteins based on edge clustering coefficient. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 9(4):1070–1080, 2012.
- [55] Tim Kastelle, John Steen, and Peter Liesch. Measuring globalisation: an evolutionary economic approach to tracking the evolution of international trade. In *DRUID Summer Conference on Knowledge, Innovation and Competitiveness: Dynamics of Firms, Networks, Regions and Institutions-Copenhagen, Denmark, June, 18-20.*, 2006.
- [56] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.
- [57] Haim Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *in: Proceedings of KDD-LDMTA’10, 2010*.
- [58] Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.
- [59] Mohammad Al Hasan and Vachik S Dave. Triangle counting in large networks: a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(2):e1226, 2018.

- [60] Charalampos E Tsourakakis, Mihail N Kolountzakis, and Gary L Miller. Triangle sparsifiers. *J. Graph Algorithms Appl.*, 15(6):703–726, 2011.
- [61] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
- [62] J. Kunegis. Konect - the koblenz network collection. <http://konect.uni-koblenz.de/networks>, May 2016.
- [63] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *Proceedings of the 20th international conference on World Wide Web*, 587-596. ACM Press, 2011.
- [64] Paolo Boldi and Sebastiano Vigna. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference*, 595-601, Manhattan, USA, 2004. ACM Press.
- [65] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, pages 243–246. ACM, 2015.
- [66] Jianguo Lu and Hao Wang. Variance reduction in large graph sampling. *Information Processing & Management*, 50(3):476–491, 2014.
- [67] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–24. ACM, 2008.
- [68] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.
- [69] P Oscar Boykin and Vwani P Roychowdhury. Leveraging social networks to fight spam. *Computer*, 38(4):61–68, 2005.
- [70] S. Wang, J. Lu, and J. Wang. Approximate common structures in xml schema matching. *Advances in Web-Age Information Management*, pages 900–905, 2005.
- [71] Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '16, pages 401–411, New York, NY, USA, 2016. ACM.
- [72] Jianguo Lu and Hao Wang. Variance reduction in large graph sampling. *Information Processing and Management*, 50(3):476–491, 2014.

- [73] Xiaowei Chen, Yongkun Li, Pinghui Wang, and John Lui. A general framework for estimating graphlet statistics via random walk. In *Proceedings of the VLDB Endowment*, pages 253–264, 2016.
- [74] Richard P Stanley. Differentiably finite power series. *European journal of combinatorics*, 1(2):175–188, 1980.
- [75] Zan Huang. Link prediction based on graph topology: The predictive value of generalized clustering coefficient. 2010.
- [76] Zhihao Wu, Youfang Lin, Jing Wang, and Steve Gregory. Link prediction with node clustering coefficient. *Physica A: Statistical Mechanics and its Applications*, 452:1–8, 2016.
- [77] Arun S Maiya and Tanya Y Berger-Wolf. Sampling community structure. In *Proceedings of the 19th international conference on World wide web*, pages 701–710. ACM, 2010.
- [78] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564. ACM, 2017.



## VITA AUCTORIS

Roohollah Etemadi was born in Iran in 1981. He obtained his B.Sc. and M.Sc. in Computer Engineering-Software from Azad University in 2005 and 2008 respectively. He was a full-time faculty member at Azad University for eight years before he started his Ph.D. in Canada in 2015. His recent research interests are big data analytics, estimation, machine learning, neural network, deep learning, and graph/text mining. His research results have been published in top-tier conferences– CIKM and IEEE Big Data. He has also received several scholarships and awards including Ontario Graduate Scholarship (OGS), SIGIR Student Travel Grant, and IEEE Big Data Student Travel Award based on his academic and research excellence.