EFFICIENT AND SCALABLE SUBGRAPH STATISTICS USING REGENERATIVE MARKOV CHAIN MONTE CARLO

by

Mayank Kakodkar

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Computer Science

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Bruno Ribeiro, Chair

Department of Computer Science

Dr. Petros Drineas

Department of Computer Science

Dr. Vinayak A.P. Rao

Department of Statistics

Department of Computer Science

Dr. Kent R. Quanrud

Department of Computer Science

Approved by:

Dr. Kihong Park

To Dr. Siya Kunde.

ACKNOWLEDGMENTS

I would like to begin by thanking my advisor Prof. Bruno Ribeiro for introducing me to the beautiful world of Markov chains and sampling algorithms. Oftentimes, when I faced setbacks in my research, his inspiring words and course correction were indispensable. I will always strive to emulate his academic rigor and love for science, and will cherish the multiple academic and non-academic discussions I had with him. I am especially grateful to Bruno for his constant support during the COVID-19 pandemic and his genuine effort to connect with me and my lab-mates outside of work.

In addition to their valuable feedback on my research, I would like to thank my committee for playing a foundational role in my training. I thank Prof. Vinayak Rao for introducing me to Bayesian Inference and various sampling algorithms, and Prof. Kent Quanrud and Prof. Petros Drineas for their stellar course on randomized algorithms. On multiple occasions my research was guided by my interactions with them, and specifically, their comments during my candidacy shaped Chapter 2.

I am grateful to Janice Zdankus (HPE), Phil Gonsalves (YWCA), and the rest of the Curated Pathways to Innovation™ initiative for funding my research while giving me the opportunity to directly affect better representation in STEM. Working with the Department of Computer Science personnel was always a pleasure, and the support extended by science-IT was unparalleled. As such I am thankful to the department for their amazing staff and the environment they foster.

I am fortunate to have very supportive lab-mates – Leo, Chandra, Cotta, Bala, Jason, Ryan, Jianfei, Beatrice, Yangze, Shishang, Josue – and am thankful for the multiple work-hours they spent brainstorming ideas with me and for providing valuable feedback on my research and talks. I will always treasure our camaraderie and conversations about topics ranging from personal matters to work, that were disguised as coffee chats and weekend hangouts. I thank my external collaborators, Carlos, Pedro, Vinicius and Prof. Meira without whom my dissertation would not exist.

I am eternally indebted to my family and friends for being my most ardent supporters throughout my six years at Purdue. I am thankful to my Bethel drive family – Rohit, Viplove,

Dr. Patil, Ashutosh, Akash, Abhijit, Karthik – for being the best roommates possible. I am thankful to my mother for always for making me believe in myself as a kid, for pushing me to succeed, and for the countless sacrifices she made which allowed me the privilege of attending grad school. I am grateful to Rahul, whose idea of sibling rivalry is competing to be more supportive of me than I am of him, to my parents-in-law for their constant belief in me and to the rest of my family.

Most importantly, I would like to thank Siya, my one true love and the wind beneath my wings. She emboldened me to start grad school when I lacked the confidence to take the plunge, was my most constant source of encouragement and was always available when I needed her. I appreciate her being there for me despite her own grueling schedule and for always keeping me centered during the toughest of times. She has been the most wonderful companion throughout this journey: she was understanding during paper deadlines, she shared in my excitement at each milestone, cursed "reviewer-2" with me during each rebuttal, and was a 100% invested in me throughout. I consider myself lucky to have her by my side and for always inspiring me to be a better person.

TABLE OF CONTENTS

LIST OF TABLES 10

LIST OF FIGURES 11

ABSTRACT 13

1 INTRODUCTION 15

1.1 Contributions 17

1.2 Outline and Previously Published Work 18

2 REGENERATIVE TREE SAMPLING 20

2.1 Introduction 20

2.2 Preliminaries 22

2.2.1 Task 22

2.2.2 *FANMOD* tree and its original algorithm 23

2.3 Regenerative Tree Sampling 26

2.3.1 Motivation: Knuth-sampling the *FANMOD* tree 27

2.3.2 Fast and parallel uniform leaf sampling 28

2.3.3 *RTS* on the *FANMOD* tree 29

2.3.4 Estimators from $\mathbf{\Phi}^{(k)}$ regenerations 30

2.3.5 *RTS* in practice: Setting edge weights 32

2.3.6 Memory complexity 33

2.4 Scaling *PSRW* to Large k 34

2.5 Related Work 35

2.5.1 Random walk-based *CIS* Sampling. 35

2.5.2 Other *CIS*-sampling methods. 36

2.5.3 Regenerations in Random Walks. 36

2.5.4 Tree Sampling. 37

2.6 Empirical Evaluation 37

2.6.1 CIS pattern distribution estimation task 38

2.6.2 Average *CIS* edge-density estimation task . . . 40
2.7 Summary . . . 41

3 SEQUENTIAL STRATIFIED REGENERATIONS . . . 42
3.1 Introduction . . . 42
3.2 Background and Prior Work . . . 43
3.2.1 Regenerations in Discrete Markov Chains . . . 44
3.2.2 Improving the Regeneration Frequency . . . 45
3.3 Sequential Stratified Regenerations . . . 46
3.3.1 Sequential Stratification . . . 46
3.3.2 Recursive Regenerations . . . 49
3.4 Applying *Ripple* to Count Subgraphs . . . 51
3.4.1 *MCMC* on the Subgraph Space . . . 52
3.4.2 Ergodicity-Preserving Stratification for Subgraph Counting . . . 53
3.4.3 Miscellaneous Optimizations . . . 54
3.5 Experiments and Results . . . 55
3.5.1 Scalability Assessment . . . 57
3.5.2 Accuracy and Convergence Assessment . . . 60
Accuracy on Small k. . . . 61
Convergence for Large k. . . . 61
3.6 Related Work . . . 61
3.6.1 Parallel *MCMC* through Splitting. . . . 63
3.6.2 Subgraph Counting through Sampling. . . . 63
3.7 Summary . . . 64

4 IMPROVING RBM TRAINING THROUGH STOPPING SETS . . . 65
4.1 Introduction . . . 65
4.2 MCLV-K Estimation with Statistical Guarantees . . . 67
4.2.1 MCLV-K Estimator . . . 71
4.2.2 MCLV-K Finite-Sample Unbiasedness . . . 73
4.3 Training Restricted Boltzmann Machines . . . 74

4.3.1 MCLV-K Gradient Estimates . . . 74
4.3.2 Correspondence and Differences Between LVS-$\boldsymbol{K}$ and CD-$\boldsymbol{K}$ . . . 75
4.3.3 Computational Complexity . . . 76
4.4 Related Work . . . 77
4.5 Empirical Results . . . 78
4.6 Summary . . . 83

5 CONCLUSION . . . 84

REFERENCES . . . 85

A DETAILS FOR CHAPTER 2 . . . 94
A.1 Additional Results . . . 94
A.2 Proofs preliminaries . . . 94
A.3 Proofs for *RTS* . . . 96
A.4 Proofs for *R-PSRW* . . . 99

B DETAILS FOR CHAPTER 3 . . . 101
B.1 Notation . . . 101
B.2 Proofs for Section 3.2 . . . 101
B.2.1 *MCMC* Estimates . . . 101
B.2.2 Proof of Lemma 3 . . . 104
B.3 Proofs for Section 3.3 . . . 106
B.3.1 Proof of Proposition 3.3.1 . . . 107
B.3.2 Proof of Theorem 3.3.1 . . . 107
B.3.3 Proof of Theorem 3.3.2 . . . 109
B.4 Proofs for Section 3.4 . . . 111
B.4.1 Proof of Proposition 3.4.1 . . . 111
B.4.2 Proof of Proposition 3.4.2 . . . 112
B.5 Additional Implementation Details . . . 113
B.5.1 Parallel Sampling with a Reservoir Matrix. . . . 113
B.5.2 *PSRW* Neighborhood . . . 114

Articulation Points 114
B.5.3 Proof of Proposition 3.4.3 114
B.6 Additional Results 117

C DETAILS FOR CHAPTER 4 121
C.1 Proof of Corollary 1 121
C.2 Proof of Lemma 4 121
C.3 Proof of Theorems 4.2.1 and 4.2.3 122

LIST OF TABLES

3.1 The graphs that I used along with their diameter D_G, maximum degree Δ_G and the estimated orders of magnitude of k-*CIS* counts, $|\mathcal{V}^{(k)}|$. 56

3.2 Running time comparison between *Ripple* and *Motivo*. The last column shows that for large k, *Ripple* provides gains of up to 9 hours when Motivo can run to completion. *Motivo* crashes for large k on large graphs. 58

3.3 Space usage comparison between *Ripple* and *Motivo*.*Motivo* runs out of *disk* space for larger datasets in which $k \geq 10$, while *Ripple* scales almost linearly. *Ripple* saves up to 600 GB of space when *Motivo* can run. 59

4.1 (Higher is better) Average log-likelihood on the `MNIST` dataset using a RBM with 32 hidden neurons. Results are means over 10 executions after 100 epochs. . . . 80

4.2 (Higher is better) Average log-likelihood on the `MNIST` dataset using a RBM with 25 hidden neurons. Results are means over 10 executions after 100 epochs, using appropriate learning rates for each method. 80

A.1 Input graphs that were used for evaluating my method *RTS* against state of the art baselines. Each dataset is available from SNAP [75] via the identifier in the parentheses. I also report the number of vertices, edges and maximum degree for each graph (largest connected component only). 94

B.1 Table of Notations 102

B.2 Dispersion, $\frac{\max - \min}{\text{mean}}$, of *Ripple*'s estimates of $|\mathcal{V}^{(k)}|$ computed using $\epsilon = 0.003$, $|\mathcal{I}_1| = 10^4$ and $\text{M} = 10^7$ for $k = 6, 8, 10, 12$ used in the analysis in Section 3.5.1. The selected hyper-parameters provide reasonably similar estimates over 10 independent runs. Orkut for $k = 8$ exhibits the largest dispersion because of the presence of a single outlier. 118

LIST OF FIGURES

2.1 *RTS* and *R-PSRW* accuracy in the *CIS* edge density estimation task from Section 2.6.2. The average edge density of all *CIS*s is estimated using each method over 10 runs with wall-clock limits of 1 hour and 3½ hours. Figures 2.1a to 2.1d show the convergence properties of the *CIS* edge density estimates for $k \in \{8, 16, 20, 25\}$ and all methods and real-world graphs. The x-axis represents the NRMSE of the 1 hour estimates computed using the median of 3½ hour estimates of the same method as the ground-truth. The y-axis represents the reduction in variance between the 1 hour and 3½ hour estimates. Marker shapes represent various methods, colors represent graphs, and larger markers represent larger graphs. For each color, methods corresponding to markers closer to the bottom-right corner have better convergence properties. *RTS* and *R-PSRW* estimates have better convergence properties compared to the other tested baseline methods, with *RTS* estimates showing superior convergence more often. In Figures 2.1e and 2.1f I show that the estimates of my proposed *RTS* and *R-PSRW* for $k \in \{16, 25\}$ agree, which I take (in the absence of ground-truth) as strong evidence that these estimates are accurate. 21

2.2 The *FANMOD* tree, $\mathcal{T}^{(3)}$, for sampling 3-*CIS*s in G_{IN}, rooted at ρ. Each tree-node contains the (VSET, EXT, MAIN) properties from Definition 2.2.3. Note how each 3-*CIS* member of $\mathcal{S}^{(3)}$ is *uniquely* represented in the *FANMOD* tree. 25

2.3 *RTS* Accuracy in the 5-*CIS* pattern distribution estimation task (Section 2.6.1) in various real-world graphs. Each box-plot represents quartiles of the L2 error (10 runs) relative to the true pattern distribution computed by *Escape* [14]. Estimates are computed after running each method for 3½ hours (*Motivo* could not finish the indexing of `LiveJournal` and `Orkut` by the deadline). *RTS* estimates are consistently as accurate as or better than the other methods. *R-PSRW* matches *RTS* in some graphs, and is optimal for `DBLP`. 39

3.1 Figure 3.1a shows a simple graph $\mathcal{G}$ that is stratified into four strata $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4\}$. Figures 3.1b to 3.1d show the second, third and fourth graph strata constructed by Definition 3.3.2. In the multi-graph $\mathcal{G}_2$ (Figure 3.1b), vertices in $\mathcal{I}_1$ are collapsed into ζ_2 and only edges incident on $\mathcal{I}_2$ are preserved. The edge set therefore contains $\mathcal{J}_2$ and the edges between ζ_2 and $\mathcal{I}_2$. Consequently, self-loops on ζ_2 and edges between $\mathcal{I}_{3:4}$ are absent. Figures 3.1c and 3.1d follow suit. In each stratum $\mathcal{G}_r$, *RWT*s from ζ_r are started by sampling u.a.r. from the dotted edges and estimates are computed over the solid edges. 47

3.2 Accuracy and convergence analysis for 5-*CIS* s. I plot the L2-norm between the *Ripple* estimate and the exact value of the count vector $\mathcal{C}^{(5)}$ (Equation (3.9)) of all non-isomorphic subgraph patterns against various configurations of the parameters ϵ and $|\mathcal{I}_1|$. As expected, the accuracy improves as the error bound ϵ decreases and the number of seed subgraphs $|\mathcal{I}_1|$ increases. Each box and whisker represents 10 runs. 60

3.3 Convergence of *Ripple* estimates of 12-*CIS* pattern counts. I estimate the total number of subgraphs $|\mathcal{V}^{(12)}|$ and the number of sparse patterns and stars. Estimates over 10 runs are presented as box and whiskers plots, which exhibit a reduction in variance as ϵ increases. Indeed, almost all patterns are sparse, and the most frequent substructure is a star. 62

4.1 CD-k bias towards longer tours for $k \geq 2$; 76

4.2 (a) Tour lengths CCDF for $n_H = \log_2 |H| \in \{16, 20, 32, 64\}$ for LVS-1; (b) Tour lengths CCDF variation for LVS-10 with $n_H = 32$, using larger Stopping Sets; (c) Comparison of frequencies of short and long tours starting from labeled states on a trained RBM 81

4.3 Visible states of tours for $n_H = 32$ neurons. The first and third rows of each image show the visible states from the training data, whereas the second and fourth show the next visible state obtained through Gibbs Sampling 82

A.1 *RTS* Vs *Ripple* Accuracy in the 5-*CIS* pattern distribution estimation task (Section 2.6.1). I compare *Ripple* with *R-PSRW* by running ripple with the Partition-Error-Bound parameter set to 0.003 as was done in *Ripple*'s evaluation [37]. I compare the error of *Ripple* estimates with that of *R-PSRW* estimates given the same wall clock time, and show box plots representing 10 runs. It is clear that *R-PSRW* converges to the ground truth faster on all the datasets I tested. . . . 95

B.1 Parallel *RWT*s and Reservoirs: Figure B.1a shows the set of m *RWT*s sampled on $\mathcal{G}_2$ in parallel, where the supernode ζ_2 is colored black. The gray, blue, red and green colors represent states in stratum 2–5, respectively. Figure B.1b shows the upper triangular reservoir matrix in which the cell in the r-th row and t-th column contains samples from $\widehat{\mathbf{U}}_{r,t}$. 113

B.2 Accuracy and convergence analysis for 5-*CIS* s using L-∞ norm. The y-axes show the L-∞ norm between the *Ripple* estimate and the ground truth vector $\mathcal{C}^{(5)}$, containing counts of all possible non-isomorphic subgraph patterns for various settings of ϵ and $\mathcal{I}_1$. As expected, the accuracy improves as ϵ decreases and $|\mathcal{I}_1|$ increases. Each box and whisker represents 10 runs. 118

B.3 Sensitivity of *Ripple* to the reservoir capacity M for $k = 5$. I verify that a larger reservoir improves the accuracy of *Ripple* estimates in all graphs because it reduces oversampling bias. Each box and whisker plot represents 10 runs. 119

B.4 Scalability w.r.t. the Number of Threads for 5-*CIS*. Despite a noticeable reduction in running time, the scalability is not linear. In fact, as we double the number of cores, the running time decreases by approximately a quarter instead of half, which may be due to the memory bandwidth limit coupled with the lack of memory locality, which is a ubiquitous problem in graph mining algorithms. . . 120

ABSTRACT

In recent years there has been a growing interest in data mining and graph machine learning for techniques that can obtain frequencies of k-node Connected Induced Subgraphs (k-*CIS*) contained in large real-world graphs. While recent work has shown that 5-*CIS* s can be counted exactly, no exact polynomial-time algorithms are known that solve this task for $k > 5$. In the past, sampling-based algorithms that work well in moderately-sized graphs for $k \leq 8$ have been proposed. In this thesis I push this boundary up to $k \leq 16$ for graphs containing up to $120M$ edges, and to $k \leq 25$ for smaller graphs containing between a million to $20M$ edges. I do so by re-imagining two older, but elegant and memory-efficient algorithms – *FANMOD* and *PSRW*– which have large estimation errors by modern standards. This is because *FANMOD* produces highly correlated k-*CIS* samples and the cost of sampling the *PSRW* Markov chain becomes prohibitively expensive for k-*CIS* s larger than $k > 8$.

In this thesis, I introduce:

(a) ***RTS*:** a novel regenerative Markov chain Monte Carlo (*MCMC*) sampling procedure on the tree, generated on-the-fly by the *FANMOD* algorithm. *RTS* is able to run on multiple cores and multiple machines (embarrassingly parallel) and compute confidence intervals of estimates, all this while preserving the memory-efficient nature of *FANMOD*. *RTS* is thus able to estimate subgraph statistics for $k \leq 16$ for larger graphs containing up to $120M$ edges, and for $k \leq 25$ for smaller graphs containing between a million to $20M$ edges.

(b) ***R-PSRW*:** which scales the *PSRW* algorithm to larger *CIS*-sizes using a rejection sampling procedure to efficiently sample transitions from the *PSRW* Markov chain. *R-PSRW* matches *RTS* in terms of scaling to larger *CIS* sizes.

(c) ***Ripple*:** which achieves unprecedented scalability by stratifying the *R-PSRW* Markov chain state-space into ordered strata via a new technique that I call *sequential stratified regeneration*. I show that the *Ripple* estimator is consistent, highly parallelizable, and scales well. *Ripple* is able to *count CIS*s of size up to $k \leq 12$ in real world graphs containing up to $120M$ edges.

My empirical results show that the proposed methods offer a considerable improvement over the state-of-the-art. Moreover my methods are able to run at a scale that has been considered unreachable until now, not only by prior *MCMC*-based methods but also by other sampling approaches.

Optimization of Restricted Boltzmann Machines. In addition, I also propose a regenerative transformation of *MCMC* samplers of Restricted Boltzmann Machines (*RBM*s). My approach, Markov Chain Las Vegas (*MCLV*) gives statistical guarantees in exchange for random running times. *MCLV* uses a stopping set built from the training data and has a maximum number of Markov chain step-count K (referred as *MCLV-K*). I present a *MCLV-K* gradient estimator (*LVS-K*) for *RBM*s and explore the correspondence and differences between *LVS-K* and Contrastive Divergence (*CD-K*). *LVS-K* significantly outperforms *CD-K* in the task of training *RBM*s over the `MNIST` dataset, indicating *MCLV* to be a promising direction in learning generative models.

1. INTRODUCTION

In this thesis, I consider the task of computing the average of some user-defined function f over the set $\mathcal{S}^{(k)}$ of all Connected Induced Subgraphs of a size k (k-*CIS*s) contained in a large real-world graph. The ability to compute this average – $\text{AVG}(\mathcal{S}^{(k)}; f)$ – is a key component of multiple tasks. For example, in social network analysis and in biological networks, one is interested in estimating the distribution (i.e. relative frequency) of induced subgraphs which are isomorphic to a given pattern (e.g. a clique, a cycle, a star) [1]–[5]. The pattern distribution is essentially the average of a 0–1 function that indicates when a subgraph is isomorphic to the pattern of interest. More generally, for many graph machine learning models, the loss function is defined as the average of a neural network embedding applied over all k-node *CIS*s [6]–[12].

Exactly computing this function average $\text{AVG}(\mathcal{S}^{(k)}; f)$, which is formally defined in Section 2.2.1, is not always tractable, in particular for large graphs and/or large *CIS* sizes, since the enumeration of all *CIS*s contained in a graph is #W[1]-hard [13]. Ergo, we do not yet know an algorithm to enumerate all k-*CIS*s contained in an input graph $G_{\text{IN}} = (V_{\text{IN}}, E_{\text{IN}})$ in $\text{O}(\gamma(k)|V_{\text{IN}}|^{o(1)})$ time, where $\gamma(\cdot)$ is arbitrary but finite. While an exact algorithm exists for the special case of computing pattern distributions for 5-*CIS*s [14], the general task for $k > 5$ is significantly harder [15]. As such, multiple algorithms have been proposed that can estimate $\text{AVG}(\mathcal{S}^{(k)}; f)$ via sampling [16]–[21].

For a *CIS* sampling algorithm to be practical to be used in real world tasks that require estimating function averages of the form $\text{AVG}(\mathcal{S}^{(k)}; f)$, specially in large graphs and for large *CIS* sizes, the algorithm (a) needs to be memory efficient – preferably fully in memory, (b) should produce estimates that converge to the true average in reasonable time – the ability to parallelize is an added benefit, and (c) cannot be restricted to certain subgraph sizes or pattern types. While state of the art methods work well in the paradigms they were designed for, their scalability is limited because they sacrifice at least one of the above desiderata.

Specifically, many algorithms that estimate $\text{AVG}(\mathcal{S}^{(k)}; f)$ by Monte Carlo sampling *CIS*s slow down significantly for large k due to the cost of computing the sampling probability, which is often factorial in k [16], [22], [23]. On the other hand, methods such as *MOSS*-5 [24],

Escape [14] and *IMPRG* [20] work extremely well when $k \leq 5$ but cannot be extended for larger k. Similarly, TURÁN-SHADOW [2] and *PEANUTS* [1] are not general purpose *CIS* methods since they are restricted to cliques and near-clique *CIS*s. Sampling the *PSRW* [18] Markov chain is not scalable to larger k, due to the large number of connectivity checks (which require $O(k^2)$ time each) required to enumerate the *neighborhood* of a *CIS*. A newer method that scales to general *CIS*s of size $k > 5$ – *Motivo* [19], [25], [26] – has memory complexity exponential in k. This exponential memory requirement is compensated for, using memory mapped files and using a highly compressed representation of *CIS*s. For the largest evaluated graph (`Orkut`, see Table A.1), *Motivo* can sample and estimate function averages for k up to 8 using high speed SSDs and a large RAM. Using commodity hardware (HPC cluster with network storage) however, it fails to scale beyond 5 node *CIS*s for the same graph.

This thesis explores methods to estimate *CIS*-statistics using *regenerative* Markov chain Monte Carlo (*MCMC*) methods. I develop Markov chains, from which transitions can be efficiently sampled in poly(k) time and memory[1], which allows them to be used in settings where other methods fail due to their complexity being exponential in k as described above. In particular, I improve the per-iterate running time of the state of the art *CIS*-sampling *MCMC* algorithm *PSRW* [18] allowing it to scale to larger k. In addition, I propose a novel Markov chain on the tree, which is implicitly built on-the-fly and traversed by the *FANMOD* [17] algorithm. In both these chains, I then use *regenerations* [27], [28] in the form of *tours* – sample paths that begin and end at a pre-designated regeneration point – to estimate the *CIS* statistics.

These *tours* are i.i.d. due to the Regenerative Cycle Theorem [28, Chap 2, Thm 7.4]. This fact, allows them to be sampled in an embarrassingly parallel fashion, taking full advantage of multi-core systems. The independence can further be leveraged to assess convergence via confidence intervals and in some cases reduce bias. Specifically, in Section 2.3.4, jackknife resampling [29], [30] is used to estimate the variance of the estimator and to reduce bias, and in Section 3.4.3, this independence is used to auto-decide the number of tours required to be sampled to achieve a desired accuracy. *Tour*-based estimators can further be used to

[1]↑poly(k) refers to an asymptotic order of complexity which is polynomial in k.

estimate un-normalized function sums of the form $\mu(\mathcal{S}^{(k)}; f) = \sum_{s \in \mathcal{S}^{(k)}} f(s)$ evaluated over $\mathcal{S}^{(k)}$, the set of all *CIS*s contained in the input graph – an objective that traditional *MCMC* methods cannot estimate.

While the methods I propose are closely tied to the problem of estimating *CIS* statistics, they are in essence applicable to other problems where Monte Carlo simulation using a finite state reversible Markov chain is performed. As an exemplar, we apply regenerative sampling on the Gibbs Sampling [31] Markov chain that is used to train Restricted Boltzmann Machines (RBMs) [32]–[34], a class of energy-based generative neural network models.

1.1 Contributions

My specific contributions to the task of estimating *CIS* statistics are

(a) ***RTS*:** I propose using weighted random walks on the tree, generated on-the-fly by the *FANMOD* algorithm. We use the fact that each k-depth leaf in this tree uniquely represents a k-*CIS* and the stationary distribution over such leaves is uniform to estimate *CIS* statistics. Regenerations are then used to (embarrassingly) parallelize the procedure, reduce bias and assess convergence. *RTS* retains the memory-efficient nature of *FANMOD* and therefore is able to estimate subgraph statistics for $k \leq 16$ for larger graphs containing up to $120M$ edges, and for $k \leq 25$ for smaller graphs containing between a million to $20M$ edges.

(b) ***R-PSRW*:** Although, the *PSRW* [18] algorithm has been shown to be extremely effective in estimating *CIS* statistics, it fails to scale to large *CIS* sizes due to the large order of complexity of sampling transitions from the *PSRW*-Markov chain. I propose a rejection sampling procedure to improve this time complexity in expectation, which allows us to sample transitions in the *PSRW*-chain for very large subgraph sizes. The resulting procedure, denoted *R-PSRW* matches the scalability of *RTS* described above, and in some cases produces faster converging estimates in practice (despite being a single-core procedure). A distinct rejection sampling procedure also forms the backbone of the *Ripple* algorithm described next.

(c) ***Ripple*:** Unlike *RTS*, sampling regenerations in the *R-PSRW* Markov chain is not trivial due to the *tour* lengths (and consequently the running time per tour) being very high. I propose a recursive procedure, named *sequential stratified regeneration*, which combines the idea of using state-aggregations in the form of *supernodes* [35] and stratification to control this running time. Using a single Breadth-First Traversal on the input graph, *Ripple* stratifies the state space of the *R-PSRW* chain without explicit enumeration. This stratification is then used to control the running time to the extent that the total number of random walk steps is linear in the number of tours, diameter and maximum degree – and invariant to the number of vertices and edges of the input graph. I show that the *Ripple* estimator is consistent, highly parallelizable, and is able to count *CIS*s of size up to $k \leq 12$ in real world graphs containing up to $120M$ edges.

In addition to estimating *CIS* statistics, I show that *tours* can be used in other finite-state reversible *MCMC* settings by using them to train *RBM*s:
While a useful tool for unsupervised learning, the primary challenge in using RBMs lies in the intractability of computing their partition function and of computing the gradient. Both these quantities can be expressed as an expectation of a function under a probability distribution over a very large state space. I propose a regenerative transformation of the *MCMC* algorithm denoted CD-K, which is traditionally used to train RBMs. I call my approach *Markov Chain Las Vegas* (MCLV). MCLV gives statistical guarantees in exchange for random running times by using a stopping set built from the training data. By appealing to the fact that the distribution of tour lengths is not heavy tailed (formalized in Theorem 4.2.2), my procedure discards tours which are longer than a hyper-parameter K. My gradient estimator LVS-K, outperforms baselines in training RBMs over the `MNIST` dataset. I then explore the correspondence and differences between LVS-K and Contrastive Divergence (CD-K). LVS-K significantly outperforms CD-K in training RBMs over the MNIST dataset, indicating MCLV to be a promising direction in learning generative models.

1.2 Outline and Previously Published Work

This thesis is organized as follows:

(a) In Chapter 2 I detail, both, the *RTS* and *R-PSRW* procedures. Further I show in this chapter that these methods scale to larger *CIS* sizes and their estimates of *CIS*-function averages converge to the ground truth faster in terms of wall-clock time compared to the state of the art. This chapter is under submission as "Regenerative Tree Sampling: Efficient Subgraph Statistics via Random Walks on Trees" Kakodkar and Ribeiro 2022 [36].

(b) Chapter 3 talks about *Ripple*, which in turn uses an alternate version of *R-PSRW* (which is more compatible with the *sequential stratification* procedure compared to the version from Chapter 2). I show that for the task of estimating *CIS*-function sums, *Ripple* scales better than the current state of the art method – *Motivo* [26]. This chapter was published as "Sequential stratified regeneration: MCMC for large state spaces with an application to subgraph count estimation" Teixeira, Kakodkar, Dias, *et al.* 2022 [37].

(c) Finally, in Chapter 4, I introduce Restricted Boltzmann Machines (RBMs) and the task of estimating its gradient and partition function. Further I detail the *MCLV* procedure, the *LVS-K* gradient estimator and experimental results that show that *LVS* outperforms baselines in training RBMs over the `MNIST` dataset. This chapter was published as "From Monte Carlo to Las Vegas: Improving Restricted Boltzmann Machine Training Through Stopping Sets" Savarese, Kakodkar, and Ribeiro 2018 [38].

2. REGENERATIVE TREE SAMPLING

2.1 Introduction

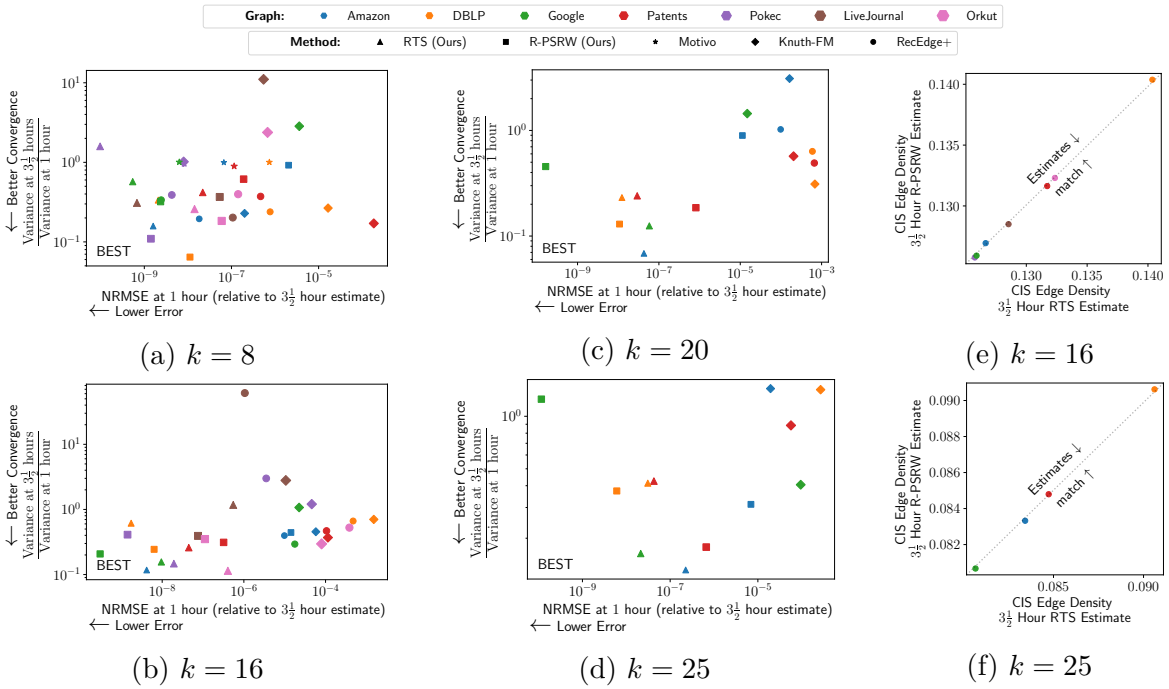
Consider the task of computing the average of some user-defined function f over the set $\mathcal{S}^{(k)}$ of all Connected Induced Subgraphs of a size k (k-*CIS*s) of a graph. The ability to compute this average – $\text{AVG}(\mathcal{S}^{(k)}; f)$ – is a key component of state-of-the-art graph based machine learning models [6]–[12] and data mining tasks such as social and biological network analysis [1]–[5].

This chapter proposes *Regenerative Tree Sampling* (*RTS*), a method that estimates bounded function averages over all k-*CIS*s of an input graph by using a specific type of random walk over the *FANMOD* [17] tree. The *FANMOD* tree is built-on-the-fly, and its vertices at depth k represent k-*CIS*s in the original input graph. The original *FANMOD* algorithm [17] samples a *randomized* depth-first traversal on the tree and uses the leaves visited by the traversal to estimate subgraph counts. In *RTS*, I significantly improve this sampling method with a weighted random walk on the *FANMOD* tree that samples k-*CIS*s uniformly at random in steady-state. Since the *FANMOD* tree is built on-the-fly, my procedure is highly memory-efficient. A key idea is the use of *regenerations* in the form of Random Walk Tours (*RWT*s), which are short random walks that begin and end at the initial node (root node in my case) and have been extensively used in Markov Chain Monte Carlo methods [35], [37]–[40]. The use of *tours* allow us to spawn many samplers in an embarrassingly parallel fashion, and their i.i.d. nature can be leveraged to better assess convergence compared to traditional random walk-based estimators. Furthermore, I provide a simple heuristic to set edge weights in the *FANMOD* tree to improve the quality of estimates.

In addition, I propose *R-PSRW*, an extension to the *PSRW* [18] algorithm which efficiently samples transitions of the *PSRW* Markov chain using a rejection sampling procedure. While the worst-case mixing times of the *PSRW* Markov chain are theoretically high [41], [42], in my experiments, *R-PSRW* estimates of *CIS* function averages converge to the true value better than all the other baselines I tested (and in a few cases even better than *RTS*).

My results show that both *RTS* and *R-PSRW* can estimate function averages for *CIS*s of size up to $k = 16$ on a wide variety of input graphs (see Table A.1) and in smaller graphs,

up to $k = 25$. This phenomenon can be observed in Figure 2.1 where I show that *RTS* and *R-PSRW* estimators have lower error and converge faster compared to all the other tested baselines, with *RTS* estimates showing superior convergence more often.



(a) $k = 8$ (c) $k = 20$ (e) $k = 16$

(b) $k = 16$ (d) $k = 25$ (f) $k = 25$

Figure 2.1. *RTS* and *R-PSRW* accuracy in the *CIS* edge density estimation task from Section 2.6.2. The average edge density of all *CIS*s is estimated using each method over 10 runs with wall-clock limits of 1 hour and 3½ hours. Figures 2.1a to 2.1d show the convergence properties of the *CIS* edge density estimates for $k \in \{8, 16, 20, 25\}$ and all methods and real-world graphs. The x-axis represents the NRMSE of the 1 hour estimates computed using the median of 3½ hour estimates of the same method as the ground-truth. The y-axis represents the reduction in variance between the 1 hour and 3½ hour estimates. Marker shapes represent various methods, colors represent graphs, and larger markers represent larger graphs. For each color, methods corresponding to markers closer to the bottom-right corner have better convergence properties. *RTS* and *R-PSRW* estimates have better convergence properties compared to the other tested baseline methods, with *RTS* estimates showing superior convergence more often. In Figures 2.1e and 2.1f I show that the estimates of my proposed *RTS* and *R-PSRW* for $k \in \{16, 25\}$ agree, which I take (in the absence of ground-truth) as strong evidence that these estimates are accurate.

2.2 Preliminaries

Before I describe the *CIS* sampling tasks, first we need some notation:

The *original input graph* is any finite graph, with or without node and edge features, and may contain multi-edges, self-loops, etc. Any such graph can be converted to a simple and undirected graph $G_{\text{IN}} = (V_{\text{IN}}, E_{\text{IN}})$ by encoding properties such as edge directions, etc. as vertex and edge features, which are accessible to any function defined on G_{IN} and the *CIS*s it contains. As such G_{IN}, which is henceforth denoted the *input graph*, does not contain self-loops and each pair of vertices share at most one edge.

Without loss of generality we can write the vertex set as $V_{\text{IN}} = \{1, 2, \ldots, |V_{\text{IN}}|\}$. Moreover, edges of G_{IN} are undirected $(u, v) \equiv (v, u)$ and , hence, are only counted once in the edge-set E_{IN}. The neighborhood of a set of vertices $V' \in V_{\text{IN}}$ is defined as $\mathbf{N}(V') \triangleq \cup_{u \in V'}\{v \in V_{\text{IN}} \colon (u, v) \in E_{\text{IN}}\,,\, v \notin V'\}$ and with a slight abuse of notation, $\mathbf{N}(u) \equiv \mathbf{N}(\{u\})$. The degree of a vertex is then $\deg(u) \triangleq |\mathbf{N}(u)|$. The volume of G_{IN} is $\text{Vol}(G_{\text{IN}}) = \sum_{u \in V_{\text{IN}}} \deg(u) = 2|E_{\text{IN}}|$.

The vertices and edges of G_{IN} are simply referred as vertices and edges. The vertices and edges of the *FANMOD* tree $\mathcal{T}^{(k)}$ (defined in Section 2.2.2) will be referred as tree-nodes and tree-edges to avoid confusion. All the trees considered in this chapter are rooted with a single root, denoted as ρ. The set of children of a tree-node x are denoted $\mathcal{C}(x)$, and the parent of x is given by $\mathcal{P}(x)$. The tree-width at node x is defined as $|\mathcal{C}(x)|$. The depth of x, $\textsc{depth}(x)$, is the number of edges in the shortest path between the root ρ and x, i.e. the root itself would be at depth zero.

2.2.1 Task

Having defined the input graph, we now define *CIS*s:

Definition 2.2.1 (Connected Induced Subgraph (*CIS*))**.** *Consider as given, an input graph $G_{IN} = (V_{IN}, E_{IN})$, which is assumed to be simple and undirected (defined above). A Connected Induced Subgraph of size k or k-*CIS *is a simple,* **connected** *and undirected graph $G_{IN}(V') = (V', E')$ whose vertex set $V' \subset V_{IN}$ follows $|V'| = k$ and whose edge set is given by $E' = \{(u, v) \in E_{IN} \colon u, v \in V'\}$. Additionally, $\mathcal{S}^{(k)}$ is the set of all k-*CIS *contained in G_{IN}.*

Practitioners are usually interested in methods which estimate the average of an arbitrary function f (generally with a real-vector image) over $\mathcal{S}^{(k)}$, the set of all *CIS*s contained in G_{IN}.

Definition 2.2.2 (*CIS* Average Task)**.** *Assume as given, an input graph G_{IN},* CIS *size k, and a bounded real vector-valued function $f\colon \mathcal{S}^{(k)} \to \mathbb{R}^{(\cdot)}$. The* CIS *Average Task requires estimating*

$$\textsc{avg}(\mathcal{S}^{(k)}; f) = \frac{1}{|\mathcal{S}^{(k)}|} \sum_{s \in \mathcal{S}^{(k)}} f(s)\,, \tag{2.1}$$

*where $\mathcal{S}^{(k)}$ denotes the set of all k-*CIS*s in G_{IN} from Definition 2.2.1.*

The above task-definition extends to many real-world applications of *CIS* sampling, for example:

- ***k*-*CIS* distributions:** When $f_{\text{PAT}}(s)$ is a vector-valued function, which one-hot encodes the pattern (graph-isomorphism based equivalence class) that s is isomorphic to, the task of estimating $\textsc{avg}(\mathcal{S}^{(k)}; f_{\text{PAT}})$ is equivalent to the quintessential *CIS* pattern distribution estimation task, which is central to multiple *CIS* sampling works [16]–[18], [20], [21], [26], [37].
- **Graph representation learning:** More generally, setting $f_{\text{ML}}(s)$ to be the decision function output of a Machine Learning (ML) model applied to the *CIS*, leads to the loss functions used in recent work in graph ML [6]–[12].

2.2.2 *FANMOD* tree and its original algorithm

Before I describe *RTS*, we briefly introduce the built-on-the-fly tree used in the *FANMOD* algorithm [17]. Given the graph G_{IN} and a *CIS* size k, the *FANMOD* tree $\mathcal{T}^{(k)}$ associated with G_{IN}, is a k-depth, rooted tree. The root node corresponds to the *empty CIS*. Every other tree-node at a depth k' corresponds to a k'-*CIS* contained in G_{IN}. Depth k tree-nodes are leaves and represent k-*CIS*s.

Since the number of *CIS*s of size k in G_{IN} is exponential in k, explicitly constructing $\mathcal{T}^{(k)}$ is not tractable. As such, the tree is explored via *Child Queries*, where we define a

function that returns the children $\mathcal{C}(x)$ of a given tree-node x. To that end, we first define the properties of a tree node.

Definition 2.2.3 (Tree-node properties)**.** *Each tree-node x in the* FANMOD *tree $\mathcal{T}^{(k)}$, is endowed with three properties, $\textsc{vset}_x \subset V_{\text{IN}}$ denotes the vertex set that makes up the corresponding* CIS*, $\textsc{main}_x \in V_{\text{IN}}$ is the main vertex which corresponds to the 1-depth ancestor of x and $\textsc{ext}_x \subset V_{\text{IN}}$ which contains candidate vertices which may be added to $\textsc{vset}_x$ to expand the* CIS*. The root node is a special case denoted by ρ and does not have any of the above properties.*

Next we use these definitions to further define the child query used to explore/generate the *FANMOD* tree as follows.

Definition 2.2.4 (*FANMOD*)**.** *Given the input graph G_{IN} whose vertices are assumed to be $V_{\text{IN}} = \{1, 2, \ldots, |V_{\text{IN}}|\}$ by convention, the set of children of any tree-node x in the* FANMOD *tree for sampling k-*CIS*s, $\mathcal{T}^{(k)}$, is defined as follows:*

- *When x represents a k-*CIS*, i.e. $|\textsc{vset}_x| = k$, x is a leaf node, i.e. $\mathcal{C}(x) = \emptyset$.*
- *When x is the root node, for each $u \in V_{\text{IN}}$, there exists a child $y \in \mathcal{C}(x)$ where $\textsc{vset}_y = \{u\}$, $\textsc{main}_y = u$ and $\textsc{ext}_y = \{v \in \mathbf{N}(u) \colon v > u\}$.*
- *When x is any other tree-node, there exists a child $y \in \mathcal{C}(x)$ corresponding to each vertex $u \in \textsc{ext}_x$ where $\textsc{vset}_y = \textsc{vset}_x \cup \{u\}$, $\textsc{main}_y = \textsc{main}_x$ and $\textsc{ext}_y = \{w \in \textsc{ext}_x \colon w > u\} \cup \{v \in \mathbf{N}_{\textsc{excl}}(u, \textsc{vset}_x) \colon v > u\}$*

where the tree-node properties MAIN*,* EXT *and* VSET *are from Definition 2.2.3 and the exclusive neighborhood of a vertex $u \in V_{\text{IN}}$ relative to a set $V' \subset V_{\text{IN}} \backslash \{u\}$ is given by $\mathbf{N}_{\textsc{excl}}(u, V') \triangleq \mathbf{N}(u) \backslash \mathbf{N}(V')$.*

From the above definition, it is clear that the *FANMOD* tree, $\mathcal{T}^{(k)}$ is a rooted tree, with maximum depth k. Moreover, the extensive manipulation of the the extension-set ensures that each *k-CIS*, $s \in \mathcal{S}^{(k)}$ contained in G_{IN} is uniquely represented by k-depth leaf in $\mathcal{T}^{(k)}$. As visualized in Figure 2.2, a one-one correspondence exists between each *CIS* and tree-node. More formally:

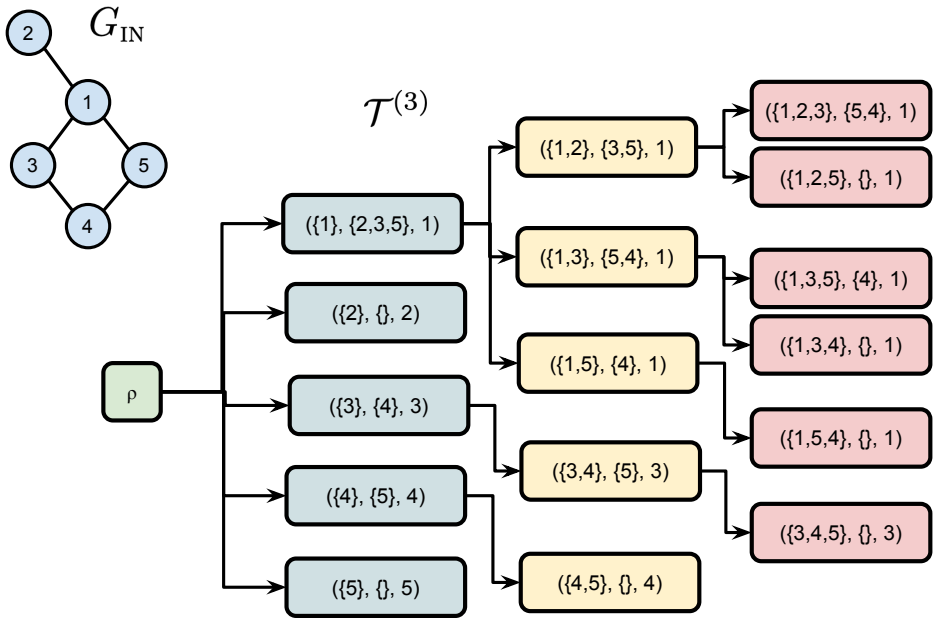


Figure 2.2. The *FANMOD* tree, $\mathcal{T}^{(3)}$, for sampling 3-*CIS*s in G_{IN}, rooted at ρ. Each tree-node contains the $(\text{VSET}, \text{EXT}, \text{MAIN})$ properties from Definition 2.2.3. Note how each 3-*CIS* member of $\mathcal{S}^{(3)}$ is *uniquely* represented in the *FANMOD* tree.

Theorem 2.2.1 (Wernicke [17] Theorem 2)**.** *Recall $\mathcal{S}^{(k)}$, the set of all k-*CIS*s from Definition 2.2.1. For each* CIS *$s \in \mathcal{S}^{(k)}$, there exists exactly one leaf at depth k of the* FANMOD *tree $\mathcal{T}^{(k)}$ generated by the algorithm defined in Definition 2.2.4.*

Theorem 2.2.1 was exploited in the original *FANMOD* algorithm: the leaves sampled by a randomized Depth First Traversal (DFT) on the tree, are drawn uniformly at random from $\mathcal{S}^{(k)}$. Specifically, *FANMOD* chooses to explore edges with a certain probability which depends on the *depth* of the edge.

FANMOD significantly improved upon the O($k!$) time to compute the sampling bias in *RecEdge* [16], which also sampled *CIS*s via recursive expansion, but without the book-keeping introduced by *FANMOD*. Despite having an elegant implementation and being memory efficient, *FANMOD* has two drawbacks:

- *FANMOD* requires careful hand-tuning of parameters, especially when k is large, because a high acceptance probability will lead to an extremely large number of samples, and hence a higher time complexity. Terminating the procedure mid-way is not an option, because the guarantee that leaves are sampled uniformly only holds when the entire procedure is complete.

- The samples drawn via the randomized DFT are highly correlated. The correlation is because, when an edge leading up to a tree-node is dropped, the entire subtree is dropped and vice-versa.

Consequently, *FANMOD* estimates suffer larger error compared to subsequently proposed methods such as *PSRW* [18, Sec 4]. As such I aim to leverage the basic insight of *FANMOD*– the fact that each *k-CIS* can be represented uniquely as the leaves of an arbitrarily large tree – to build a better *CIS* sampling algorithm without sacrificing the memory efficiency.

2.3 Regenerative Tree Sampling

In what follows I propose my method, abbreviated as *RTS*. I begin by describing the Markov chain on the *FANMOD* tree, and show that, in steady state, this chain samples k node subgraphs (*CIS*s) uniformly at random from $\mathcal{S}^{(k)}$, the set of all *k-CIS*s contained in

G_{IN} as defined in Definition 2.2.2. I then define a Random Walk Tour on the above chain, used to estimate the average of any bounded function f over $\mathcal{S}^{(k)}$. Next, I show that the *RTS* estimator can be computed embarrassingly in parallel, and discuss practical methods to analyze the quality of estimates. Finally, I describe how *RTS* is implemented in practice and its memory complexity.

2.3.1 Motivation: Knuth-sampling the *FANMOD* tree

I motivate my method (*RTS*) by examining an alternate leaf sampling algorithm, different than the original weighted DFT but applied to the same *FANMOD* tree. Specifically I look at Knuth's leaf sampling algorithm [43]. Knuth's algorithm samples a non-backtracking random walk from the root to a k-depth leaf, $X_0, \ldots, X_k$, where $X_0 = \rho$ (the root node) and subsequently, $X_t \sim \textsc{unif}(\mathcal{C}(X_{t-1}))$, $\mathcal{C}(\cdot)$ being the child query from Definition 2.2.4. The k-depth leaf X_k corresponds to a *k-CIS* whose probability of sampling is denoted by $b(X_k) = \prod_{\text{i}=0}^{k-1} 1/|\mathcal{C}(X_{\text{i}})|$.

Lemma 1. *Consider the* FANMOD *tree* $\mathcal{T}^{(k)}$ *generated according to Definition 2.2.4 and the subgraph-function* f *as in Definition 2.2.2. Run Knuth's leaf sampling algorithm* n *times. Let* k-CIS*s,* $X_{k,1}, \ldots, X_{k,n}$ *be the* n FANMOD *leaves sampled by the algorithm. Then, the importance sampled Knuth-estimator given by*

$$\hat{\mu}_{KNUTH}\left(\{X_{k,\text{i}}\}_{\text{i}=1}^{t}; f\right) = \left(\sum_{\text{i}=1}^{t} \frac{f(X_{k,\text{i}})}{b(X_{k,\text{i}})}\right) \Big/ \left(\sum_{\text{i}=1}^{t} \frac{1}{b(X_{k,\text{i}})}\right), \tag{2.2}$$

is an asymptotically unbiased estimator of $\textsc{avg}(\mathcal{S}^{(k)}; f)$ *– the average of* f *over all the* k-CIS*s contained in* G_{IN} *per Definition 2.2.2.*

While Knuth's algorithm is hyperparameter-free and eliminates the correlation between samples, the variance of $\hat{\mu}_{\text{KNUTH}}$ is very large due to the very disparate probabilities of sampling different *k-CIS*s as shown next:

Proposition 2.3.1. *Let* $P(X_k = s)$ *be the probability of sampling a* k-CIS $s \in \mathcal{S}^{(k)}$ *using Knuth's algorithm. The variability of* $b(X_k)$ *defined as* $\max_{s \in \mathcal{S}^{(k)}} P(X_k = s) \big/ \min_{s \in \mathcal{S}^{(k)}} P(X_k = s)$ *is in* $\text{O}(k! \Delta_{IN}^{k})$*, where* Δ_{IN} *is the maximum vertex degree in the input graph* G_{IN}*.*

Hence, using Knuth's algorithm to estimate the function average $\textsc{avg}(\mathcal{S}^{(k)}; f)$ (Definition 2.2.2) for large k becomes impractical due to the high variance of the estimator $\hat{\mu}_{\textsc{knuth}}$ in Equation (2.2).

2.3.2 Fast and parallel uniform leaf sampling

When the *FANMOD*-tree is treated like an undirected graph, all k-depth leaves have equal degree. The stationary distribution of a simple random walk (RW) will hence be the same for all leaves, since it is proportional to the node's degree. But, using a RW has other challenges:

- Since undirected-unweighted trees have a low conductance [44, Section 5], RWs on the *FANMOD* tree will mix very slowly, which in turn creates sampling biases.
- Samples generated via RWs are correlated, which is precisely the drawback that the original *FANMOD* algorithm had and that the Knuth-sampling algorithm from Section 2.3.1 was designed to fix.
- It is non-trivial to assess the mixing of RW-based estimates in practice. Mixing also imposes constraints to the usefulness of running multiple RWs in parallel (since the bias due to mixing does not go down with the number of parallel random walks).

Towards addressing these challenges, I first improve mixing by increasing the conductance of the *FANMOD* tree via an edge-weighting scheme. I then use *Regenerations* in the form of Random Walk Tours (*RWT*s) [35], [37]–[40], which are short walks that begin and end at the root node to estimate the function average $\textsc{avg}(\mathcal{S}^{(k)}; f)$ from Definition 2.2.2. By definition, these *RWT*s can be sampled embarrassingly in parallel. Moreover, their independence can be leveraged to assess estimator convergence and reduce bias. Their independence also limits the correlation between samples (although samples from the same tour are correlated, those in two different tours are uncorrelated). In particular, due to the independence of *RWT*s, the variance of my estimator reduces as $\mathrm{O}(1/n)$, where n is the number of *RWT*s.

2.3.3 *RTS* on the *FANMOD* tree

The basic building-block of the *RTS* method is a weighted random walk on the *FANMOD* tree, which is defined as follows:

Definition 2.3.1 (Tree Sampling Markov chain (*TSMC* $\mathbf{\Phi}^{(k)}$))**.** *Given the* FANMOD*-tree* $\mathcal{T}^{(k)}$ *whose* k*-depth leaves represent the set of* k-CIS*s contained in* G_{IN}*. Let* $\mathcal{X}$ *be the set of tree-nodes contained in* $\mathcal{T}^{(k)}$*. We define the* TSMC*,* $\mathbf{\Phi}^{(k)}$*, as a Markov chain with state space* $\mathcal{X}$ *and with transition probability between* $x, y \in \mathcal{X}$ *given by*

$$p_{\mathbf{\Phi}^{(k)}}(y|x) \propto \begin{cases} \alpha_{\text{DEPTH}(x)} & \text{when } y \equiv \mathcal{P}(x) \\ 1 & \text{when } y \in \mathcal{C}(x) \end{cases}, \tag{2.3}$$

where $\text{DEPTH}(x)$ *is the depth of the tree-node* x*,* $\alpha_0, \ldots, \alpha_k$ *are positive parameters indexed by the depth, which define the probability of choosing the parent node* $\mathcal{P}(x)$ *relative to a child in* $\mathcal{C}(x)$*. Since the root-node* ρ *does not have parents, we assume* $\alpha_0 = 0$*, and since* k*-depth tree-nodes don't have children,* $\alpha_k = 1$*.*

The *TSMC* is a (simple) random walk on an undirected, carefully weighted version of the *FANMOD* tree $\mathcal{T}^{(k)}$. The weights are defined such that, for a tree-node at depth $\text{DEPTH}(x)$, an edge leading to a parent has weight which is $\alpha_{\text{DEPTH}(x)}$ times that of an edge leading to a child. Note that the actual edge weights are a function of the $\alpha_{(\cdot)}$ parameters and are precisely defined in Section A.3-Proposition A.3.1. I defer the discussion on setting these weights to Section 2.3.5. The *TSMC* sampling can be nearly as memory-efficient as the Knuth-sampling algorithm on the *FANMOD* tree:

Proposition 2.3.2. *Consider the access model, where our algorithm has apriori access solely to the root-node* ρ *of the* FANMOD*-tree from Definition 2.2.4. All other states can only be accessed using the child queries* $\mathcal{C}(\cdot)$ *from Definition 2.2.4. The* TSMC $\mathbf{\Phi}^{(k)}$ *can be sampled as long as the algorithm has enough memory to store* k *tree-nodes.*

Next, I state some basic properties of the *TSMC*.

Lemma 2. *The weighted random walk* $\mathbf{\Phi}^{(k)}$ *of Definition 2.3.1 over the tree-nodes* $\mathcal{X}$ *of the* FANMOD *tree* $\mathcal{T}^{(k)}$ *is a time-homogenous, reversible, irreducible Markov chain with a unique steady-state. If* $\mathcal{L} \subset \mathcal{X}$ *is the set of leaves of* $\mathcal{T}^{(k)}$ *at depth* k*, then for any* $x, y \in \mathcal{L}$*,* $\boldsymbol{\pi}_{\mathbf{\Phi}^{(k)}}(x) = \boldsymbol{\pi}_{\mathbf{\Phi}^{(k)}}(y)$*.*

Lemma 2 states that if we initialize a chain $\mathbf{\Phi}^{(k)}$ at the root-node ρ and sample it for a sufficiently long number of steps, the sampling distribution asymptotically becomes uniform on k-depth leaves. The required number of steps to converge, is formally referred to as the mixing time of the chain, and after that many steps, states visited by the chain are distributed according to $\boldsymbol{\pi}_{\mathbf{\Phi}^{(k)}}$.

Since $\boldsymbol{\pi}_{\mathbf{\Phi}^{(k)}}$ is the same for all k-depth leaves (recall that a leaf in $\mathcal{T}^{(k)}$ is a *k-CIS*), the random walk sampling can be used to sample *k-CIS*s uniformly at random. Moreover, due to the Convergence Theorem [44, Thm 2.2], a simple average over the leaves visited by this chain asymptotically converges to $\textsc{avg}(\mathcal{S}^{(k)}; f)$, the average of any bounded function f over $\mathcal{S}^{(k)}$ of Definition 2.2.2.

2.3.4 Estimators from $\mathbf{\Phi}^{(k)}$ regenerations

Next, I propose using regenerations in the form of *tours* to compute the estimates of $\textsc{avg}(\mathcal{S}^{(k)}; f)$, the average of the function f over the set of all *k-CIS*s contained in G_{IN}. Doing so makes our algorithm embarrassingly parallel and aids in estimating its convergence. To this end, we first need to define *rooted random walk tours*.

Definition 2.3.2 (Rooted Random Walk Tour (*RWT*))**.** *Given the* TSMC $\mathbf{\Phi}^{(k)}$ *of Definition 2.3.1 over the tree-nodes* $\mathcal{X}$ *of the* FANMOD *tree* $\mathcal{T}^{(k)}$*, the* RWT $\mathbf{R}$ *is the sample path of* $\mathbf{\Phi}^{(k)}$ *between two consecutive visits to the root-node* ρ*. Formally,* $\mathbf{R} = (X_1, X_2, \ldots, X_\xi)$ *where* $X_{\mathrm{i}} \in \mathcal{X}$*,* $\mathrm{i} = 1, \ldots, \xi$ *and* $X_1 = \rho$*. Transitions are sampled according to* $X_{\mathrm{j}+1} \sim p_{\mathbf{\Phi}^{(k)}}(x|X_{\mathrm{i}})$ *(Definition 2.3.1) and the tour length* ξ *is a random variable such that* X_ξ *is the last state visited by the chain before returning to* ρ *(that is,* $X_{\xi+1} = \rho$*).*

*RWT*s are independent and identically distributed due to the Regenerative Cycle Theorem [45, Thm 2.5.11]. Moreover, when *stitched* together, the concatenated sample path is

governed by $\mathbf{\Phi}^{(k)}$ due to the strong Markov property [45, Thm 2.5.11]. This fact can be used to estimate $\textsc{avg}(\mathcal{S}^{(k)}; f)$, the average of any function f over $\mathcal{S}^{(k)}$ using a set of *RWT*s as follows:

Theorem 2.3.1 (Ratio of Tours Estimate)**.** *Given a set of* n RWT*s* $\mathcal{R}^{(n)} = \{\mathbf{R}_1, \ldots, \mathbf{R}_n\}$ *sampled according to Definition 2.3.2, the Ratio of Tours Estimate is defined as*

$$\hat{\mu}_{ROT}(\mathcal{R}^{(n)}; f) = \sum_{\mathrm{i}=1}^{n} \sum_{\mathrm{j}=1}^{\xi_\mathrm{i}} \mathbf{1}\{X_{\mathrm{i,j}} \in \mathcal{L}\} f(X_{\mathrm{i,j}}) \Big/ \sum_{\mathrm{i}=1}^{n} \xi_\mathrm{i} \tag{2.4}$$

where ξ_i *is the length of the* i*-th tour,* $X_{\mathrm{i,j}}$ *is the* j*-th state visited by the* i*-th tour and* $\mathcal{L} \subset \mathcal{X}$ *is the set of* k*-depth leaves. We assume that* $f(X)$ *is the function* f *evaluated on the* CIS *associated with the leaf node* X*. Then, the Ratio of Tours Estimate is an asymptotically unbiased (in the number of tours) estimate of* $\textsc{avg}(\mathcal{S}^{(k)}; f)$*, i.e.* $\lim_{n\to\infty} \hat{\mu}_{ROT}(\mathcal{R}^{(n)}; f) \overset{a.s.}{=} \textsc{avg}(\mathcal{S}^{(k)}; f)$.

Recall the traditional *MCMC* estimator, which computes an empirical average over states visited by a long Markov chain. It is clear that the Ratio of Tours Estimate from Equation (2.4) is essentially an empirical average computed over the sample path obtained by concatenating *RWT*s. However, there is a key difference. Unlike a traditional *MCMC* sample path of pre-defined length, the *RWT*-concatenated sample path is guaranteed to contain i.i.d. sub-sequences. This fact can be leveraged to empirically assess convergence – which is non-trivial for traditional *MCMC*.

We can now use the jackknife resampling technique [29], [30], which for $\mathrm{i} = 1, \ldots, n$, computes jackknife replicates $\hat{\mu}_{\mathrm{ROT}}(\mathcal{R}^{(n)}_{-\mathrm{i}}; f)$, where, $\mathcal{R}^{(n)}_{-\mathrm{i}}$ is the set of *RWT*s, with the i-th *RWT* removed. The set of replicates $\{\hat{\mu}_{\mathrm{ROT}}(\mathcal{R}^{(n)}_{-1}; f), \ldots, \hat{\mu}_{\mathrm{ROT}}(\mathcal{R}^{(n)}_{-n}; f)\}$ then gives us an approximation of the distribution of estimates, whose empirical mean and variance can be used to compute the final estimate and the variance of Equation (2.4). The jackknife estimator also allows us to reduce the (finite n) bias of Equation (2.4) [29, Comment 6]:

$$\hat{\mu}^{\mathrm{JACK}}_{\mathrm{ROT}}(\mathcal{R}^{(n)}; f) = n\, \hat{\mu}_{\mathrm{ROT}}(\mathcal{R}^{(n)}; f) - \frac{n-1}{n} \sum_{\mathrm{i}=1}^{n} \hat{\mu}_{\mathrm{ROT}}(\mathcal{R}^{(n)}_{-\mathrm{i}}; f). \tag{2.5}$$

One can also estimate the variance of $\hat{\mu}_{\text{ROT}}(\mathcal{R}^{(n)}; f)$ using Taylor-series approximations, when the variance of *RWT*-lengths is small [46].

Another advantage of the Ratio of Tours Estimate is its embarrassingly parallel computation (i.e., without inter-thread communication). Let us define $\textsc{tour}\textsc{sum}(\mathbf{R}_{\text{i}}) = \sum_{\text{j}=1}^{\xi_{\text{i}}} \mathbf{1}\{X_{\text{i,j}} \in \mathcal{L}\} f(X_{\text{i,j}})$, which is the inner sum in the numerator of Equation (2.4). We can then task each thread $\text{i} \in \{1, \ldots, n\}$ to sample $\mathbf{R}_{\text{i}}$ and compute $\textsc{tour}\textsc{sum}(\mathbf{R}_{\text{i}})$. These, per-thread estimates can then be reduced to yield the estimate in Equation (2.4). Additionally, all methods discussed above to compute confidence intervals and reduce bias are compatible with this reduction step. This makes *RTS* very scalable.

The variance of the Ratio of Tours Estimate is $\text{O}(1/n)$, where n is the number of *RWT*s sampled [46, Sec 2], a fact which is used in Section 2.3.5 to set edge weights which maximize the number of tours per unit time.

2.3.5 *RTS* in practice: Setting edge weights

The asymptotic unbiasedness of the Ratio of Tours Estimate from Theorem 2.3.1 guarantees that *RTS* estimates will eventually converge to $\textsc{avg}(\mathcal{S}^{(k)}; f)$, the average of f over the set of all k-*CIS* s in the input graph. In practice, this convergence depends both on the probability of sampling k-depth leaves in an *RWT*, and on the probability of returning to the root-node. In the entire procedure defined above, the only hyperparameters that are available to be tuned to control this behavior are the relative backtracking weights, $\alpha_1, \ldots, \alpha_{k-1}$ from Definition 2.3.1. I propose the following heuristic to set these hyperparameters.

First I simplify the task, by using a single hyperparameter $w > 0$ which *RTS* uses to set the edge weights. For each $\text{i} = 1, \ldots, k-1$, I set the relative backtracking weight as $\alpha_{\text{i}} = w\bar{d}_{\text{i}}$, where $\bar{d}_{\text{i}}$ is the estimated, average tree-width of all tree-nodes at depth i. Recall that the tree-width for each tree-node x is defined as the number of its children, $|\mathcal{C}(x)|$. These estimates of the average depth-wise tree widths are computed by sampling leaves using a small number of non-backtracking random walks from the root to k-depth leaves, similar to the Knuth-sampling algorithm in Section 2.3.1.

Intuitively, setting the hyper-parameter w to be very low, would lead to very long *RWT*s and consequently, large running times. Alternately, a large value of w would lead to *RWT*s that do not visit leaves. This can also be seen in the idealized case, where the average tree-widths are known before hand:

Proposition 2.3.3. *Assume that the average tree-width at each depth* d_{i} *is known exactly for each* $\mathrm{i} = 1, \ldots, k-1$. *When the relative back-tracking edge weights are set as* $\alpha_{\mathrm{i}} = wd_{\mathrm{i}}$, *the expected length of an* RWT *from Definition 2.3.2 is in* $\mathrm{O}(1/w)$ *and the number of leaves sampled in each such* RWT *is in* $\mathrm{O}(1/w^{k-1})$.

As such we need to choose a value of w which is large enough to reduce the tour length but not so large, that it does not sample leaves. To find such a parameterization in practice, I run a grid-search over various values of w, and choose values that maximize the number of *RWT*s that contained leaves that were sampled per unit wall-clock time.

In fact, the interplay between the requirement that the *RWT*, both, returns to the root node and samples a k-depth node is the limiting factor in scaling my method to larger k. As we will see later, this dependence on memory is negligible (linear in k without any further optimization).

2.3.6 Memory complexity

Since the *FANMOD* tree only provides child-query access in Definition 2.2.4, to enable parent-query access we store the ancestors of the tree-node the random walker is at. As such:

Proposition 2.3.4 (*RTS* Memory Requirement)**.** *The memory requirement of sampling the* TSMC *from Definition 2.3.1 is in* $\mathrm{O}(k|V_{\mathrm{IN}}|)$.

While Theorem 2.2.1 can be exploited to reduce the memory requirement to $\mathrm{O}(|V_{\mathrm{IN}}|)$ in lieu of increasing time complexity, I prefer to cache the ancestors, since the linear complexity was not a limiting factor in my experiments. If the memory complexity of storing estimates is ignored (which are of the size of the desired output), there is no further memory overhead in the *RTS* procedure.

2.4 Scaling *PSRW* to Large k

The *PSRW* algorithm proposed by Wang, Lui, Ribeiro, *et al.* [18], estimates function averages $\text{AVG}(\mathcal{S}^{(k)}; f)$ (Definition 2.2.2) over the set of k-*CIS*s using stationary random walks on the Higher Order Network (*HON*) defined as follows.

Definition 2.4.1 (Higher Order Network)**.** *Given the input graph G_{IN} and the* CIS *size of interest k, the $k-1$-th order* HON *is a simple, undirected graph $\mathcal{G}^{(k-1)} = (\mathcal{S}^{(k-1)}, \mathcal{E}^{(k-1)})$, whose set of vertices is equivalent to the set of all $k-1$-*CIS*s contained in G_{IN} (Definition 2.2.1) and where an edge exists between a pair of* CIS*s if they share $k-2$ vertices. That is $(x, y) \in \mathcal{E}^{(k-1)}$ if $|V(x) \cap V(y)| = k-2$, where $V(s)$ refers to the vertices of s in the input graph.*

It is easy to see that each edge $(x, y) \in \mathcal{E}^{(k-1)}$ corresponds to a k-*CIS*, since $|V(x) \cup V(y)| = k$ and these k vertices are necessarily connected. The fact that the stationary of the random walk on the *HON* is uniform on its edges and the number of repetitions of each *CIS* in $\mathcal{E}^{(k-1)}$ can be efficiently computed is used by *PSRW* to estimate $\text{AVG}(\mathcal{S}^{(k)}; f)$.

These random walks are memory efficient, since only one *CIS* needs to be kept in memory and reservoir sampling over the neighbors can be used to sample transitions. For large k, however, enumerating the neighborhood of a $k-1$-*CIS* gets prohibitively expensive.I significantly reduce this complexity via the rejection sampling procedure in Algorithm 1.

Proposition 2.4.1. *Given the $k-1$-*HON *defined above, the naive procedure to enumerate the neighbors of a* CIS *x requires $\text{O}(k|\mathbf{N}(V(x))|)$ graph traversals (over a k-*CIS*), where $V(x)$ is the set of vertices contained in the* CIS*, x. The rejection sampling procedure from Algorithm 1 samples a* CIS *from the neighborhood of the original* CIS*, x, uniformly at random in $\text{O}(|\text{AP}_x|) \in \text{O}(k)$ expected graph traversals, where AP_x is the set of articulation points [47] of x.*

In practice, we observe that the number of rejections is $\ll \mathbf{N}(V(\cdot))k$, allowing *R-PSRW* to scale much better than *PSRW*. The rest of the algorithm is exactly the same as *PSRW* and I refer the reader to the original paper for the exact form of the estimator and to Matsuno and Gionis [41] and Bressan [42] for upper bounds on the mixing time. Our proposed

neighborhood sampling technique may also help scale variants of *PSRW*, such as the *shotgun sampler* [48].

Algorithm 1: *R-PSRW* Neighborhood Sampling

Input: *CIS* $x \in \mathcal{S}^{(k-1)}$.
Output: $k-1$-*CIS*, uniformly sampled from $\mathbf{N}_{\mathcal{G}^{(k-1)}}(x)$.
Let AP_x be the set of articulation points [47] of x.
while true **do**
  Sample an anchor-vertex $a \sim V(x)$ with probability $\propto \deg(a)$ and further sample $v \sim \mathbf{N}(a)$.
  if $v \notin V(x)$ **then**
    Let $z = G_{\text{IN}}\left(V(x) \cup \{v\}\right)$ be the *CIS* obtained by adding v to the original *CIS*.
    Let $\mathbf{N}_z(\cdot)$, be the neighborhood within z.
    if $\text{UNIF}(0,1) < \frac{1}{|\mathbf{N}_z(v)|}$ **then**
      Sample u from $V(x)$ uniformly at random.
      if $u = a$ *or* $u \in AP_x$ **then**
        if $G_{IN}\left(V(x) \cup \{v\} \backslash \{u\}\right)$ *is connected* **then**
          return $G_{IN}\left(V(x) \cup \{v\} \backslash \{u\}\right)$
      else
        return $G_{IN}\left(V(x) \cup \{v\} \backslash \{u\}\right)$

2.5 Related Work

Since my method samples k-*CIS*s by posing it as the problem of sampling k-depth leaves of the *FANMOD* [17] tree, and then uses regenerations in a random walk on this tree, I divide the related work into logical sub-sections for the reader's convenience.

2.5.1 Random walk-based *CIS* Sampling.

Random walk based *CIS* sampling techniques use a Markov chain, which samples *CIS*s with a bias (known up to a constant) to estimate function averages over *CIS*s contained in the input graph (Definition 2.2.2). Methods like *GUISE* [21], *RSS* [41], *PSRW* [18] define chains over the state-space of *CIS*s. Alternately, Waddling [49] and *IMPRG* [20] sample sequences of vertices via walks on the input graph G_{IN} and use specialized estimators to

sample up to 5-node *CIS*s. *RGPM* [50] and *Ripple* [50] use regenerations in the *PSRW* chain to parallelize it.

2.5.2 Other *CIS*-sampling methods.

Specialized methods exist that sample *CIS*s for $k \leq 5$ extremely efficiently [20], [24]. For larger k, however, their scalability is limited by the cost of computing the sampling probability, which usually requires an iteration over all permutations of the vertices that were sampled [16], [22], [23], [48]. *FANMOD* [17] improves this by enforcing an ordering over the sampled vertices, but suffers from a large error. Efficient methods that sample dense *CIS*s are unfortunately not extensible to all patterns [1], [2]. Similarly, methods for sparse graphs do not scale to arbitrary graphs Bera, Pashanasangi, and Seshadhri [51]. *Motivo* [19], [25], [26] is unique in that it uses a color-coding [52], [53] based index of non-induced trees to sample *CIS*s which requires memory exponential in k. While faster methods to sample *CIS*s exist in theory, they are yet to be evaluated in practice [42], [54]. An extensive survey of *CIS* sampling methods can be found in Ribeiro, Paredes, Silva, *et al.* [55]. Recently proposed methods such as *HOPS* [56], MaNIACS [57] and TieredSampling [58] are closely related, but focus on sampling non-induced sub-trees, frequent patterns (in terms of *MNI*), and in streaming settings, respectively.

2.5.3 Regenerations in Random Walks.

Multiple techniques exist which *split MCMC* sample paths into i.i.d. *tours* to compute *ergodic sums* in parallel and with confidence intervals [27], [46], [59]–[61]. Random walks on graphs, however are restricted to Regeneration point-based methods [35], [38]–[40], [50], [62]. Cooper, Radzik, and Siantos [39] and Massoulié, Le Merrer, Kermarrec, *et al.* [40] first used tours to estimate graph properties. Avrachenkov, Ribeiro, and Sreedharan [35], Savarese, Kakodkar, and Ribeiro [38], Teixeira, Cotta, Ribeiro, *et al.* [50], and Avrachenkov, Borkar, Kadavankandy, *et al.* [62] used state aggregations or *supernodes* to reduce running times. *Ripple* [37] further improves on the running times of these methods via stratification.

2.5.4 Tree Sampling.

The task of sampling leaves of large trees occurs in various problems, like in back-tracking algorithms [63] for combinatorial problems and in the task of sampling leaves uniformly from B+ trees [64]. One solution to this task is Knuth's algorithm [43] (discussed in detail in Section 2.3.1). Methods such as [65]–[68] attempt to reduce the variance of the Knuth estimate. To the best of the authors' knowledge, no method exists that does so via weighted random walks and *RWT*s.

2.6 Empirical Evaluation

I now evaluate my*RTS* and *R-PSRW* based estimators against the previously mentioned baselines that are capable of sampling large *CIS*s on real-world graphs. Specifically, I compare against the following methods which scale to $k > 5$ in large real-world graphs with reasonable accuracy. I defer the description of the graphs used in my evaluation and my experimental setup to Section A.1. The methods I evaluate against are:

- ***Motivo*:** *Motivo* [19] uses *Color Coding* to create an index of non-induced trees, which it then uses to sample *CIS*s. I observed in my experiments that the crucial index creation step in *Motivo* is very resource intensive. Once created however, sampling from this index is very fast and estimates converge fairly quickly. *Motivo* estimates, however, have a bias which depends on the initial coloring of the graph. A bias-free version can be constructed, where the graph is recolored multiple times, at the expense of having the index rebuilt every time. This unbiased version, however, is impractical since in many of my tasks, even a single indexing is unable to finish by the 3½-hour deadline.

- ***Ripple*:** In *Ripple* [37] the underlying Markov chain is a rejection sampled version of *PSRW* [18]. *Ripple* is distinct from my approach since it was designed to estimate sums and not function averages. Moreover, *Ripple* estimates suffer from a bias (which depends on k) due to the stratification it introduces to control running times [37, Theorem 2]. Even for small values of k, my*R-PSRW* implementation comprehensively outperforms

Ripple in the *CIS* pattern distribution task defined later. I defer a comprehensive set of *Ripple* results to Figure A.1 in Section A.1.

- ***Knuth-FM*:** I use Knuth's sampling procedure on the *FANMOD* [17] tree as defined in Lemma 1 in order to better control the number of samples and to measure the efficacy of using *RWT*s on the *FANMOD* tree as opposed to this much simpler (in terms of implementation) alternative.

- ***RecEdge+*:** I implement a highly efficient version of the *RecEdge* algorithm [16], using memoization and articulation points [47] to reduce the time to compute the sampling probability, which is $O(k!)$ in the worst case.

2.6.1 CIS pattern distribution estimation task

Since *CIS* function averages for size $k = 5$ can be computed exactly using *Escape* [14], I consider the 5-*CIS* pattern distribution estimation task of Wang, Lui, Ribeiro, *et al.* [18]. I define a set of non-isomorphic patterns $\mathcal{H}$, where each $H \in \mathcal{H}$ is a 5-*CIS*. The goal is to compute the distribution of all such 5-*CIS patterns* in the input graph. By Definition 2.2.2, my goal is to compute $\textsc{avg}(\mathcal{S}^{(5)}; f_{\text{PAT}})$, where $f_{\text{PAT}}(s) = (\mathbf{1}\{s \sim H\})_{H \in \mathcal{H}}$ and $\sim$ denotes graph isomorphism.

Figure 2.3 evaluates the L_2 norm between the estimates and true value of $\textsc{avg}(\mathcal{S}^{(5)}; f_{\text{PAT}})$, which is the distribution of 5-*CIS* patterns in the input graph. More precisely, for each method, I evaluate the estimates generated after $3^1/_2$ hours and compare the quartiles of the L2 error relative to the exactly computed value of $\textsc{avg}(\mathcal{S}^{(5)}; f_{\text{PAT}})$ over 10 runs. *RTS* consistently performs either similar to the best baseline or better than all of the other baselines. *R-PSRW* is comparable to *RTS* in some of the tasks, and has the lowest error for `DBLP`. *Motivo* exceeds memory limits and crashes for the larger graphs, `LiveJournal` and `Orkut`.

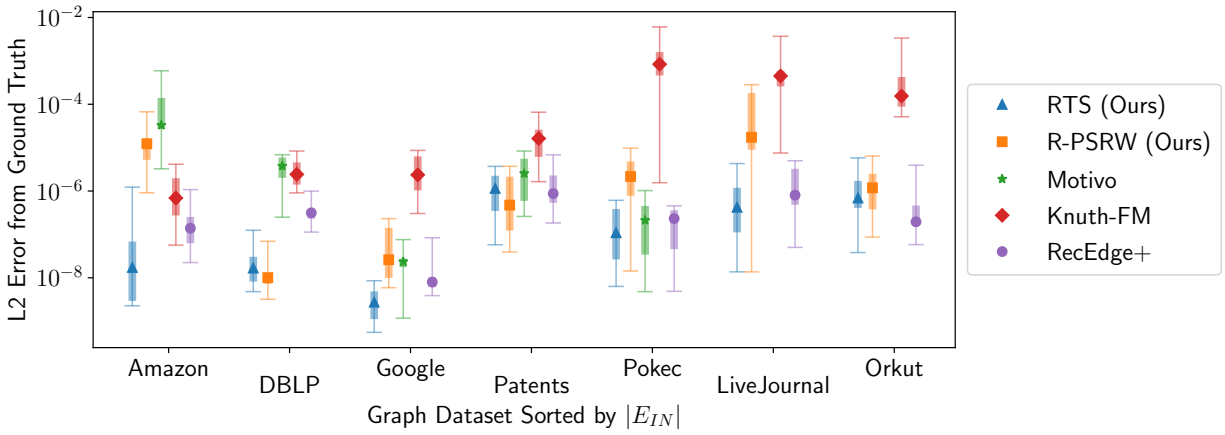


Figure 2.3. *RTS* Accuracy in the 5-*CIS* pattern distribution estimation task (Section 2.6.1) in various real-world graphs. Each box-plot represents quartiles of the L2 error (10 runs) relative to the true pattern distribution computed by *Escape* [14]. Estimates are computed after running each method for $3^1/_2$ hours (*Motivo* could not finish the indexing of `LiveJournal` and `Orkut` by the deadline). *RTS* estimates are consistently as accurate as or better than the other methods. *R-PSRW* matches *RTS* in some graphs, and is optimal for `DBLP`.

2.6.2 Average *CIS* edge-density estimation task

Since enumerating *CIS*s of size $k > 5$ is prohibitively expensive, the task defined in Section 2.6.1 cannot be used to evaluate my method for large k, since it requires one to compute the ground truth pattern distribution. Additionally, I cannot compute *CIS* pattern distributions for $k = 16$ (like I did for $k = 5$) since the number of patterns is large enough that storing the pattern distribution in memory is already prohibitively expensive. Previous work have used their own proposed method (running for a longer time) as ground-truth [19], [37], but I fear this evaluation would give an unfair advantage towards my method.

Therefore, I propose a task that summarizes information of all *CIS*s: I define the edge density function as $f_{\text{ED}}(s) = {}^{|E(s)|}/_{\binom{|V(s)|}{2}}$, where $E(s)$ and $V(s)$ are the set of edges and vertices of the *CIS* s. The task, then, is to compute $\textsc{avg}(\mathcal{S}^{(k)}; f_{\text{ED}})$, which according to Definition 2.2.2 is the average edge density of all *CIS*s in $\mathcal{S}^{(k)}$. Since this task does not favor specific patterns (e.g. sparse patterns or cliques) it is a good candidate task to evaluate the utility of *RTS*.

Since we no longer have access to ground-truth, I run experiments for $k \in \{8, 16, 20, 25\}$, and evaluate the convergence of each method using the following two quantities:

- **Initial Error:** I use the median over 10 runs of the estimates computed after running each algorithm for $3^1/_2$ hours as the ground truth and compute the normalized root mean square error (NRMSE) of the estimates computed after 1 hour of runtime. Note that the NRMSE of each method uses their own $3^1/_2$ hour estimates as "ground truth estimates". The lower the initial error, the better the method.

- **Variance Reduction:** I compute the ratio of the variance in the estimates computed by running each method for $3^1/_2$ hours to the variance of the 1 hour estimates. When comparing two methods, a smaller value indicates that the method converges faster to the true value compared to the other method.

In Figures 2.1a to 2.1d, each method is plotted as a point whose x-coordinate represents the initial error and y-coordinate, the variance reduction, described above. I use different colors for different input graphs, and marker sizes represent the size of the input graph.

Marker shapes represent various methods. Our *RTS* and *R-PSRW* estimates have better convergence properties (they tend to be situated in the bottom left hand corner) compared to the other tested baseline methods, with my*RTS* estimates showing superior convergence more often compared to my*R-PSRW* estimates. Note that *Motivo* does not run for $k \geq 8$ (even for $k = 8$ *Motivo* does not run for large input graphs).

In Figures 2.1e and 2.1f I plot the median of the $3^1/_2$ hour k-*CIS* edge-density estimates of *R-PSRW* against that of *RTS* for $k \in \{16, 25\}$, for all graphs where I obtained estimates. Despite having very different sampling strategies, the points lie on the diagonal, indicating that *R-PSRW* and *RTS* estimates converge to the same value for all tested graphs. I take this as strong evidence that both methods' estimates are correct, even though the computationally intractable ground-truth for $k \in \{16, 25\}$ is not known.

2.7 Summary

This chapter introduced two estimators of k-*CIS* function averages, *RTS* and *R-PSRW*, that push the state of the art from $k = 12$ to $k = 16$ for larger graphs and $k = 25$ for smaller graphs. By introducing regeneration and weighted random walks to the *FANMOD* tree, *RTS* is able to scale function estimation to an unprecedented range of k. *R-PSRW* proposes adding a novel randomized estimator to *PSRW* [18] that allows it to scale from $k = 8$ of the original algorithm to $k = 25$ over the same graphs. My empirical results show that *RTS* is the most consistently accurate method across all graphs, followed by *R-PSRW*, followed by the baselines (as distant 3rd places for larger k).

3. SEQUENTIAL STRATIFIED REGENERATIONS

3.1 Introduction

This chapter considers the following general task: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a simple graph, where $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ is the set of edges, and $\mathcal{E}$ contains at most a single edge between any pair of vertices and no self-loops. Our goal is to efficiently estimate the sum of a bounded function over all the edges of $\mathcal{G}$,

$$\mu(\mathcal{E}) = \sum_{(u,v)\in\mathcal{E}} f(u,v) \tag{3.1}$$

where $f\colon \mathcal{E} \to \mathbb{R}$, $f(\cdot) < B$ is a bounded function for some constant $B \in \mathbb{R}$ under the following query model from [69].

Assumption 1 (Query Model)**.** *Assume we are given arbitrary seed vertices and can query the neighborhood* $\mathbf{N}(u) \triangleq \{v \in \mathcal{V} : (u,v) \in \mathcal{E}\}$ *for any vertex* $u \in \mathcal{V}$ *such that accessing the entire graph* $\mathcal{G}$ *is prohibitively expensive.*

This setting arises naturally in the subgraph counting problem, which we study in Section 3.4. Simple Monte Carlo procedures are not useful because random vertex and edge queries are not directly available, and reservoir sampling would require iteration over all edges. Standard Markov chain Monte Carlo (*MCMC*) methods cannot estimate the quantity in Equation (3.1) and are limited to estimate $\mu(\mathcal{E})/|\mathcal{E}|$, because $|\mathcal{E}|$ in our task is unknown [70]. Generally, under Assumption 1, Equation (3.1) is estimated using specialized *MCMC* estimators that use a random-walk-like Markov chain that has a uniform distribution over the edges $\mathcal{E}$ as its equilibrium distribution. However, these estimators [69] are impractical in large graphs because their running time is $O(|\mathcal{E}|)$.

Traditional *MCMC* methods are limited by their reliance on the Markov chain on $\mathcal{G}$ reaching equilibrium or *burning in*. Because the rate of convergence to equilibrium depends on the spectral gap [44], a significant number of Markov chain steps is needed to *burn in* in order to produce accurate estimates of Equation (3.1), particularly in large graphs. Parallel approaches that divide the state space into disjoint "chunks", which are to be processed in parallel [71], [72], offer no respite because we cannot access the entire graph. In fact, $\mathcal{G}$

may not even have disconnected components (i.e., disjoint chunks) that can be parallelized. Therefore, traditional *MCMC* on $\mathcal{G}$ offers no meaningful parallelization opportunities and running times may be arbitrarily long.

Contributions. This work introduces *sequential stratified regeneration* (*Ripple*), a novel parallel *MCMC* technique that expands the application frontier of *MCMC* to large state-space graphs $\mathcal{G}$. *Ripple* stratifies the underlying Markov chain state space into ordered strata that need not be disjoint chunks, rather, they need to be connected. Markov chain regeneration [27] is then used to compute estimates in each stratum sequentially, using a recursive method, which improves regeneration frequencies and reduces variance. *Ripple* offers an unprecedented level of efficiency and parallelism for *MCMC* sampling on large state-space graphs while retaining the benefits of *MCMC*-based algorithms, such as low memory demand (polynomial w.r.t. output).

Surprisingly, the parallelism of *Ripple* comes from the regeneration rather than the stratification: the strata's job is to keep regeneration times short. I demonstrate that the estimates obtained by *Ripple* are consistent, among other theoretical guarantees. In addition, I empirically show the power of *Ripple* in a real-world application by specializing Equation (3.1) to subgraph counting in multi-million-node attributed graphs–to the best of my knowledge, a task at a scale that has been thought unreachable by any other *MCMC* method. My specific contributions to the subgraph counting problem include streaming-based optimizations coupled with a parallel reservoir sampling algorithm, novel efficiency improvements to the random walk on the *HON* [18] and a theoretical analysis of scalability in terms of running time and memory w.r.t. the subgraph size, verified empirically on large datasets.

3.2 Background and Prior Work

The *MCMC* random-walk-like Markov chain over the graph $\mathcal{G}$ is defined as:

Definition 3.2.1 (Random Walk on $\mathcal{G}$)**.** *Given a simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a simple random walk is a time-homogenous Markov chain $\mathbf{\Phi}$ with state space $\mathcal{V}$ and transition probability $p_{\mathbf{\Phi}}(u, v) = 1/\mathbf{d}(u)$, when $(u, v) \in \mathcal{E}$ and 0 otherwise, where $\mathbf{d}(u) = |\mathbf{N}(u)|$ is the degree of u in $\mathcal{G}$ and $\mathbf{N}(u) = \{v \colon (u, v) \in \mathcal{E}\}$ is the neighborhood of u.*

It is easy to check that the above random walk can be sampled under Assumption 1 and that on a connected graph, this walk samples edges uniformly at random in a steady state (check Section B.2.1 for details). My notation is summarized in Section B.1.

3.2.1 Regenerations in Discrete Markov Chains

The rate of convergence to stationarity of the random walk $\mathbf{\Phi}$ from Definition 3.2.1 depends on the spectral gap[1] [44]. As such, practitioners are encouraged to run a single, long sample path, which prevents them from splitting the task among multiple cores. Usually, because the spectral gap is unknown or loosely bounded, practitioners use various diagnostics to *eyeball* if the chain has mixed [73]. The variance of an estimate computed from a stationary chain [70] also depends on the spectral gap.

A solution to the above problems is to *split* [27] the Markov chain using regenerations. Discrete Markov chains regenerate every time they enter a fixed state, which is referred to as a regeneration point. This naturally yields the definition of a *random walk tour* (*RWT*).

Definition 3.2.2 (*RWT* over $\mathbf{\Phi}$)**.** *Given a time-homogenous Markov chain $\mathbf{\Phi}$ over finite state space $\mathcal{V}$ and a fixed point $x_0 \in \mathcal{V}$, an* RWT $\mathbf{X} = (X_\mathrm{i})_{\mathrm{i}=1}^{\xi}$ *is a sequence of states visited by $\mathbf{\Phi}$ between two consecutive visits to x_0, that is, $X_1 = x_0$ and $\xi = \min\{\mathrm{i} > 1 \colon X_{\mathrm{i}+1} = x_0\}$ is the first return time to x_0.*

Because of the strong Markov property [28, Chap-2, Thm-7.1], *RWT*s started at x_0 are i.i.d. and can be used to estimate $\mu(\mathcal{E})$ from Equation (3.1) when $|\mathcal{E}|$ is unknown [38]–[40], [50], [62], [69].

Lemma 3 (*RWT* Estimate)**.** *Given the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and the random walk $\mathbf{\Phi}$ from Definition 3.2.1, consider $f \colon \mathcal{E} \to \mathbb{R}$ bounded by B, and $\mathcal{T}$, a set of m* RWT*s started at $x_0 \in \mathcal{V}$ (Definition 3.2.2) sampled in a parallel z core environment assuming each core samples an equal number of tours. Then,*

$$\hat{\mu}_*(\mathcal{T}; f, \mathcal{G}) = \frac{\mathbf{d}(x_0)}{2m} \sum_{\mathbf{X} \in \mathcal{T}} \sum_{\mathrm{j}=1}^{|\mathbf{X}|} f(X_\mathrm{j}, X_{\mathrm{j}+1}) \,, \tag{3.2}$$

[1] The spectral gap is defined as $\delta = 1 - \max\{|\lambda_2|, |\lambda_{|\mathcal{V}|}|\}$, where λ_i denotes the i-th eigenvalue of the transition probability matrix of $\mathbf{\Phi}$.

is an unbiased and consistent estimator of $\mu(\mathcal{E}) = \sum_{(u,v)\in\mathcal{E}} f(u,v)$ *if* $\mathcal{G}$ *is connected, where each* X_{j} *refers to the* j*th state in the* RWT $\mathbf{X} \in \mathcal{T}$.

The expected running time for sampling m *tours is* $\mathrm{O}\left(m/z\frac{2|\mathcal{E}|}{\mathbf{d}(x_0)}\right)$, *and when* $\mathcal{G}$ *is non-bipartite, the variance of the estimate is bounded as*

$$\mathrm{Var}\left(\hat{\mu}_*(\mathcal{T})\right) \leq \frac{3B^2}{m}\frac{|\mathcal{E}|^2}{\delta(\mathbf{\Phi})}, \tag{3.3}$$

where $\delta(\mathbf{\Phi})$ *is the spectral gap as defined in the beginning of this section.*

The *RWT* Estimate can be considered a Las Vegas transformation of *MCMC*, which takes random time but yields unbiased estimates of objectives, such as Equation (3.1). The parallelism in the expected running time in Lemma 3 is directly due to the independence of *RWT*s. Moreover, confidence intervals for the *RWT* Estimate can be computed, because $\sqrt{m}\frac{\hat{\mu}_*(\mathcal{T})-\mu(\mathcal{E})}{\hat{\sigma}(\mathcal{T})}$ approaches the standard normal distribution for sufficiently large m, where $\hat{\sigma}(\mathcal{T})^2$ is the empirical variance of $\hat{\mu}_*(\mathbf{X})$, the *RWT* Estimate computed using an individual tour $\mathbf{X} \in \mathcal{T}$.

3.2.2 Improving the Regeneration Frequency

From Lemma 3, it is clear that increasing the degree of the regeneration point $\mathbf{d}(x_0)$ and spectral gap $\delta(\mathbf{\Phi})$ and decreasing $|\mathcal{E}|$ reduces the variance as well as the running time of the *RWT* Estimate. [69] showed that using the *supernode* in a contracted graph as a regeneration point achieves the above reductions.

Definition 3.2.3 (Contracted Graph)**.** *[69] Given a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *from Definition 3.2.1 and a set of vertices* $I \subset \mathcal{V}$, *a contracted graph is a multigraph* $\mathcal{G}_I$ *formed by collapsing* I *into a single node* ζ_I. *The vertex set of* $\mathcal{G}_I$ *is then given by* $\mathcal{V}\backslash I \cup \{\zeta_I\}$, *and its edge multiset is obtained by conditionally replacing each endpoint of each edge with* ζ_I *if it is a member of* I *and removing self-loops on* ζ_I. *We refer to the set* I *and the vertex* ζ *as the supernode.*

Contractions benefit *RWT*s because the supernode degree $\mathbf{d}_{\mathcal{G}_I}(\zeta_I)$ in $\mathcal{G}_I$ and the spectral gap $\delta(\mathbf{\Phi}_I)$ of the random walk on the contracted graph increase monotonically with $|I|$ [69]. Moreover, *RWT*s can be sampled on $\mathcal{G}_I$ without explicit construction, as we see next.

Remark 1. *Let the multi-set* $\mathbf{N}(\zeta_I) \triangleq \uplus_{u\in I}\mathbf{N}_{\mathcal{G}}(u)\backslash I$ *be the neighborhood of the supernode in* $\mathcal{G}_I$ *from Definition 3.2.3. Let* $\mathbf{\Phi}_I$ *be the simple random walk on* $\mathcal{G}_I$*. An* RWT $(X_{\mathrm{i}})_{\mathrm{i}=1}^{\xi}$ *on* $\mathbf{\Phi}_I$ *from* ζ_I *is sampled by setting* $X_1 = \zeta_I$*, sampling* X_2 *u.a.r. from* $\mathbf{N}(\zeta_I)$ *and subsequently sampling transitions from* $\mathbf{\Phi}$ *until the chain enters* I*, i.e.,* $\xi = \min\{\mathrm{i} > 1 \colon X_{\xi+1} \in I\}$.

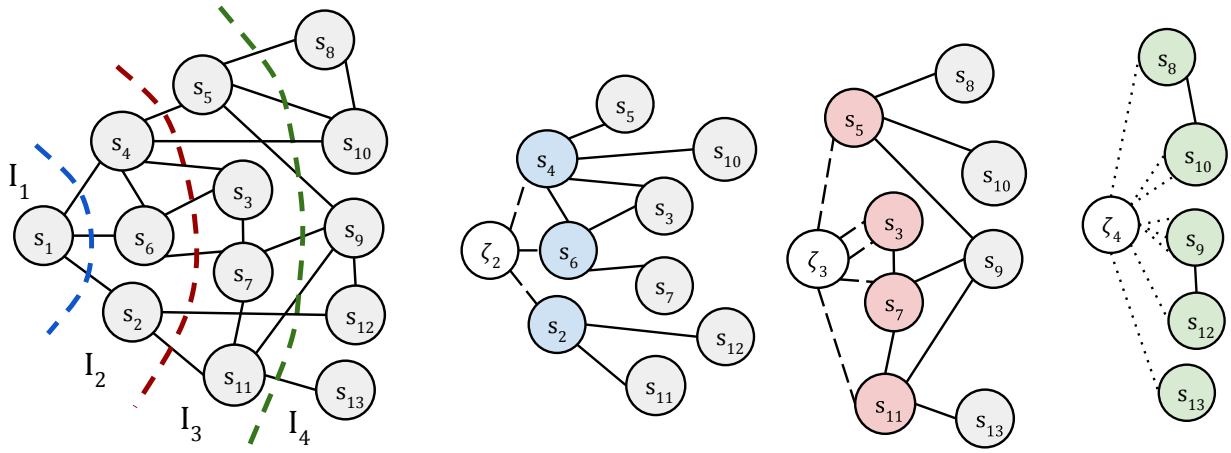
This construction naturally stratifies $\mathcal{E}$ and decomposes $\mu(\mathcal{E})$ as $\mu(\mathcal{E}_*)+\mu(\mathcal{E}\backslash\mathcal{E}_*)$, where we can exactly compute the $\mu(\mathcal{E}_*)$ and compute an *RWT* Estimate of $\mu(\mathcal{E}\backslash\mathcal{E}_*)$ on the contracted graph. However, to compute the supernode degree, $\mathbf{d}_{\mathcal{G}_I}(\zeta_I)$; furthermore, to sample from $\mathbf{N}(\zeta_I)$, we need to enumerate the set of the edges incident on I in $\mathcal{G}$ given by $\mathcal{E}_* \subset \mathcal{E}$. As such, a massive supernode I (which is crucial when $|\mathcal{E}|$ is large) makes enumerating $\mathcal{E}_*$ prohibitively expensive. We overcome these issues and gain additional control over regenerations by further stratifying $|\mathcal{E}|$.

3.3 Sequential Stratified Regenerations

Ripple controls regeneration times through a *sequential stratification* of the vertices and edges of $\mathcal{G}$ into ordered strata as illustrated in Figure 3.1, which allows us to control the regeneration frequency and the *RWT* Estimate variance. For each stratum, we then construct a graph in which the supernode is created by collapsing all prior strata, from which *RWT*s can be sampled. We use the *RWT*s from the previous strata to estimate the degree of and sample transitions from the supernode. The core idea is described in two steps: Section 3.3.1 details the stratification and conditions that it needs to satisfy and Section 3.3.2 describes the recursion. Finally, we see that the estimator bias converges to zero asymptotically in the number of tours. Particularly for subgraph counting, I show that *Ripple*'s time complexity is independent of the (higher-order) graph size ($|\mathcal{E}|$) and only depends polynomially on the diameter and maximum degree of the *input* graph and the subgraph size (Section 3.4).

3.3.1 Sequential Stratification

Consider the following vertex and edge stratification procedure.



(a) Stratification $\mathcal{V} \equiv \mathcal{I}_{1:4}$ (b) $\mathcal{G}_2$ (c) $\mathcal{G}_3$ (d) $\mathcal{G}_4$

Figure 3.1. Figure 3.1a shows a simple graph $\mathcal{G}$ that is stratified into four strata $\{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4\}$. Figures 3.1b to 3.1d show the second, third and fourth graph strata constructed by Definition 3.3.2. In the multi-graph $\mathcal{G}_2$ (Figure 3.1b), vertices in $\mathcal{I}_1$ are collapsed into ζ_2 and only edges incident on $\mathcal{I}_2$ are preserved. The edge set therefore contains $\mathcal{J}_2$ and the edges between ζ_2 and $\mathcal{I}_2$. Consequently, self-loops on ζ_2 and edges between $\mathcal{I}_{3:4}$ are absent. Figures 3.1c and 3.1d follow suit. In each stratum $\mathcal{G}_r$, *RWT*s from ζ_r are started by sampling u.a.r. from the dotted edges and estimates are computed over the solid edges.

Definition 3.3.1 (Sequential Stratification). *Given $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from Definition 3.2.1, a function $\rho\colon \mathcal{V} \to \{1, \ldots, R\}$ induces the stratification $(\mathcal{I}_r, \mathcal{J}_r)_{r=1}^R$ if $s \in \mathcal{I}_{\rho(s)}$, for each $s \in \mathcal{V}$, and $(u, v) \in \mathcal{J}_{\min(\rho(u),\rho(v))}$, for each $(u, v) \in \mathcal{E}$.*

Note that these strata are pairwise disjoint and their union is the set of vertices and edges of the graph. Next, I describe the contracted graph over which *RWT*s are to be sampled in each stratum.

Definition 3.3.2 (r-th Graph Stratum). *Let $\mathcal{A}_{\text{i:j}} \triangleq \cup_{x=\text{i}}^{\text{j}} \mathcal{A}_x$ be defined for any ordered tuple of sets. Let $(\mathcal{I}_r, \mathcal{J}_r)_{r=1}^R$ be the stratification induced by ρ from Definition 3.3.1 on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The r-th graph stratum $\mathcal{G}_r = (\mathcal{V}_r, \mathcal{E}_r)$, $r > 1$, is obtained by removing all edges not incident on $\mathcal{I}_r$ and vertices that do not neighbor vertices in $\mathcal{I}_r$ and subsequently contracting $\mathcal{I}_{1:r-1}$ into ζ_r according to Definition 3.2.3. Further, let $\mathbf{\Phi}_r$ denote the simple random walk on $\mathcal{G}_r$.*

It can be shown that the vertex set $\mathcal{V}_r$ contains the r-th stratum $\mathcal{I}_r$, the r-th supernode ζ_r, obtained by collapsing $\mathcal{I}_{1:r-1}$, and vertices from subsequent strata neighboring $\mathcal{I}_r$, $\cup_{u\in\mathcal{I}_r}\mathbf{N}(u)\cap\mathcal{I}_{r+1:R}$. The edge multiset $\mathcal{E}_r$ is the union of $\mathcal{J}_r$ and edges that connect ζ_r to vertices in $\mathcal{I}_r$ resulting from the graph contraction. A detailed example is shown in Figure 3.1. Note that when $R = 2$, *Ripple* reduces to the estimator from [69].

Ergodicity-Preserving Stratification. Because the *RWT* Estimate is consistent only if the underlying graph is connected according to Lemma 3, we have the following definition:

Definition 3.3.3 (Ergodicity-Preserving Stratification (*EPS*)). *The stratification due to ρ from Definition 3.3.1 is an Ergodicity-Preserving Stratification if each graph stratum from Definition 3.3.2 is connected, i.e., $\mathbf{\Phi}_r$, $r > 1$, is irreducible.*

I propose necessary and sufficient conditions on ρ that yield an *EPS*.

Proposition 3.3.1. *ρ yields an* EPS *if the following three conditions are satisfied:*

(a) for at least one vertex in each connected component of $\mathcal{G}$, ρ evaluates to 1;

(b) for each $u\colon \rho(u) = r$, there exists $v \in \mathbf{N}(u)$ such that $\rho(v) \leq r$; and

(c) there exists $(u_0, v_0) \in \mathcal{E}$ such that $\rho(u_0) = r$ and $\rho(v_0) < r$.

Although the optimal stratification would depend on $\mathcal{G}$ and the quantity being estimated, an ideal stratification would yield graph strata wherein the supernode degree and connectivity are maximized (Lemma 3) while minimizing the number of strata (because of the bias propagation described in Theorem 3.3.2). ρ needs to be efficient as well because we will see that it is evaluated at each step of the random walk and the *Ripple* estimators from Definitions 3.3.5 and 3.3.6 heavily depend on it. In Proposition B.5.1 I show that return times to the supernode are inversely proportional to the fraction of vertices in $\mathcal{I}_r$ connected to $\mathcal{I}_{1:r-1}$.

3.3.2 Recursive Regenerations

Assume for the moment that in each stratum, $r = 2, \ldots, R$, we know the degree of the supernode $\mathbf{d}(\zeta_r)$ and can sample directly from $p_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$, which is the transition probability out of ζ_r in the graph stratum $\mathcal{G}_r$. We could then sample *RWT*s $\mathcal{T}_r$ and compute stratum-wise *RWT* Estimates, which when combined as $\mu(\mathcal{J}_1) + \sum_{r=2}^{R} \hat{\mu}_r(\mathcal{T}_r)$ provide an unbiased estimate of $\mu(\mathcal{E})$ as a direct consequence of Lemma 3 and the linearity of expectations. Unfortunately, the impracticality of this assumption, especially under Assumption 1 (when $R > 2$), necessitates the following relaxation.

Definition 3.3.4 (Supernode Estimates, $\widehat{\mathbf{d}}(\zeta_r)$ and $\widehat{p}_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$)**.** *Given an* EPS *of $\mathcal{G}$ (Definition 3.3.3), the supernode estimates in the r-th graph stratum $\mathcal{G}_r$ consist of the estimate of the degree $\widehat{\mathbf{d}}(\zeta_r)$ and a sample from some approximate transition probability out of the supernode $\widehat{p}_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$. Let $\widehat{\mathbf{\Phi}}_r$ be the random walk on $\mathcal{G}_r$, where transitions are sampled according to $\mathbf{\Phi}_r$ everywhere except ζ_r, where they are sampled from $\widehat{p}_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$.*

Although $\widehat{\mathbf{\Phi}}_r$ may not be reversible, *RWT*s on $\widehat{\mathbf{\Phi}}_r$ retain pairwise independence and the benefits stated after Lemma 3. We leverage this fact in the following recursive solution that computes *supernode estimates* in the current stratum using supernode estimates and tours sampled in the previous strata.

Definition 3.3.5 (*Ripple*'s Recurrence Relation)**.** *Given a graph $\mathcal{G}$ stratified according to ρ (Definition 3.3.3) and some stratum r, $1 < r \leq R$, assume access to the result of previous recursive steps, i.e., the set of m_q* RWT*s ($\mathcal{T}_q^\dagger$), supernode degree estimates $\widehat{\mathbf{d}}(\zeta_q)$ and estimated*

transition probabilities out of the supernode $\widehat{p}_{\mathbf{\Phi}_q}(\zeta_q, \cdot)$ *(Definition 3.3.4) for all* $2 \leq q < r$. *The estimate of the number of edges between* $\mathcal{I}_q$ *and* $\mathcal{I}_r$ *is given by*

$$\widehat{\beta}_{q,r} = \frac{\widehat{\mathbf{d}}(\zeta_q)}{|\mathcal{T}_q^\dagger|} \sum_{\mathbf{X} \in \mathcal{T}_q^\dagger} \sum_{\mathrm{j}=2}^{|\mathbf{X}|} \mathbf{1}\{\rho(X_\mathrm{j}) = r\}\,, \tag{3.4}$$

where X_j *is the* j*-th state visited in tour* $\mathbf{X}$, *and by convention,* $\widehat{\beta}_{1,r} = |\mathcal{E} \cap \mathcal{I}_1 \times \mathcal{I}_r|$ *is exactly computed. The* r*-th supernode degree is then estimated as*

$$\widehat{\mathbf{d}}(\zeta_r) = \sum_{q=1}^{r-1} \widehat{\beta}_{q,r}\,. \tag{3.5}$$

Transitions from $\widehat{p}_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$ *are sampled by sampling* $q \in \{1, \ldots, r-1\}$ *with probability* $\widehat{\beta}_{q,r}$ *and then sampling u.a.r. from* $\widehat{\mathbf{U}}_{q,r}$, *which is defined as*

$$\widehat{\mathbf{U}}_{q,r} = \uplus_{\mathbf{X} \in \mathcal{T}_q^\dagger} \uplus_{\mathrm{j}=2}^{|\mathbf{X}|} \{X_\mathrm{j} \colon \rho(X_\mathrm{j}) = r\}\ , \text{when}\, q > 1\,, \tag{3.6}$$

and as $\uplus_{u \in \mathcal{I}_1} \mathbf{N}(u) \cap \mathcal{I}_r$ *by convention when* $q = 1$, *where* $\uplus$ *is the multi-set union.* $\widehat{\mathbf{U}}_{q,r}$, $q > 1$, *is thus the multi-set of all states in* $\mathcal{I}_r$ *visited by* RWT*s on* $\widehat{\mathbf{\Phi}}_q$. *An* RWT *so started stops when it reaches some state* X', *where* $\rho(X') = r$.

Proposition B.3.1 (Section B.3) contains additional details for sampling *RWT*s on $\mathbf{\Phi}_r$. The above recursion therefore allows us to estimate supernode degrees and sample *RWT*s to compute an estimate of $\mu(\mathcal{E})$ from Equation (3.1) as follows:

Definition 3.3.6 (*Ripple*'s μ Estimator)**.** *Given the supernode degree estimates* $\widehat{\mathbf{d}}(\zeta_r)$ *and* RWT*s* $\mathcal{T}_r^\dagger$ *sampled in each graph stratum from Definition 3.3.5 and the edge strata* $\mathcal{J}_r$, $2 \leq r \leq R$ *based on an* EPS *of* $\mathcal{G}$ *from Definition 3.3.3, the* Ripple *estimate is defined as*

$$\hat{\mu}_{Ripple} = \mu(\mathcal{J}_1) + \sum_{r=2}^{R} \hat{\mu}\left(\mathcal{T}_{2:r}^\dagger; f\right)\,, \tag{3.7}$$

$$\text{where, } \hat{\mu}\left(\mathcal{T}_{2:r}^\dagger; f\right) = \frac{\widehat{\mathbf{d}}(\zeta_r)}{2|\mathcal{T}_r^\dagger|} \sum_{\mathbf{X} \in \mathcal{T}_r^\dagger} \sum_{\mathrm{j}=2}^{|\mathbf{X}|-1} f(X_\mathrm{j}, X_{\mathrm{j}+1})\,, \tag{3.8}$$

and X_{j} *is the* j*th state visited by the* RWT $\mathbf{X} \in \mathcal{T}_r^\dagger$*. The dependence of* $\mathcal{T}_r^\dagger$ *and* $\widehat{\mathbf{d}}(\zeta_r)$ *on* $\mathcal{T}_{2:r-1}^\dagger$ *is suppressed for brevity.*

This estimate of $\mu(\mathcal{E})$ is unbiased when the number of tours is infinite.

Theorem 3.3.1. *The* Ripple *estimate from Definition 3.3.6 is a consistent estimator of* $\mu(\mathcal{E})$ *(asymptotically unbiased in the number of tours), that is,*

$$\lim_{|\mathcal{T}_2^\dagger|\to\infty} \dots \lim_{|\mathcal{T}_R^\dagger|\to\infty} \mu(\mathcal{J}_1) + \sum_{r=2}^{R} \hat{\mu}\left(\mathcal{T}_{2:r}^\dagger; f\right) \overset{a.s.}{=} \mu(\mathcal{E})\,.$$

In the finite regime, however, there exists a bias in each stratum that depends on the estimation bias in the previous strata, which we quantify as follows:

Theorem 3.3.2. *Given the random walk* $\mathbf{\Phi}_r$ *on the* EPS*-stratum* $\mathcal{G}_r$ *from Definitions 3.3.2 and 3.3.3, the estimates of the degree and transition probability at the supernode* $\widehat{\mathbf{d}}(\zeta_r)$ *and* $\widehat{p}_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$ *from Definition 3.3.4, and assuming aperiodic* $\mathbf{\Phi}_r$*, the bias of the* Ripple *estimate in the* r*th stratum from Equation (3.8) is given by*

$$\left|\mathbb{E}\left[\hat{\mu}\left(\mathcal{T}_{2:r}^\dagger; f\right)\middle|\mathcal{T}_{2:r-1}^\dagger\right] - \mu(\mathcal{J}_r)\right| \leq (\lambda_r\nu_r + |1-\lambda_r|)\,\frac{\sqrt{3}B|\mathcal{E}_r|}{\sqrt{\delta_r}}\,,$$

where δ_r *is the spectral gap of* $\mathbf{\Phi}_r$*,* B *is the upper bound of* f*,* $\nu_r = \|\widehat{p}_{\mathbf{\Phi}_r}(\zeta_r, \cdot) - p_{\mathbf{\Phi}_r}(\zeta_r, \cdot)\|_2$ *is the* L^2 *distance between transition probabilities out of* ζ_r *[44](Definition B.3.2) and* $\lambda_r = \widehat{\mathbf{d}}(\zeta_r)/\mathbf{d}(\zeta_r)$*.*

Therefore, the bias in each stratum affects the bias in subsequent strata. Consequently, we control the empirical variance in each stratum by increasing the number of tours sampled (I detail this for subgraph counting in Section 3.4).

3.4 Applying *Ripple* to Count Subgraphs

We now focus on a concrete implementation of *Ripple* to count subgraphs on a given simple input graph $G = (V, E, L)$ with vertices V, edges E, and attribute function L, which is assumed to be finite and undirected. In general, a subgraph induced by any $V' \subset V$ on

G is given by $G(V') = (V', E \cap (V' \times V'), L)$. However, in this chapter, I am interested in subgraphs $G(V')$ that are connected and where $|V'| = k$, referred to as a connected, induced subgraph (*CIS*) of size k or k-*CIS*. As such, the task is defined as

Definition 3.4.1 (Subgraph Count)**.** *Let $\mathcal{V}^{(k)}$ be the set of all k-*CIS*s of graph G, let $\sim$ denote the graph isomorphism equivalence relation (or any equivalence relation), and let $\mathcal{H}$ be an arbitrary set of pairwise nonequivalent k-*CIS*s. The subgraph count is defined as the $|\mathcal{H}|$-dimensional vector $\mathcal{C}^{(k)} = (\mathcal{C}_H^{(k)})_{H \in \mathcal{H}}$, where $\mathcal{C}_H^{(k)} = \sum_{s \in \mathcal{V}^{(k)}} \mathbf{1}\{s \sim H\}$, and $\mathbf{1}\{\cdot\}$ is the indicator function.*

Therefore, $\mathcal{C}^{(k)}$ contains the count of subgraphs in $\mathcal{V}^{(k)}$ equivalent to each subgraph in $\mathcal{H}$. I suppress the dependence of $\mathcal{C}^{(k)}$ on $\mathcal{H}$ for simplicity.

Subgraph counting is challenging when $k > 3$ in real-world input graphs because $\mathcal{V}^{(k)}$ is not tractably enumerable and naively sampling k vertices to obtain *CIS*s is challenging because ${}^{|\mathcal{V}^{(k)}|}/_{|V|^k} \to 0$ (as evidenced by Table 3.1). Next, I address this issue by reducing the subgraph counting problem to an edge sum (Equation (3.1)) over a higher-order graph that only provides neighborhood query access for large-real-world input graphs. I also propose a stratification strategy compatible with the access model and introduce novel solutions to improve speed and memory requirements. I defer the straightforward aspects to Section B.5, wherein I summarize the entire algorithm (Algorithm 3).

3.4.1 *MCMC* on the Subgraph Space

[18] proposed a network over subgraphs called the *HON*, which exposes neighborhood query access from Assumption 1 and is therefore amenable to *MCMC* solutions (which I optimize in Algorithm 2).

Definition 3.4.2 (Higher-Order Network (k-*HON*) [18])**.** *The higher-order network or* HON *$\mathcal{G}^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)})$ is a graph whose vertices are the set of all k-*CIS *contained in the input graph G, and (u, v) form an edge in $\mathcal{E}^{(k)}$ if they share all but $k-1$ vertices, that is, $|V(u) \cap V(v)| = k - 1$.*

In the $k-1$-*HON*, the subgraph induced by an edge $(u, v) \in \mathcal{E}^{(k-1)}$, i.e., $G\left(V(u) \cup V(v)\right)$, is a k-*CIS*. Thus, the subgraph counts from Definition 3.4.1 can then be expressed as an edge sum over $\mathcal{G} \equiv \mathcal{G}^{(k-1)}$ as

$$\mathcal{C}^{(k)} = \mu(\mathcal{E}^{(k-1)}) = \sum_{(u,v)\in\mathcal{E}^{(k-1)}} \left(\frac{\mathbf{1}\{G\left(V(u) \cup V(v)\right) \sim H\}}{\gamma(u, v)} \right)_{H\in\mathcal{H}}, \tag{3.9}$$

where $\gamma(u, v) = |\{(\ddot{u}, \ddot{v}) \in \mathcal{E} \colon V(u) \cup V(v) \equiv V(\ddot{u}) \cup V(\ddot{v})\}|$ is the number of edges that represent the same subgraph as (u, v). The set of edges sampled by a random walk on $\mathcal{G}^{(k-1)}$ is called the *pairwise subgraph random walk* (*PSRW*). Having reduced the subgraph counting task to Equation (3.1), we proceed with implementing *Ripple*.

3.4.2 Ergodicity-Preserving Stratification for Subgraph Counting

Toward using *Ripple*, I propose an Ergodicity-Preserving Stratification of $\mathcal{G}$ via the stratification function ρ.

Proposition 3.4.1 (*EPS* for subgraphs)**.** *Consider the set of* n_1 seed *subgraphs* $\mathcal{I}_1$ *whose vertex sets in* G *are pairwise non-intersecting. Let* $V(\mathcal{I}_1) \triangleq \cup_{\ddot{s}\in\mathcal{I}_1} V(\ddot{s})$ *be the set of all vertices in* G *forming subgraphs in* $\mathcal{I}_1$*. Let* $\textsc{dist}(u)$ *be the shortest path distance from* $u \in V$ *to any vertex in* $V(\mathcal{I}_1)$*. Define* ρ *as*

$$\rho(s) = 1 + \sum_{u\in V(s)} \left(\textsc{dist}(u) + \mathbf{1}\{u \in V(\mathcal{I}_1) \backslash V^*)\}\right),$$

where V^* *is the largest connected subset of* $V(s)$ *such that* $V^* \subseteq V(\ddot{s})$ *for some seed vertex* $\ddot{s} \in \mathcal{I}_1$ *with ties broken arbitrarily. If* $\mathcal{I}_1$ *contains a subgraph from each connected component of* G*, the stratification from Definition 3.3.1 generated using* ρ *is an Ergodicity-Preserving Stratification (Definition 3.3.3).*

$\textsc{dist}$ can be precomputed for all $u \in V$ using a single BFS in $\mathrm{O}(|V| + |E|)$, and ρ can be computed in $\mathrm{O}(k)$. Although R is unknown a priori, it is upper bounded as $(k-1) \cdot D_G$, where D_G is the diameter of G and the *Ripple* estimator simply ignores *empty* strata, i.e.,

strata in which the estimated degree of the supernode $\mathbf{d}(\zeta_r) = 0$. To control bias, we aim to reduce $\max_{u \in V} \textsc{dist}(u)$ by recruiting seed subgraphs in $\mathcal{I}_1$, which are far apart in G.

3.4.3 Miscellaneous Optimizations

Controlling Memory through Streaming. In each pair of strata $r < t$, Definition 3.3.5 uses tours $\mathcal{T}_r^\dagger$ to compute $\hat{\mu}(\mathcal{T}_{2:r}^\dagger; f)$, $\widehat{\beta}_{r,t}$ and $\widehat{\mathbf{U}}_{r,t}$, which are, respectively, the estimates of $\mu(\mathcal{J}_r)$ and the size of and sample from the set of vertices in $\mathcal{I}_t$ connected to $\mathcal{I}_r$. Although $\hat{\mu}(\mathcal{T}_{2:r}^\dagger; f)$ and $\widehat{\beta}_{r,t}$ can be computed as running sums, storing $\widehat{\mathbf{U}}_{r,t}$ requires memory on the order of the sum of all tour lengths, which is random. My solution is to use *Algorithm R* [74], to sample a fixed-size ($\textsc{m}$) sample without replacement from all the tours in $\mathcal{T}_r^\dagger$ (See Section B.5.1). Note that although the hyperparameter $\textsc{m}$ controls memory, it may introduce bias when the number of tours $|\mathcal{T}_r^\dagger| > \textsc{m}$ due to (possible) oversampling, which we observe in Figure B.3 (Section B.6).

Speeding up Subgraph Random Walks. To sample a random walk in the *HON*, naively sampling u.a.r. from the neighborhood of a $k-1$-*CIS* requires $\mathrm{O}(k^4 \Delta_G)$ operations, where Δ_G is the maximum degree in the input graph (see Section B.5.2). In Algorithm 2, I propose a rejection sampling algorithm that does so efficiently using *articulation points* [47].

Algorithm 2: Efficient Neighborhood Sampling in $\mathcal{G}^{(k-1)}$

Input: $k-1$-*CIS* s, Graph G
Output: $x \sim \textsc{unif}(\mathbf{N}_{\mathcal{G}^{(k-1)}}(s))$

Let $\deg_s = \sum_{u \in V(s)} \mathbf{d}(u)$ and $\mathcal{A}_s$ be the *articulating points* of s
while *True* **do**
  Sample u from $V(s)$ w.p. $\propto \deg_s - \mathbf{d}(u)$; `// u is the vertex to remove`
  Sample a from $V(s) \backslash \{u\}$ w.p. $\propto \mathbf{d}(a)$
  Sample $v \sim \textsc{unif}(\mathbf{N}(a))$; `// v is the vertex to add`
  $\textsc{bias} = |N(v) \cap V(s) \backslash \{u\}|$; `// v's sampling bias`
  if $\textsc{unif}(0,1) \leq 1/\textsc{bias}$ **then**
    $x = G\left(V(s) \cup \{v\} \backslash \{u\}\right)$
    if $u \neq v$ **and** *(*$u \notin \mathcal{A}_s$ **or** x *is connected)* **then**
      return x ; `// Connectivity Check`

Proposition 3.4.2. *Given a subgraph $s \in \mathcal{V}^{(k-1)}$, Algorithm 2 samples u.a.r. from $\mathbf{N}_{\mathcal{G}^{(k-1)}}(s)$ in $\mathrm{O}(k^2 \frac{\Delta_s + k|\mathcal{A}_s|}{k-|\mathcal{A}_s|})$ expected time, where $\Delta_s \triangleq \max_{u \in V(s)} \mathbf{d}_G(u)$ is the maximum degree of vertices in s, and $\mathcal{A}_s$ contains articulation points of s.*

Therefore, the running time of Algorithm 2 is $\in \mathrm{O}(k\Delta_s + k^2)$ when s is dense ($|\mathcal{A}_s| \approx 0$) and increases to $\mathrm{O}(k^2\Delta_s + k^4)$ for sparse subgraphs, which is faster than the naive algorithm.

From Error Bounds to Tour Counts. *Ripple* auto-decides the number of *RWT*s required in each stratum based on an approximate error bound ϵ provided as input such that the number of tours $\to \infty$ as $\epsilon \to 0$, and the *Ripple* estimate converges to the ground truth (Theorem 3.3.1). Specifically, *RWT*s are sampled until we satisfy

$$\hat{\sigma}(\mathcal{T}_r^\dagger; f_1)/\sqrt{|\mathcal{T}_r^\dagger|} \leq \epsilon\, \hat{\mu}(\mathcal{T}_r^\dagger; f_1)\,, \tag{3.10}$$

where $\hat{\mu}(\mathcal{T}_r^\dagger; f_1)$ is the *Ripple* estimate from Equation (3.8) of the number of edges in the r-th graph stratum $\mathcal{G}_r$ (i.e., $f_1(\cdot) = 1$), and $\hat{\sigma}^2(\mathcal{T}_r^\dagger; f_1) = \widehat{\mathrm{Var}}_{\mathbf{X} \sim \mathcal{T}_r^\dagger}(\hat{\mu}(\mathbf{X}; f_1))$ is the former's empirical variance over tours.

Performance Guarantees. Ignoring the complexity of loading the input graph into memory, I show that for subgraph counting, the memory and time requirements of *Ripple* are a polynomial in k. In Section B.5, I state and prove a detailed version in which the complexity also depends polynomially on the diameter and maximum degree of G and is invariant to $|V|$ and $|E|$.

Proposition 3.4.3. *Assuming a constant m* RWT*s sampled per stratum and ignoring graph loading, the* Ripple *estimator for k-*CIS *counts detailed in Section B.5-Algorithm 3 has total memory and time complexity in $\widehat{\mathrm{O}}(k^3 + |\mathcal{H}|)$ and $\widehat{\mathrm{O}}(k^7 + |\mathcal{H}|)$, respectively, when all factors other than k and $|\mathcal{H}|$ are ignored.*

More details for subgraph counting with *Ripple* are provided in Section B.5.

3.5 Experiments and Results

I now evaluate the *Ripple* estimator for k-node subgraph (*k-CIS*) counts on large-real-world networks. I show that *Ripple* outperforms the state-of-the-art method in terms of time

Table 3.1. The graphs that I used along with their diameter D_G, maximum degree Δ_G and the estimated orders of magnitude of k-*CIS* counts, $|\mathcal{V}^{(k)}|$.

Graph	**|V|**	**|E|**	$\mathbf{D_G}$	$\mathbf{\Delta_G}$	**Magnitude of Est. # of *CIS*s** $\|\mathcal{V}^{(6)}\|$	$\|\mathcal{V}^{(8)}\|$	$\|\mathcal{V}^{(10)}\|$	$\|\mathcal{V}^{(12)}\|$
Amazon	334,863	925,872	44	549	10^{11}	10^{15}	10^{19}	10^{22}
DBLP	317,080	1,049,866	21	343	10^{12}	10^{16}	10^{19}	10^{23}
Cit-Pat.	3,774,768	16,518,948	22	793	10^{14}	10^{18}	10^{22}	10^{26}
Pokec	1,632,803	30,622,564	11	14,854	10^{18}	10^{25}	10^{32}	10^{38}
LiveJ.	3,997,962	34,681,189	17	14,815	10^{19}	10^{25}	10^{32}	10^{38}
Orkut	3,072,441	117,185,083	9	33,313	10^{21}	10^{28}	10^{35}	10^{43}

and space and that *Ripple* converges to the ground truth for various pattern sizes as hyperparameters are varied. Additional experiments that evaluate the parallelism, etc., are deferred to Section B.6. My code is available at `https://github.com/PurdueMINDS/Ripple`.

- *Execution environment.* My experiments were performed on a dual Intel Xeon Gold 6254 CPU with 72 virtual cores (total) at 3.10 GHz and 392 GB of RAM. In addition, this machine is equipped with a fast SSD NVMe PCIex4 with 800 GB of free space available.

- *Baselines.* I use Motivo [19], a fast and parallel C++ system for subgraph counting, as the baseline because it is the only method capable of counting large patterns (k>6), to the best of my knowledge. Additionally, notice that existing *MCMC* methods for subgraph counting, such as *IMPRG* [20] and *RGPM* [50], cannot count beyond $k = 5$ in practice.

- *Datasets.* I use large networks from SNAP [75], representing diverse domains, which have been used to evaluate many subgraph counting algorithms [19], [25]. Table 3.1 presents the basic features of these datasets, including the order of magnitude of the *Ripple* estimates of the subgraph counts $|\mathcal{V}^{(k)}|$, $k = 6, 8, 10, 12$.

- *Hyper-parameters* $\mathcal{I}_1$, M *and* ϵ. Finally, I evaluate the trade-off between accuracy and resource consumption by varying the aforementioned hyperparameters, detailed in Sections 3.4.2 and 3.4.3. (M is evaluated in Section B.6.)

3.5.1 Scalability Assessment

I start by assessing the scalability of the methods when estimating k-*CIS* counts for $k \geq 6$. To the best of my knowledge, Motivo is the only existing method capable of estimating these patterns. Motivo has two phases: a build-up phase, which constructs an index table in the disk, and a sampling phase that queries this table. I only measure the time taken by the build-up phase and the out-of-core (disk) usage because this is a bottleneck for Motivo. As such, I report the best-case scenario for Motivo, and the reported values are lower bounds for the actual time and space requirement. For *Ripple*, I report the total time and the RAM usage as the space cost because my method works purely in memory. Both methods were executed using all threads available.

In Tables 3.2 and 3.3, I compare the running time and space usage of *Ripple* and Motivo. I also report their rate of increase in terms of the subgraph size k in columns $\text{Time}^{(k)}/\text{Time}^{(k-2)}$ and $\text{Space}^{(k)}/\text{Space}^{(k-2)}$. I fix $\epsilon = 0.003$, $|\mathcal{I}_1| = 10^4$ and $\text{M} = 10^7$ based on the analysis in Section 3.5.2 and Section B.6. For Motivo, I follow the authors' suggestions. In Section B.6-Table B.2, I report the dispersion $\frac{\max - \min}{\text{mean}}$ of the *Ripple* estimates generated in the measured runs to ensure that the results are not arbitrary.

Running time Scalability (Table 3.2). Although *Motivo* outperforms *Ripple* for $k = 6, 8$, it does not scale well for $k = 10, 12$, where the execution terminates because of insufficient storage space. Particularly, for DBLP, Motivo required approximately 10 minutes to process 10-*CIS* but almost 9 hours for 12-*CIS*, a growth rate of 58×. On the other hand, *Ripple* not only succeeded in **all** configurations in less than 4 hours on average but also exhibited a smoother growth in running time, with the largest increase ratio being 2.7×, observed for DBLP and LiveJournal when k went from 8 to 10. Furthermore, $\text{Time}^{(k)}/\text{Time}^{(k-2)} < (\text{k}/(\text{k-2}))^7$ in all cases according to Proposition B.5.1.

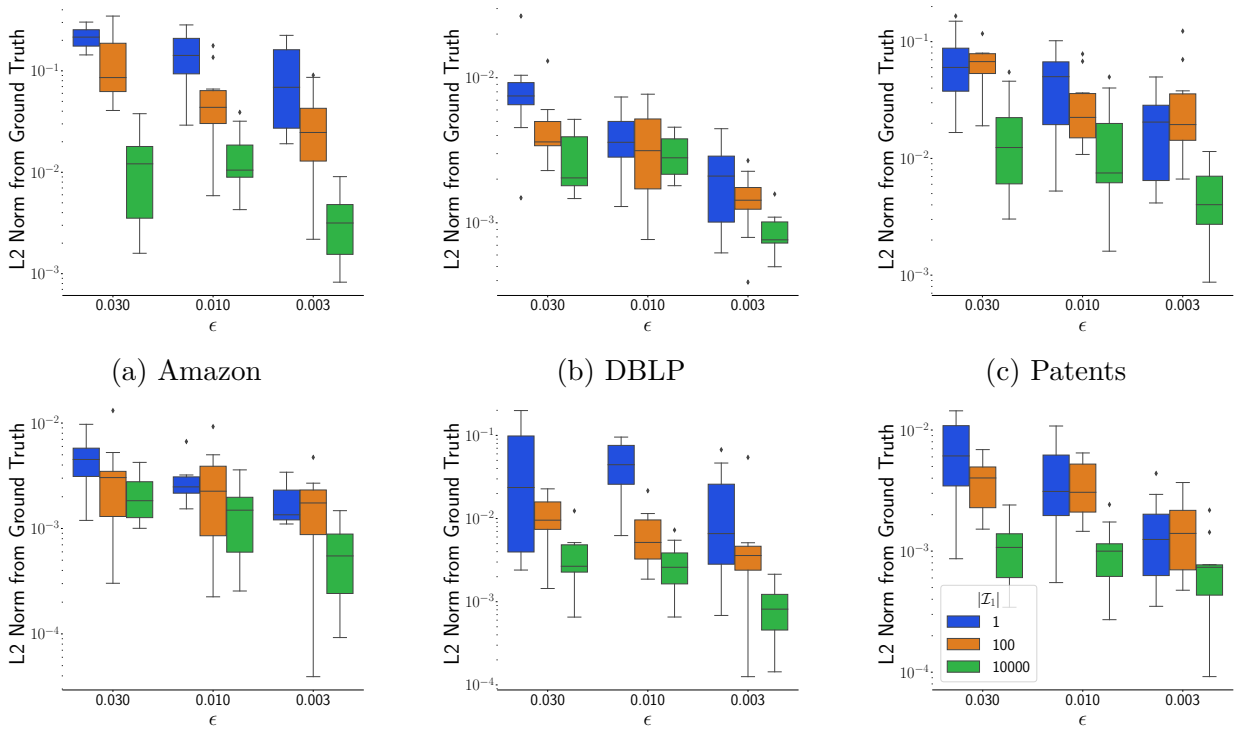
Space Scalability (Table 3.3). The trends in space usage mirror those of the running time, where we see an almost exponential increase w.r.t. k for *Motivo* compared to a near constant increase for *Ripple* despite its polynomial complexity (Proposition B.5.1). For example, in Amazon, *Motivo*'s space demand increases by 7.5× when k goes from 6 to 8 and

Table 3.2. Running time comparison between *Ripple* and *Motivo*. The last column shows that for large k, *Ripple* provides gains of up to 9 hours when Motivo can run to completion. *Motivo* crashes for large k on large graphs.

		Motivo Build-up only		***Ripple*** ($\epsilon = 0.003$)		***Ripple***
Dataset	**k**	**Time (hrs)**	$\frac{\text{Time}^{(k)}}{\text{Time}^{(k-2)}}$	**Time (hrs)**	$\frac{\text{Time}^{(k)}}{\text{Time}^{(k-2)}}$	**gain (hrs)**
Amazon	6	$\mathbf{0.002 \pm 0.000}$	–	0.020 ± 0.000	–	-0.018
	8	$\mathbf{0.006 \pm 0.000}$	3×	0.029 ± 0.000	1.4×	-0.023
	10	0.082 ± 0.000	13.7×	$\mathbf{0.056 \pm 0.000}$	1.9×	+0.026
	12	3.630 ± 0.002	44.3×	$\mathbf{0.095 \pm 0.002}$	1.7×	+3.535
DBLP	6	$\mathbf{0.002 \pm 0.000}$	–	0.013 ± 0.000	–	-0.011
	8	$\mathbf{0.007 \pm 0.000}$	3.5×	0.030 ± 0.000	2.3×	-0.023
	10	0.156 ± 0.000	22.3×	$\mathbf{0.082 \pm 0.000}$	2.7×	+0.074
	12	9.099 ± 0.002	58.3×	$\mathbf{0.105 \pm 0.002}$	1.3×	+8.994
Patents	6	$\mathbf{0.022 \pm 0.000}$	–	0.033 ± 0.000	–	-0.011
	8	0.098 ± 0.000	4.5×	$\mathbf{0.051 \pm 0.000}$	1.5×	+0.047
	10	> 1.1 *hrs, crashed*	–	$\mathbf{0.090 \pm 0.001}$	1.8×	–
	12	> 0.5 *hrs, crashed*	–	$\mathbf{0.117 \pm 0.003}$	1.3×	–
Pokec	6	$\mathbf{0.012 \pm 0.000}$	–	0.459 ± 0.142	–	-0.447
	8	$\mathbf{0.128 \pm 0.000}$	10.7×	0.759 ± 0.282	1.7×	-0.631
	10	5.965 ± 0.000	46.6×	$\mathbf{1.400 \pm 0.592}$	1.8×	+4.565
	12	> 1.5 *hrs, crashed*	–	$\mathbf{1.469 \pm 0.334}$	1×	–
LiveJ.	6	$\mathbf{0.024 \pm 0.000}$	–	0.351 ± 0.009	–	-0.327
	8	$\mathbf{0.205 \pm 0.000}$	8.5×	0.642 ± 0.074	1.8×	-0.437
	10	> 2.3 *hrs, crashed*	–	$\mathbf{1.76 \pm 1.550}$	2.7×	–
	12	> 0.7 *hrs, crashed*	–	$\mathbf{2.189 \pm 1.350}$	1.2×	–
Orkut	6	$\mathbf{0.032 \pm 0.000}$	–	0.669 ± 0.026	–	-0.637
	8	$\mathbf{0.585 \pm 0.006}$	18.3×	1.744 ± 0.983	2.6×	-1.159
	10	> 8.9 *hrs, crashed*	–	$\mathbf{2.633 \pm 1.065}$	1.5×	–
	12	> 1.8 *hrs, crashed*	–	$\mathbf{3.967 \pm 3.162}$	1.5×	–

Table 3.3. Space usage comparison between *Ripple* and *Motivo.Motivo* runs out of *disk* space for larger datasets in which $k \geq 10$, while *Ripple* scales almost linearly. *Ripple* saves up to 600 GB of space when *Motivo* can run.

		Motivo Build-up only		***Ripple*** ($\epsilon = 0.003$)		***Ripple***
Dataset	**k**	**Space (GB)**	$\frac{\text{Space}^{(k)}}{\text{Space}^{(k-2)}}$	**Space (GB)**	$\frac{\text{Space}^{(k)}}{\text{Space}^{(k-2)}}$	**gain (GB)**
Amazon	6	$\mathbf{0.53 \pm 0.00}$	–	4.69 ± 0.06	–	-4.16
	8	$\mathbf{4.00 \pm 0.00}$	7.5×	5.73 ± 0.12	1.2×	-1.73
	10	48.00 ± 0.00	12×	$\mathbf{7.38 \pm 0.36}$	1.3×	+40.62
	12	559 ± 0.00	11.6×	$\mathbf{9.09 \pm 1.02}$	1.2×	+549.91
DBLP	6	$\mathbf{0.50 \pm 0.00}$	–	4.58 ± 0.02	–	-4.08
	8	$\mathbf{4.00 \pm 0.00}$	8×	6.31 ± 0.00	1.4×	-2.31
	10	50.00 ± 0.00	12.5×	$\mathbf{7.99 \pm 0.01}$	1.3×	+42.01
	12	611.00 ± 0.00	12.2×	$\mathbf{10.45 \pm 0.02}$	1.3×	+600.55
Patents	6	$\mathbf{7.00 \pm 0.00}$	–	11.50 ± 0.05	–	-4.5
	8	66.00 ± 0.00	9.4×	$\mathbf{13.80 \pm 0.03}$	1.2×	+52.2
	10	> 800, *crashed*	–	$\mathbf{15.85 \pm 0.08}$	1.1×	> 800
	12	> 800, *crashed*	–	$\mathbf{18.12 \pm 0.10}$	1.1×	> 800
Pokec	6	$\mathbf{3.7 \pm 0.00}$	–	13.69 ± 0.06	–	-9.99
	8	36.00 ± 0.00	9.7×	$\mathbf{17.17 \pm 0.03}$	1.3×	18.83
	10	407.00 ± 0.00	11.3×	$\mathbf{20.31 \pm 0.01}$	1.2×	+386.69
	12	> 800, *crashed*	–	$\mathbf{22.82 \pm 0.03}$	1.1×	> 800
LiveJ.	6	$\mathbf{7.70 \pm 0.00}$	–	18.26 ± 0.02	–	-10.56
	8	73.00 ± 0.00	9.5×	$\mathbf{21.26 \pm 0.00}$	1.2×	+51.74
	10	> 800, *crashed*	–	$\mathbf{24.43 \pm 0.72}$	1.1×	> 800
	12	> 800, *crashed*	–	$\mathbf{27.75 \pm 0.00}$	1.1×	> 800
Orkut	6	$\mathbf{7.90 \pm 0.00}$	–	40.38 ± 0.00	–	-32.48
	8	78.00 ± 0.000	9.9×	$\mathbf{43.49 \pm 0.00}$	1.1×	+34.51
	10	> 800, *crashed*	–	$\mathbf{46.63 \pm 0.00}$	1.1×	> 800
	12	> 800, *crashed*	–	$\mathbf{49.73 \pm 0.00}$	1.1×	> 800



(a) Amazon (b) DBLP (c) Patents

(d) Pokec (e) Live Journal (f) Orkut

Figure 3.2. Accuracy and convergence analysis for 5-*CIS* s. I plot the L2-norm between the *Ripple* estimate and the exact value of the count vector $\mathcal{C}^{(5)}$ (Equation (3.9)) of all non-isomorphic subgraph patterns against various configurations of the parameters ϵ and $|\mathcal{I}_1|$. As expected, the accuracy improves as the error bound ϵ decreases and the number of seed subgraphs $|\mathcal{I}_1|$ increases. Each box and whisker represents 10 runs.

increases to 12× from 10 to 12. *Ripple*'s largest rate of increase is 1.4× when k goes from 6 to 8 for DBLP, and it saves up to 600 GB of space when Motivo does not crash.

3.5.2 Accuracy and Convergence Assessment

Next, I evaluate the accuracy and convergence of *Ripple* on small and large subgraph patterns, where the former refers to subgraph sizes in which the number of isomorphic subgraphs can be exactly computed using *ESCAPE* [14], i.e., $k \leq 5$.

Accuracy on Small k.

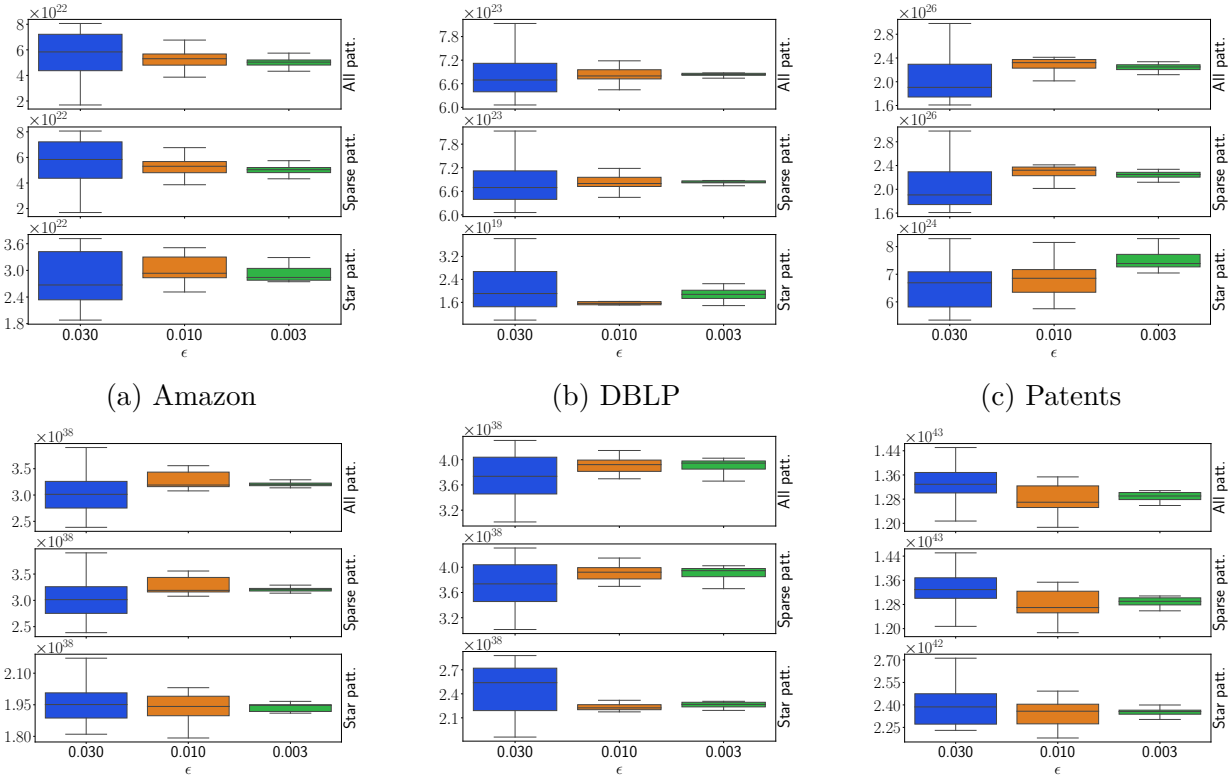
For $k \in \{3, 5\}$, I evaluate the L2-norm between the *Ripple* estimate and the exact value of the count vector $\mathcal{C}^{(k)}$ (Equation (3.9)) of all non-isomorphic subgraph patterns. Figure 3.2 shows results for $k = 5$ (where the number of patterns of interest $|\mathcal{H}| = 21$) for different settings of the parameters ϵ and $|\mathcal{I}_1|$. In all datasets, we note that the L2-norm decreases as ϵ decreases from 0.3 to 0.003 and as $|\mathcal{I}_1|$ increases from 100 to 10^4. Between the worst setting, $(\epsilon, |\mathcal{I}_1|) = (0.3, 100)$, and the best $(\epsilon, |\mathcal{I}_1|) = (0.003, 10^4)$, we see an error reduction close to an order of magnitude. This is due to Theorem 3.3.2 and Lemma 3 because reducing ϵ increases the number of tours, lowers the error and therefore leads to reduced error propagation. Increasing $\mathcal{I}_1$ also reduces the number of strata and therefore error propagation. Results for $k = 5$ using the L-∞ norm are deferred to Section B.6-Figure B.2.

Convergence for Large k.

When $k > 5$, subgraph counts for real-world graphs are computationally intractable. Therefore, I show that *Ripple* converges in these cases as I increase the computing effort. Consider the hypothesis that sparse patterns are frequent in power-law networks as k increases. To glean empirical evidence for this, I choose an appropriate pattern set $\mathcal{H}$ and equivalence relationship in Definition 3.4.1, and I use *Ripple* to compute the total number of k-*CIS* s and the number of sparse subgraphs and stars. A subgraph is defined as sparse if its density lies between 0 and 0.25, according to [76]. In Figure 3.3, I show that *Ripple* converges for all datasets, and as expected, most patterns are sparse, with close to half of the patterns in many of the studied networks being stars. This proportion is attenuated in DBLP and Patents, where dense substructures naturally emerge from collaboration/citation among the authors that these graphs represent.

3.6 Related Work

For better presentation, I split this section into two parts: (1) parallel *MCMC* techniques and (2) methods for subgraph counting.



(a) Amazon (b) DBLP (c) Patents

(d) Pokec (e) Live Journal (f) Orkut

Figure 3.3. Convergence of *Ripple* estimates of 12-*CIS* pattern counts. I estimate the total number of subgraphs $|\mathcal{V}^{(12)}|$ and the number of sparse patterns and stars. Estimates over 10 runs are presented as box and whiskers plots, which exhibit a reduction in variance as ϵ increases. Indeed, almost all patterns are sparse, and the most frequent substructure is a star.

3.6.1 Parallel *MCMC* through Splitting.

Since [27], [59], multiple techniques have been proposed to circumvent the burn-in period by splitting the chain into i.i.d. sample paths. This approach allows practitioners to compute unbiased estimates in parallel and determine confidence intervals. Perfect sampling methods based on coupling [61] require the transitions to be monotonic w.r.t. some ordering over the state space, and annealing/tempering [77] methods require some notion of temperature, which are absent in general graph random walks. Methods such as [46], [60], [78] require a minorization condition to hold, albeit implicitly.

Regeneration point-based methods on finite state chains [38]–[40], [50], [62], [69] are more general because they only rely on standard ergodicity conditions. Although [39], [40] used tours to estimate graph properties, [62], [69] proposed supernodes to reduce running times. The studies in [38], [50] further used supernode-based tours to estimate gradients in *RBM*s and to count subgraphs. To the best of my knowledge, no existing regeneration point method controls running times through stratification.

3.6.2 Subgraph Counting through Sampling.

Many random walk algorithms have been proposed to sample subgraphs, with some methods only capable of estimating subgraph pattern *distributions*, which is much easier than estimating *counts*. The studies of *GUISE* [21] and *RSS* [41] use a Metropolis-Hastings [79] walk, and the latter improves the mixing time of the underlying Markov chain using canonical paths [80]. Waddling [49] and *IMPRG* [20] perform a simple random walk over the input graph and use specialized estimators to sample 5-node patterns. Although *PSRW* [18] first proposed the *HON*-based random walk and *RGPM* [50] used tours on it to estimate subgraph counts, both are limited to $k \leq 5$ due to the size of the *HON*.

Multiple attempts to Monte Carlo sample subgraphs have been proposed whose scaling is limited because of the complexity of computing either the importance weights, rejection rate or variance [16], [17], [22]–[24]. Efficient methods that sample dense regions/subgraphs are unfortunately not extensible to sparse patterns [1], [2]. Motivo [19], [25] is an example of color-coding methods in which an index table is built using a deterministic dynamic program-

ming algorithm, which is then exploited to sample subgraphs uniformly and independently. However, *CC* methods suffer from the exponential time and space complexities associated with building and accessing the index table. *Motivo* proposed succinct index tables and efficient out-of-core I/O mechanisms to ameliorate this issue and extended the applicability of *CC* methods to larger subgraphs. Please, check [55] for an extensive survey on subgraph counting methods.

3.7 Summary

In this chapter, I propose the *Ripple* estimator that uses *sequentially stratified regenerations* to control the running time of a random walk tour-based *MCMC*. I prove that the estimator is consistent (w.r.t. the number of random walk tours) and that the time and memory complexity of my implementation for the subgraph counting problem is linear in the number of patterns of interest and polynomial in the subgraph size. I empirically verify my claims on multiple graph datasets and show that *Ripple* can accurately estimate subgraph counts with a smaller memory footprint compared to that of the state-of-the-art Motivo [19]. *Ripple* is currently the only subgraph pattern count estimator that can estimate $k = 10, 12$ node patterns in million-node graphs. Beyond my specific application, *Ripple* provides a promising way to expand the sphere of influence of regenerative simulation in finite-state reversible *MCMC*.

4. IMPROVING RBM TRAINING THROUGH STOPPING SETS

4.1 Introduction

Despite the significant recent advances in training discriminative neural network models, training generative models has proven more elusive. As with most neural network training methods, algorithms for training Restricted Boltzmann Machines (RBMs) [32]–[34], a class of energy-based generative neural network models, are unreasonably effective. Though, some argue, not yet effective enough for modern applications. In this chapter I seek to better understand and improve the training of RBMs.

RBM is a family of energy-based models with probability distribution over a state vector $\mathbf{x} = (\mathbf{v}, \mathbf{h})$ (assumed discrete w.l.o.g.), $\mathbf{v} \in \{0,1\}^{n_V}$ (e.g. an image) and binary latent variables $\mathbf{h} \in \{0,1\}^{n_H}$,

$$p(\mathbf{x}; \mathbf{W}) = \frac{1}{Z(\mathbf{W})} \mathrm{e}^{-E(\mathbf{x};\mathbf{W})}, \tag{4.1}$$

where $Z(\mathbf{W}) = \sum_{\mathbf{x}} \mathrm{e}^{-E(\mathbf{x};\mathbf{W})}$ is a partition function with finite mean, $\mathbb{E}[Z(\mathbf{W})] < \infty$, and $E(\mathbf{x}; \mathbf{W})$ is an energy function given by

$$E\Big(\mathbf{x} = (\mathbf{v}, \mathbf{h}); \mathbf{W} = (\mathbf{W}', \mathbf{b}, \mathbf{a})\Big) = -\mathbf{v}^\mathsf{T}\mathbf{W}'\mathbf{h} - \mathbf{b}^\mathsf{T}\mathbf{v} - \mathbf{a}^\mathsf{T}\mathbf{h}.$$

RBMs proved highly successful in many tasks, such as data generation [32], [33], [81] and as a pre-training step for feedforward neural networks [82], among others (see Bengio and Delalleau [83] and Erhan, Bengio, Courville, *et al.* [84]).

As computing $Z(\mathbf{W})$ directly is intractable for large state spaces, Markov Chain Monte Carlo (MCMC) methods are widely used to compute statistics of these models (including estimating the gradient $\partial p(\mathbf{x}; \mathbf{W})/\partial \mathbf{W}$). MCMC works by running a Markov chain (MC) $\Phi(\mathbf{W})$ with steady state $p(\mathbf{x}; \mathbf{W})$ to equilibrium. Metropolis-Hastings and Gibbs sampling are two general such approaches.

However, in the real world, one is expected to run the MC $\Phi(\mathbf{W})$ for only K steps, returning a state $\mathbf{x} \sim \hat{\boldsymbol{\pi}}(K)$, "approximately sampled" from the Markov chain's true steady

state distribution $p(\mathbf{x}; \mathbf{W})$. Starting from a random state, K needs to be quite large for this method to work.

Contrastive Divergence (CD-K) [32], [33], improves this procedure by starting the MC from the visible states of the training data. Empirically, CD-K works tremendously well to train RBMs with few hidden units (n_H small) even for K as low as $K = 1$ [32], [33], [85].

For high-dimensional RBMs, CD-K is less efficient and the reason is conjectured to be the longer mixing times [86], although concrete evidence is anecdotal as mixing times are hard to assess in high dimensions. While not the main focus of my work, armed with my techniques, I will further empirically explore possible reasons for this high-dimensional difficulty.

A Las Vegas transformation of RBM training. The main focus of this chapter is to recast MCMC estimation of RBMs as a Markov chain algorithm with stopping sets obtained from the training data. The size of the stopping set is a hyper-parameter that can be dynamically adapted during training based on computational trade-offs.

In standard RBM training using MCMC, the MC stops after a predefined number of K steps. In my approach, the MCMC can also stop if it reaches one of the states in the stopping set. Thus, MCMC running times are random (and are, in average, shorter than K). This approach is closer to a Las Vegas algorithm than a Monte Carlo algorithm: we aim to get *perfect samples* of a quantity with an algorithm that has random running times. I denote this approach Markov Chain Las Vegas with K maximum steps (MCLV-K).

I show that, by dynamically adapting K, MCLV-K can find unbiased estimates of the direction of the RBM gradient. Moreover, in contrast to standard MCMC, MCLV-K has an extra piece of information: whether or not the stopping set has been reached. I show that this knowledge provides novel ways to estimate gradients and partition functions of energy-based models.

And perhaps, one of the most interesting observations in this chapter comes from the **correspondence between CD-K and MCLV-K**. MCLV-1 is quite similar to CD-1 except for an added $\mathcal{S}$-*stopped* flag, where $\mathcal{S}$ is a set of *stopping* states defined later. Clearly, for $K \geq 2$, MCLV-K is distinct from CD-K, as the MC of MCLV-K may stop before performing all K steps.

Analyzing CD-K through my Las Vegas transformation, it is clear that CD-K has an unintended inspection paradox bias that can be corrected to further improve the RBM learning. Using the *reached-stopping-set* flag of MCLV-K, I design a new gradient estimator, denoted *Las Vegas Slope* (LVS), that empirically gives significantly better parameter estimates than standard CD-1 and CD-10 over the `MNIST` dataset according to the model's likelihood. `MNIST` is used in my experiments due to the long history of RBM development over this dataset.

Contributions. I claim the following contributions: **(1)** I introduce Markov Chain Las Vegas (MCLV-K). I show MCLV-K gives finite-sample unbiased and asymptotically consistent estimates of a variety of statistics of RBMs; further, I give two convergence bounds. I also show how to theoretically and empirically reduce the MCLV-K random running times using the training examples. **(2)** I show how MCLV-K can be used to design new ways to train Restricted Boltzmann Machines; I use MCLV-K to propose a novel RBM gradient estimator, *Las Vegas Slope* (LVS), which my empirical results show (for $K \in \{1, 3, 10\}$) improves parameter estimates of RBM over CD-1 and CD-10, over the `MNIST` dataset.

4.2 MCLV-K Estimation with Statistical Guarantees

In what follows I introduce some of the definitions used throughout this chapter. I introduce the concept of a tour (a MC which returns to the same state) and show that the return probability can be increased by collapsing a set of stopping states into a single state in Definition 4.2.1. The MC stops when it either reaches K steps or one of the states in the stopping set. Corollary 1 describes how this collapsing can be performed while preserving the statistical properties of the MC.

Theorem 4.2.1 introduces the MCLV-K estimator (that, among others, can estimate the partition function) and proves it is consistent, giving error bounds. And Theorem 4.2.2 shows that this estimator is also finite-running-time unbiased. The first results provides unbiased estimates of the partition function, and generalize these unbiased estimates to a broad family of functions.

The reader only interested in RBM gradient estimates can safely skip to the next section on training RBMs, after reading the preliminaries and the definition of the RBM stopping sets.

Preliminaries. We define state $\mathbf{x} = (\mathbf{v}, \mathbf{h})$ to consist of a visible vector, $\mathbf{v} \in V$, and a hidden vector $\mathbf{h} \in H$, where H and V are the set of all hidden and visible states, respectively. Let $\Phi(\mathbf{W})$ be an irreducible Markov chain with steady state $p(\mathbf{x}; \mathbf{W})$ over the states $\Omega := V \times H$. A MC is irreducible if all states communicate, which is trivially true for RBMs since the co-domain of the logistic function is $(0, 1)$ for any input in $\mathbb{R}$. If the Markov chain $\Phi(\mathbf{W})$ starts in equilibrium (or runs until equilibrium), the next transition gives us *one* independent sample $\mathbf{x}$ from the steady state $p(\mathbf{x}; \mathbf{W})$. The set $\{\mathbf{v}_n\}_{n=1}^N$ denotes the N visible examples of the training data. We often use $(\cdot)$, as in $g(\cdot)$, to denote that the statement over g is true for any valid input value.

RBMs can be trained by optimizing its parameters $\mathbf{W}$ in order to maximize the likelihood of the training data. Taking partial derivatives with respect to the weights results in a surprisingly simple update rule for $\mathbf{W}$:

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\frac{\partial \log(\sum_{\mathbf{h}} p(\mathbf{x}=(\mathbf{v}_n,\mathbf{h});\mathbf{W}))}{\partial \mathbf{W}} &= \sum_{\mathbf{h}\in H}\left(\frac{1}{N}\sum_{n=1}^{N} p(\mathbf{h}|\mathbf{v}_n;\mathbf{W})\mathbf{v}_n\mathbf{h}^T - \sum_{\mathbf{v}\in V} p((\mathbf{v},\mathbf{h});\mathbf{W})\mathbf{v}\mathbf{h}^T\right) \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{v}_n\mathbb{E}_{\mathbf{W}}[\mathbf{h}|\mathbf{v}_n]^T - \mathbb{E}_{\mathbf{W}}[\mathbf{v}\mathbf{h}^T],\end{aligned} \tag{4.2}$$

where the l.h.s. term of eq. (4.2) (also called *positive statistics*) is easily calculated from the training data. However, the r.h.s. term of eq. (4.2) (*negative statistics*) corresponds to the gradient of the partition function $Z(\mathbf{W})$, which is generally intractable to compute. More specifically, computing $E[\mathbf{v}\mathbf{h}^T]$ requires collecting model statistics $p(\mathbf{v}, \mathbf{h})$, either by running the MCMC Markov chain $\Phi(\mathbf{W})$ to equilibrium from any starting state or by direct computation of the expected value if we know the partition function $Z(\mathbf{W})$.

If the Markov chain $\Phi(\mathbf{W})$ is not run until equilibrium the gradient estimates have an unknown bias. In what follows we use Markov chain tours to take care of this bias.

Tours and Stopping Sets. Define a *tour* to be a sequence of ξ steps of the Markov chain $(\mathbf{X}(1), \ldots, \mathbf{X}(\xi))$ s.t. the state of the $(\xi+1)$-st step is the same as the starting state, i.e., $\mathbf{X}(1) = \mathbf{X}(\xi+1)$. Let $\Xi^{(r)} = (\mathbf{X}^{(r)}(1), \ldots, \mathbf{X}^{(r)}(\xi^{(r)}))$ denote the r-th tour, $r \geq 1$. The strong Markov property guarantees that if $s \neq r$, the sequences $\Xi^{(r)}$ are independent of $\Xi^{(s)}$. This independence guarantees that both $\Xi^{(r)}$ and $\Xi^{(s)}$ are sample paths obtained from the equilibrium distribution of the Markov chain. We will later use this property to obtain unbiased estimators of the partition function.

However, as is, tours are not a practical concept for RBMs because in such a large state space Ω, the tour is unlikely to return to the same starting state. We will, however, use a Markov chain property common to Metropolis-Hastings and Gibbs sampling Markov chains to significantly increase the probability of return by collapsing a large number of states into a single state.

RBM stopping set. Our stopping set $\mathcal{S}$ uses sampled hidden states from the training data, $\mathcal{H}_N^{(m)} = \{\mathbf{h}_n^{(1)}, \ldots, \mathbf{h}_n^{(m)} : \mathbf{h}_n \sim p(\mathbf{h}|\mathbf{v}_n; \mathbf{W})\}$, where $\{\mathbf{v}_n\}_{n=1}^N$ is the training data. Often we will use $m = 1$, but we can change the size of $\mathcal{H}_N^{(m)}$ by changing m. The stopping set contains all hidden states in $\mathcal{H}_N^{(m)}$ and all possible visible states

$$\mathcal{S}_{\text{HN}}^{(m)} = \bigcup_{\mathbf{h} \in \mathcal{H}_N^{(m)}, \mathbf{v} \in V} \{(\mathbf{v}, \mathbf{h})\}, \tag{4.3}$$

and $p(\mathbf{h}|\mathbf{v}; \mathbf{W})$ is the conditional probability of $\mathbf{h}$ given $\mathbf{v}$ using model parameters $\mathbf{W}$. Most of our theoretical results apply to any stopping set that is a proper subset of the state space, $\mathcal{S} \subset \Omega$. In practice, note that we *do not* store $\mathcal{S}_{\text{HN}}^{(m)}$ in memory, rather we *only keep* $\mathcal{H}_N^{(m)}$ *in memory*, as reaching a hidden state in $\mathcal{H}_N^{(m)}$ is enough to guarantee we need to stop. This requires only $O(mN)$ space, where N is the number of training observations.

Definition 4.2.1 (Stopping-set-Collapsed MC)**.** *Consider an arbitrary stopping set $\mathcal{S} \subset \Omega$. A state-collapsed MC is a transformation of MC $\Phi(\mathbf{W})$ with state space Ω, into a new MC $\Phi'(\mathbf{W})$ with state space $\Omega' = \Omega \backslash \mathcal{S} \cup \{\mathcal{S}\}$, where $\mathcal{S}$ is a new state formed by collapsing all the*

states in $\mathcal{S}$*. The transition probabilities between states* $\Omega' \cap \Omega$ *are the same as in* $\Phi(\mathbf{W})$*. The transition probabilities from* S *to states in* $\Omega' \backslash \{S\}$ *are*

$$p_{\Phi'}(S, \mathbf{x}) = \frac{\sum_{\mathbf{y} \in \mathcal{S}} \mathrm{e}^{-E(\mathbf{y};\mathbf{W})} p_{\Phi}(\mathbf{y}, \mathbf{x})}{Z_{\mathcal{S}}(\mathbf{W})}, \quad \forall \mathbf{x} \in \Omega' \backslash \{S\},$$

where $Z_{\mathcal{S}}(\mathbf{W}) = \sum_{\mathbf{y} \in \mathcal{S}} \mathrm{e}^{-E(\mathbf{y};\mathbf{W})}$*, and* p_a *indicates the probability transition matrix of MC* a*.*

The transitions from states $\Omega' \backslash \{S\}$ *to state* S *are*

$$p_{\Phi'}(\mathbf{x}, S) = \sum_{\mathbf{y} \in \mathcal{S}} p_{\Phi}(\mathbf{x}, \mathbf{y}), \quad \forall \mathbf{x} \in \Omega' \backslash \{S\}.$$

It is important to distinguish the MC in Definition 4.2.1 from general MC state aggregation methods such as lumpability [87] and interactive aggregation-disaggregation methods [88]. In the following corollary, we see that the MC in Definition 4.2.1 affects the steady state, unlike general MC aggregation methods that leave the steady state undisturbed. Thankfully, later we will be able to correct the distortion imposed by Definition 4.2.1 because we know the steady state distribution of the states inside $\mathcal{S}$ up to a normalizing constant.

Corollary 1 (Simulating $\Phi'(\mathbf{W})$ from $\Phi(\mathbf{W})$)**.** *For any MC* $\Phi(\mathbf{W})$ *resulting from standard Gibbs sampling or Metropolis-Hastings (MH) MCMCs, we can cheaply simulate the transitions in and out of* S *of Definition 4.2.1 by: (a)* $p_{\Phi'}(S, \mathbf{x})$*, we first sample a state* $\mathbf{y}$ *with replacement from* $\mathcal{S}$ *with probability* $\mathrm{e}^{-E(\mathbf{y};\mathbf{W})}/Z_{\mathcal{S}}(\mathbf{W})$ *and then perform a transition* $p_{\Phi}(\mathbf{y}, \mathbf{x})$*; (b)* $p_{\Phi'}(\mathbf{x}, S)$ *is also simulated by performing a transition* $p_{\Phi}(\mathbf{x}, \mathbf{y})$*, and stopping the MC if* $\mathbf{y} \in \mathcal{S}$*. The simulated* $\Phi'(\mathbf{W})$ *is ergodic and time-reversible.*

The proof is in the appendix. It follows from the fact that $\Phi(\mathbf{W})$ is the MC of Gibbs sampling and MH and, thus, time-reversible [89]. Time reversibility imposes a set of necessary and sufficient conditions in the form of detailed balance equations [90, Theorem 6.5.2]. A little algebra shows that the sampling procedure in Corollary 1 using $\Phi(\mathbf{W})$ is stochastically equivalent to $\Phi'(\mathbf{W})$.

4.2.1 MCLV-K Estimator

Following Corollary 1, a tour starts by sampling the initial tour state $\mathbf{x}$ and stopping when the tour reaches the stopping set $\mathcal{S}$. We now want to truncate all return times of tours greater than some value $K \geq 1$, i.e., we will only observe the complete r-th tour $(\mathbf{x}, \mathbf{X}^{(r)}(2), \ldots, \mathbf{X}^{(r)}(\xi^{(r)}))$ if $\xi^{(r)} \leq K$. Otherwise, we observe only the first K states of the tour: $(\mathbf{x}, \mathbf{X}^{(r)}(2), \ldots, \mathbf{X}^{(r)}(K))$. The $\mathcal{S}$-*stopped* flag for tour r is **true** if $\xi^{(r)} \leq K$, otherwise it is **false**.

Lemma 4 (Perfect sampling of tours)**.** *Let*

$$\mathcal{C}_k = \{(\mathbf{x}, \mathbf{X}^{(\mathrm{i})}(2), \ldots, \mathbf{X}^{(\mathrm{i})}(k))\}_{\mathrm{i}}$$

be a set of tours of length $k \leq K$, with $\mathbf{x}$ sampled from $\mathcal{S}$ according to some distribution.

Then, there exists a distribution G_k such that the random variables

$$\mathcal{G}_k \equiv \{g(\sigma) : \forall \sigma \in \mathcal{C}_k\}$$

are i.i.d. samples of G_k, with g defined over the appropriate σ-algebra (e.g., k RBM states) with $\|g(\cdot)\|_1 \leq \infty$.

Moreover, if we perform M tours, these tours finish in finite time and $\{\xi^{(r)}\}_{r=1}^{M}$ is an i.i.d. sequence with a well-defined probability distribution $p(\xi^{(\cdot)} = k)$.

The Las Vegas parallel is observed when we notice that any MCMC metric can be perfectly sampled from the tours. The tour lengths are sampled from a distribution $p(\xi^{(\cdot)} = k)$. And, for any given tour length k, the metric of interest g is perfectly sampled from G_k. The maximum tour length K only cuts off the tail of $p(\xi^{(\cdot)} = k)$ beyond $k > K$, which allows us to bound the sampling error.

Theorem 4.2.1 (MCLV-K RBM Estimator)**.** *Let $p(\mathbf{x}; \mathbf{W})$, $E(\mathbf{x}; \mathbf{W})$, and $Z(\mathbf{W})$ be as described in eq.(4.1). Let*

$$F(\mathbf{W}, f) = Z(\mathbf{W}) \sum_{\mathbf{x} \in \Omega} f(\mathbf{x}) p(\mathbf{x}; \mathbf{W}), \tag{4.4}$$

where $f : \Omega \to \mathbb{R}^n$, $n \geq 1$, $\|f(\cdot)\|_1 < \infty$, *and* $\|\cdot\|_1$ *is the* l_1 *norm. Let* $\Phi(\mathbf{W})$ *be a time-reversible MC with state space* Ω *and steady state distribution* $\{p(\mathbf{x}; \mathbf{W})\}_{\mathbf{x}\in\Omega}$. *Let* $\mathcal{S} \subset \Omega$ *be a proper subset of the states of* $\Phi(\mathbf{W})$.

Sample $\mathbf{x}' \in \mathcal{S}$ *with probability* $\mathrm{e}^{-E(\mathbf{x}';\mathbf{W})}/Z_{\mathcal{S}}(\mathbf{W})$ *and let*

$$(\mathbf{X}^{(r)}(1) = \mathbf{x}', \mathbf{X}^{(r)}(2), \ldots, \mathbf{X}^{(r)}(\xi^{(r)}))$$

be a sequence of discrete states of the r*-th* $\mathcal{S}$*-stopped tour, where we stop the tour if one of two conditions are met: (a) we have reached* K *steps, or (b) when we reach any state in* $\mathcal{S}$, *i.e.,* $\mathbf{X}^{(r)}(\xi + 1) \in \mathcal{S}$. *Then, for* $R \geq 1$ *tours, let* $\mathcal{C}_k^{(R)}$ *be the set of finished tours in* $k \leq K$ *steps, (as defined in Lemma 4). For the sake of simplicity, we henceforth refer to* $\mathcal{C}_k^{(R)}$ *simply as* $\mathcal{C}_k$. *The estimator*

$$\hat{F}^{(K,R)}(\mathbf{W}, f) = \frac{1}{\sum_{k=1}^{K} |\mathcal{C}_k|} \sum_{\mathbf{y}\in\mathcal{S}} \mathrm{e}^{-E(\mathbf{y};\mathbf{W})} \times \sum_{k=1}^{K} \sum_{(\mathbf{X}(1),\mathbf{X}(2),\ldots,\mathbf{X}(k))\in\mathcal{C}_k} \sum_{h=1}^{k} f(\mathbf{X}(h)) \tag{4.5}$$

is an estimate of $F(\mathbf{W}, f)$ *in eq. (4.4) with a bias upper bounded by* $B \cdot (E[\xi] - \sum_{k=1}^{K-1} p(\xi > k))$, *where* $p(\xi > k)$ *is the probability that a tour has length greater than* k *and* $B \geq \sup_{\mathbf{x}\in\Omega} \|f(\mathbf{x})\|_1$.

Theorem 4.2.1 gives a basic estimator from the MCMC tours. The gradient estimates will be explicitly derived in the next section. In my experiments I show how to estimate $p(\xi > k)$. For the partition function and gradient estimates, it is also trivial to obtain a bound on B using the RBM weights $\mathbf{W}$ [83].

Theorem 4.2.2 (Geometrically Decaying Tour Length Tails)**.** *Let* $p(\xi > k)$ *be the probability that a tour has length greater than* k. *If there exists a constant* $\epsilon > 0$ *s.t.* $\inf_{\mathbf{x}\in\Omega\setminus\mathcal{S}} \sum_{\mathbf{y}\in\mathcal{S}} p_{\Phi}(\mathbf{x}, \mathbf{y}) \geq \epsilon$ *then, there exists* $0 < \alpha < 1$, $\log p(\xi > k) = k \log \alpha + o(k)$, *i.e.,* ξ *has a geometrically decaying tail.*

Theorem 4.2.2 shows conditions of a geometric decay in the tail of $p(\xi > k)$. And in practice it means that tours cannot be "heavy tail" long and, thus, making the bound in Theorem 4.2.1 tighter.

4.2.2 MCLV-K Finite-Sample Unbiasedness

In what follows we dynamically increase K until the MC reaches a state in the stopping set.

The following theorem shows that this procedure gives unbiased estimates of $F(\mathbf{W}, f)$.

Theorem 4.2.3 (Unbiased Partition-scaled Function Estimates by Dynamic Adaptation of K)**.** *Consider the estimator in Theorem 4.2.1 and let us dynamically grow K (denoted K_{dyn}) until the MC reaches a stopping state in $\mathcal{S}$. Then, for $R \geq 1$ tours,*

$$\mathbb{E}[\hat{F}^{(K_{dyn},R)}(\mathbf{W}, f)] = F(\mathbf{W}, f), \tag{4.6}$$

is an **unbiased estimator** *and the estimator is consistent, that is, almost surely*

$$\lim_{R\to\infty} \hat{F}^{(K_{dyn},R)}(\mathbf{W}, f) = F(\mathbf{W}, f)\,,$$

and K_{dyn} is finite.

Moreover, for $\epsilon > 0$,

$$p\left(\left\|\hat{F}^{(K_{dyn},R)}(\mathbf{W}, f) - F(\mathbf{W}, f)\right\|_1 \geq \epsilon\right) \leq \alpha_{R,Z_{\mathcal{S}}(\mathbf{W})},$$

where, R is the number of tours, $\alpha_{R,Z_{\mathcal{S}}(\mathbf{W})} = \frac{B^2}{\epsilon^2 R}\left(\frac{(Z(\mathbf{W}))^2}{(Z_{\mathcal{S}}(\mathbf{W}))^2\delta} + 1\right)$, $B \geq \sup_{\mathbf{x}\in\Omega} \|f(\mathbf{x})\|_1$ is an upper bound on the absolute value of $f(\cdot)$ over the state space Ω, δ is the spectral gap of the transition probability matrix of $\Phi(\mathbf{W})$.

Corollary 2 (Unbiased Partition Function Estimation)**.** *Let $f_1(x) = 1$, then*

$$\mathbb{E}[\hat{F}^{(K_{dyn},R)}(\mathbf{W}, f_1)] = Z(\mathbf{W}),$$

is an unbiased estimator of the partition function.

4.3 Training Restricted Boltzmann Machines

In what follows I explore the connections between MCLV-K and learning RBMs using MCMC methods. First, I show how MCLV-K can provide a finite-sample unbiased and asymptotically consistent estimate of the direction of the RBM gradient.

4.3.1 MCLV-K Gradient Estimates

In what follows I will provide an estimate of the gradient of the negative log-likelihood of RBMs using MCLV-K. My gradient will have a scaling factor but the gradient direction is the same as the original gradient:

$$\nabla_{\mathbf{W}}\mathcal{L}_Z = \frac{Z(\mathbf{W})}{Z_{\mathcal{S}}(\mathbf{W})}\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{v}_n^{\mathsf{T}}\mathbb{E}_{\mathbf{W}}[\mathbf{h}|\mathbf{v}_n] - \mathbb{E}_{\mathbf{W}}[\mathbf{v}^{\mathsf{T}}\mathbf{h}]\right).$$

The scaling $Z(\mathbf{W})/Z_{\mathcal{S}}(\mathbf{W})$ is constant given $\mathbf{W}$. In my current implementation, I use Corollary 2 to estimate $Z(\mathbf{W})/Z_{\mathcal{S}}(\mathbf{W})$ and divide the gradient by it, compensating for the scaling at essentially no computational or memory cost.

Corollary 3 (LVS-K: The *Las Vegas Slope* Estimator)**.** *Let* $\Phi(\mathbf{W})$, $\mathcal{S}$, $\mathbf{x}$, *the tour* $(\mathbf{x}, \mathbf{X}^{(r)}(2), \ldots, \mathbf{X}^{(r)}(\xi^{(r)}))$, R, K, *and* $\mathcal{C}_k$ *be as defined in Theorem 4.2.1. Then, for a learning rate* $\eta > 0$,

$$\widehat{\nabla_{\mathbf{W}}\mathcal{L}_{LVS}}(K,R) = \eta\left(\frac{\widehat{\mathbb{E}}[\xi]}{N}\sum_{n=1}^{N}\frac{\partial E(\mathbf{x}_n;\mathbf{W})}{\partial\mathbf{W}} - \frac{\sum_{k=1}^{K}\sum_{(\mathbf{X}(1),\ldots,\mathbf{X}(k))\in\mathcal{C}_k}\sum_{\mathrm{i}=1}^{k}\frac{\partial E(\mathbf{X}^{(r)}(k);\mathbf{W})}{\partial\mathbf{W}}}{\sum_{k=1}^{K}|\mathcal{C}_k|}\right), \tag{4.7}$$

is a consistent $(K, R \to \infty)$ *estimator of the energy-model gradient in eq.*(4.2), *where* $\widehat{\mathbb{E}}[\xi] = \frac{\sum_{k=1}^{K}|\mathcal{C}_k|k}{\sum_{k=1}^{K}|\mathcal{C}_k|}$ *is the empirical expectation of the tour lengths.*

Moreover, the contribution of a tour of length k *to the negative statistics of the gradient is proportional to*

$$P[\xi = k]\cdot k\cdot\mathbb{E}[\partial E(\tilde{\mathbf{X}}_k;\mathbf{W})/\partial\mathbf{W}],$$

where $\tilde{\mathbf{X}}_k$ *is a random state of a tour of length* k*. If the Markov chain* $\Phi(\mathbf{W})$ *satisfies the conditions of Theorem 4.2.2, then* $P[\xi = k] \cdot k = \mathrm{e}^{-O(k)}$*, so that* extremely long *tours do not influence the gradient.*

Corollary 4 (Unbiased Gradient Direction Estimator: LVS-K_{dyn})**.** *Consider the estimator in Corollary 3 and let us dynamically grow* K *(denoted* K_{dyn}*) until the MC reaches a stopping state in* $\mathcal{S}$*. Then,*

$$\mathbb{E}\left[\widehat{\nabla_{\mathbf{W}}\mathcal{L}_{LVS}}(K_{dyn}, R)\right] \propto \nabla_{\mathbf{W}}\mathcal{L}_Z,$$

is an unbiased estimate of the RBM gradient direction.

The proofs of the two above corollaries follow directly from Theorems 4.2.1 and 4.2.3, respectively.

4.3.2 Correspondence and Differences Between LVS-K and CD-K

In this section I explore a correspondence between LVS-K (proposed in Corollary 3) and CD-K to train RBMs. I will also emphasize some differences that will give us some new insights into CD-K. The correspondence is as follows: **(a)** consider a mini-batch of training examples $\{\mathbf{v}_{\mathrm{i}}\}_{\mathrm{i}=1}^{N}$; **(b)** the stopping set is $\mathcal{S}_{HN}^{(m)}$, described in eq.(4.3); **(c)** the number of tours R of LVS-K is the number of training examples in the mini-batch N, i.e., $R = N$.

One can readily verify that the Gibbs sampling updates of LVS-K and CD-K are similar except for the following key differences: **(i)** LVS-K starts at a state $\mathbf{x}$ of $\mathcal{S}_{HN}^{(m)}$ with probability proportional to $\exp(-E(\mathbf{x}; \mathbf{W}))$, CD-$K$ starts uniformly over the training examples. Thus, the negative phase of LVS-K tends to push the model away from unbalanced probabilities over the training examples. **(ii)** at every Markov chain step, LVS-K stops early if it has reached a state in $\mathcal{S}_{HN}^{(m)}$, while CD-K will always perform all K steps. **(iii)** the gradient estimates of LVS-K use only the completed tours, while CD-K uses all tours; **(iv)** the gradient estimates of LVS-K use all states visited by the MC during a tour, while CD-K uses only the last visited state.

A long sequence of states visited by the CD-K Gibbs sampler can be broken up into tours if the stopping state contains only the starting state. Figure 4.1 illustrates three MCMC

runs starting at visible states representing "7", "3", and "4", broken up into tours whenever the starting hidden state is sampled again. Starting from visible state "7", CD-K ignores the completed tour *Tour 1*, which LVS-K uses for its gradient estimate; and CD-K proceeds to use the state in the middle of *Tour A* for its gradient estimate. CD-K also uses a state in the incomplete *Tour 2*, which LVS-K ignores as incomplete. Finally, CD-K ignores *Tour 3* and proceeds to use the state in the beginning of *Tour B* for its gradient estimate.

This means that, for $K \geq 2$, CD-K is more likely to sample states from longer tour than shorter tours. This bias is the inspection paradox [91]. Interestingly, this bias makes CD-K, $K \geq 2$, significantly different from CD-1, which has no such bias. Note that LVS-K has the opposite bias: it ignores tours longer than K; the bias of LVS is measurable (Theorem 4.2.1) if we can estimate the average tour length.

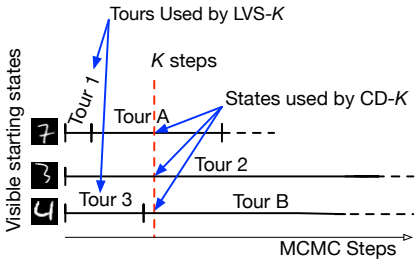


Figure 4.1. CD-k bias towards longer tours for $k \geq 2$;

4.3.3 Computational Complexity

In this section I give the time and space complexities of LVS-K and CD-K. Let $|\mathbf{W}|$ denote the number of elements in $\mathbf{W}$, $= n_V * n_H$ and $n_X = n_V + n_H$. In terms of space, LVS-K needs $O(N\, m\, n_X)$ space to store the $\mathcal{H}_N^{(m)}$ which is m times the requirement for CD-K. At every epoch, LVS-K samples a stopping set, which involves a matrix multiplication followed by algebraic operations over a matrix. The matrix multiplication which takes $O(Nm|\mathbf{W}|)$ upper bounds the time. Computing the free energies of the hidden state also takes the same time. Adding the states of the stopping set to a heap for easier sampling takes $O(Nm)$ time and allows us to sample starting states for the tours in $O(N \log(Nm))$. Every Gibbs step is

again bounded by the time taken for matrix multiplication which takes a total of $O(N|\mathbf{W}|K)$ time. Checking stopping set membership takes $O(NKn_X)$ amortized time assuming that standard algorithms used by hash sets, e.g. MD5, take $O(n_X)$ time to evaluate. Computing the gradient and updating $\mathbf{W}$ takes $O(N|\mathbf{W}|)$ time.

Therefore LVS-K takes $O(NKn_X + Nm|\mathbf{W}| + NK|\mathbf{W}|) \equiv O(N|\mathbf{W}|(m+K))$ time, compared to CD-K which takes $O(NK|\mathbf{W}|)$. In the general case $m \in O(K)$, $\therefore$ the asymptotic complexity of CD-K and LVS-K are the same.

4.4 Related Work

AIS, MC changes, and CD-K extensions. Annealed Importance Sampling (AIS) [92], [93] uses two distinct Markov transitions kernels and has been applied by Salakhutdinov and Murray [94] to obtain unbiased estimates of the partition function of an RBM. Like AIS, the Russian roulette pseudo-marginal likelihood is also a Markov chain modification to sample from the steady state distribution [95]. These modifications cannot be readily applied to the original RBM Markov chain, nor they provide insights into the learning process. MCLV-K is a new tool that can be used from visual inspection of convergence to proposing new gradient estimators, as seen in my empirical results.

RBMs are powerful models [96] and the analysis of CD-K has a long history [33]. A few past studies have focused on how CD-K learns well RBMs [85], [97], have some fixable issues learning RBMs [98]–[100], may approximate some objective function [32], or do not approximate any objective function [83], [86]. Orthogonally, Persistent Contrastive Divergence (PCD) [101] improves CD-K in some problems by using the starting state of the CD-K Markov chain as the end state of the previous training epoch (simulating a single sample path, assuming the MC does not change much between epochs, which is not always true [98]). Clearly, PCD could be adapted as a MCLV method, which I see as future work.

The presence of training data is key to the practicality of MCLV. Without training data, obtaining error bounds with MCLV can be prohibitively expensive. In the worst-case, there is no polynomial time algorithm that can estimate the probabilities of an RBM model within a constant factor [102], assuming P$\neq$NP. But most real-world machine learning problems are

supposed to be much easier than general MCMC results would have us believe. We are *given* a good hint of what should be a larger number of high-probability states in the steady state: the states containing the training examples. Unfortunately, vanilla MCMC methods do not incorporate this extra information to speed up convergence in a principled way. I believe the lessons learned in this chapter will be invaluable to design new classes of Markov chain methods tailored to machine learning applications.

Las Vegas algorithms for Markov chain sampling. Perfect Sampling [61], [103]–[107] is an example of a Las Vegas algorithm for MCMC applications. Unfortunately, energy-based models can easily reach trillions of states while perfect sampling methods rarely scale well unless some specific MC structure can be exploited. I are unaware of clever CFTP constructions for arbitrary energy-based models.

Mykland, Tierney, and Yu [46] with a few follow-up works first proposed the use of regeneration in the context of MCMC to estimate mixing times, however these techniques are mostly of theoretical interest [108]–[111] rather than of practical utility for energy-based models. Path coupling is another alternative to estimate mixing times [112]. More recently, path coupling was used to develop a theory of Ricci curvature for Markov chains [113]. The connections between Ricci curvature estimation and MCLV-K are worth exploring in future work.

4.5 Empirical Results

My experiments use the `MNIST` dataset, which consists of 70,000 images of digits ranging from 0 to 9, each image having 28×28 pixels (a total of 784 pixels per image), divided into 55,000 training examples and 15,000 test examples. I use this dataset for historical reasons. `MNIST` is arguably the most extensively studied dataset in RBM training, e.g. [32], [33], [81]–[83], [85], [101]. My goal is to show that MCLV-K is able to give new insights into RBM training (and improved performance) even in a studied-to-death dataset such as `MNIST`. The experimental details of my empirical results are presented in the appendix. I use LVS-1 to train the RBM model used in the following experiments (CD-K tends to give very

high probability to a few examples in the training data). I observe little difference between LVS-1, LVS-3, and LVS-10 (for reasons that will be clear soon).

RBM learning. My first set of empirical results compares LVS-K, $K \in \{1, 3, 10\}$, CD-K, $K \in \{1, 10\}$ and PCD-K, $K \in \{1, 10\}$ by training an RBM using stochastic gradient descent, where the gradient estimates are computed using the respective methods. I train RBMs with $n_H = 32$ hidden neurons for a total of 100 epochs (inclusive of 15 warm-up epochs of CD-1 for LVS-K), using a learning rate of 0.1 which decays according to a Robbins-Monro schedule. Weight decay and momentum were not used. The initial $\mathbf{W}$ weights are sampled uniformly from $U\left(\frac{-0.1}{\sqrt{n_V+n_H}}, \frac{0.1}{\sqrt{n_V+n_H}}\right)$, where n_V and n_H denote the number of visible and hidden neurons, respectively. Hidden biases are initialized as zero, while visible biases are initialized as $\log(p_\mathrm{i}/(1-p_\mathrm{i}))$ [33], where p_i is the empirical probability of feature i being active.

The small number of hidden units is to enable me to evaluate the true performance: I compute the exact partition function of the trained RBM and calculate the average log-likelihood $\frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{v}_n)$. All results are means calculated from 10 executions. In all LVS-K experiments I use $m = 1$, for simplicity. The negative log-likelihood of LVS-K, PCD-K and CD-K are presented in Table 4.1.

Subsequently, in order to compare my results with those presented in Tieleman (2008), I train RBMs with $n_H = 25$ and initial learning rates between 10^{-4} and 1. I observe that larger learning rates (10^{-1} to 1) are more appropriate for LVS-K, resulting in faster convergence and increased performance. Small rates (e.g. 10^{-4}) cause tours to rarely finish, severely slowing down the training. On the other hand, CD-K and PCD-K fail to converge with learning rates slightly larger than 10^{-2}. The results for this experiment, along with the best learning rates for each method, are presented in Table 4.2.

In conclusion, LVS-K drastically (and paired t-test significantly) outperforms CD-K and PCD-K w.r.t. the log-likelihood in all settings, even LVS-1 performs significantly better than PCD-10. However I was unable to reproduce the likelihood of ≈ -130 for PCD achieved by Tieleman (2008).

Tours lengths and stopping state. I now analyze the tour lengths as a function of: (a) n_H, the number of hidden units, and (b) the size of the stopping set $|\mathcal{S}_{NH}^{(m)}|$, where $\mathcal{S}_{NH}^{(m)}$ is

Table 4.1. (Higher is better) Average log-likelihood on the `MNIST` dataset using a RBM with 32 hidden neurons. Results are means over 10 executions after 100 epochs.

Method	Training	Testing
CD-1	-167.3 (2.7)	-166.6 (2.8)
CD-10	-154.3 (3.3)	-153.4 (3.3)
PCD-1	-153.0 (4.9)	-152.1 (4.7)
PCD-10	-139.3 (3.2)	-138.5 (3.3)
LVS-1	**-134.0 (1.0)**	**-133.3 (1.0)**
LVS-10	**-133.3 (1.0)**	**-132.6 (1.0)**
LVS-3	**-133.7 (0.8)**	**-132.9 (0.7)**

Table 4.2. (Higher is better) Average log-likelihood on the `MNIST` dataset using a RBM with 25 hidden neurons. Results are means over 10 executions after 100 epochs, using appropriate learning rates for each method.

Method	Learning Rate	Training	Testing
CD-1	0.01	-169.8 (2.6)	-169.0 (2.6)
CD-10	0.01	-156.4 (0.5)	-155.6 (0.5)
PCD-1	0.01	-147.8 (0.5)	-147.0 (0.5)
PCD-10	0.01	-147.4 (0.5)	-146.7 (0.5)
LVS-1	**0.1**	**-138.3 (1.3)**	**-137.5 (1.4)**
LVS-10	**0.1**	**-138.1 (1.1)**	**-137.4 (1.2)**
LVS-3	**0.1**	**-138.2 (1.0)**	**-137.5 (1.1)**

built from the training data as defined in eq.(4.3). Note that the r-th tour ends at state $\mathbf{X}^{(r)}(\xi) = \left(\mathbf{v}^{(r)}(\xi), \mathbf{h}^{(r)}(\xi)\right)$ whenever $\mathbf{X}^{(r)}(\xi+1) \in \mathcal{S}_{NH}$, and that the stopping criteria only truly depends on $\mathbf{h}^{(r)}(\xi+1)$ since $\mathcal{S}_{NH}^{(m)}$ contains all possible visible states.

Figure 4.2a shows the CCDF of the tour lengths for different values of n_H. Most tours are extremely short for RBMs with few hidden neurons (for $n_H = 16$, more than 99% have length one), but significantly increase as we increase n_H with a very heavy tail. Thus, it is expected that we see little difference between LVS-1, LVS-3, and LVS-10. Moreover, these heavy tails likely causes strong inspection-paradox biases for CD-K in high-dimensional RBMs.

Most importantly, Figure 4.2a shows that tours either return within one step or are unlikely to return for a very long time. A closer inspection at these one-step tours, shows that over 99% of the cases have the hidden state being the starting state. *Thus, it seems that RBMs (even with few hidden neurons) are just memorizing the training data, not learning*

how to generate new digits. I conjecture, however, that if my training could force the tours to stop at distinct hidden states, and requires the tours to be possibly longer (but not too-long), the RBM might be taught how to generate new digits.

Using $n_H = 32$ hidden neurons, Figure 4.2b shows the probability that a tour takes more than k steps, as we increase the number of stopping states by setting the values of $m \in \{1, 4, 7\}$ in $\mathcal{S}_{NH}^{(m)}$. We see that the probability of tours finishing in a single step increases as we add more states to the stopping state. Thus, increasing the stopping set size can significantly shorten the tours, which in turn improves the estimates of MCLV-K and LVS-K, and is an avenue to ameliorate MCMC issues in high-dimensional RBMs.

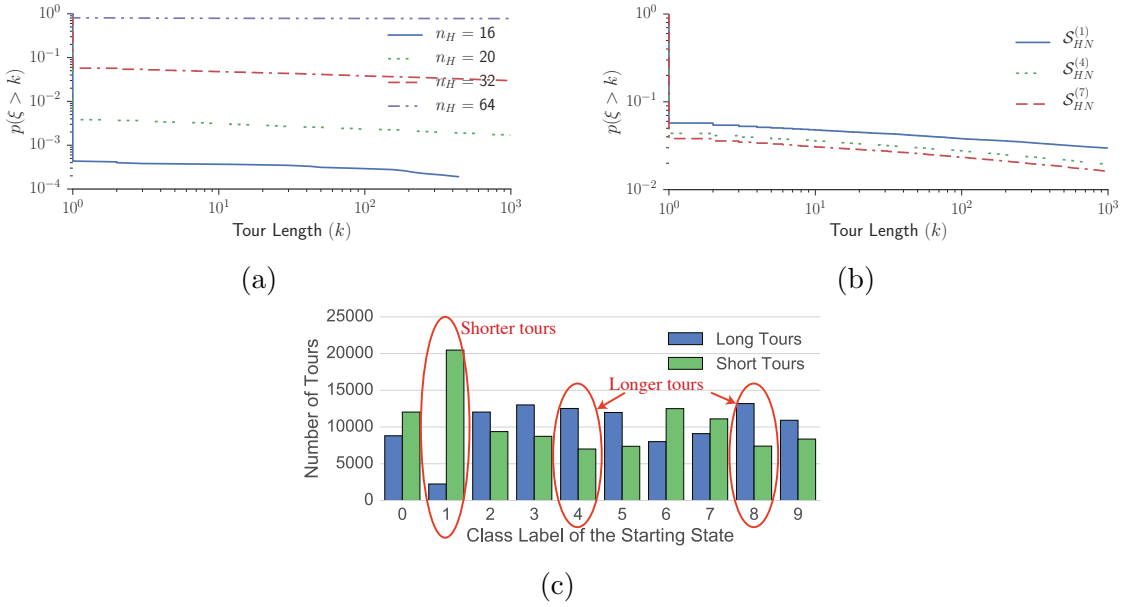


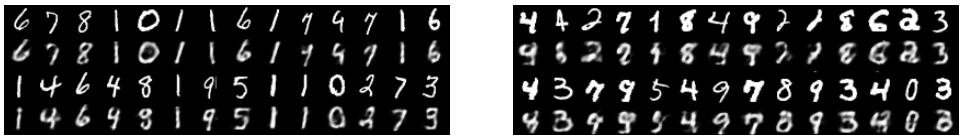
Figure 4.2. (a) Tour lengths CCDF for $n_H = \log_2 |H| \in \{16, 20, 32, 64\}$ for LVS-1; (b) Tour lengths CCDF variation for LVS-10 with $n_H = 32$, using larger Stopping Sets; (c) Comparison of frequencies of short and long tours starting from labeled states on a trained RBM

Distribution modes of the learned RBM. Overall, we may want to ask which digits (pictures) the model is learning to reproduce well.

Figure 4.2c shows the length of the tours split by the type of digit starting the tour. Note that the RBM seems to learn well digits that are more consistent across the training data (e.g., numbers one and six) and have more trouble with digits that have more variability (e.g., numbers four and eight).

As a visual inspection, Figure 4.3 shows the next visible states of extremely short (length = 1) and long (unfinished after 99,999 steps) tours, for $n_H = 32$ hidden neurons. There is a clear relation between long tours and not-so-common examples in the training data. The first and third rows show the training examples; the next row shows their first visible state after one Gibbs sampling step. Note that the majority of the training examples are easy-to-recognize digits, with still recognizable digits after sampling.

The second part of Figure 4.3 shows the training example and first visible samples of long tours. Note that the long tours tend to be digits that are either thicker (rarer in the data), or come in a not-so-standard shape than the digits in the first row. Note that in half of the examples, their first Gibbs samples are not too similar to the original digit. This shows that the model is having trouble learning these less standard digits. That long tours tend to start in odd-looking-examples, should help us better understand and avoid *fantasy particles* (visible states $\mathbf{v} \in V$ that are not characteristic of the dataset but have high probability nonetheless [101]).



(a) Short Tours (b) Long Tours

Figure 4.3. Visible states of tours for $n_H = 32$ neurons. The first and third rows of each image show the visible states from the training data, whereas the second and fourth show the next visible state obtained through Gibbs Sampling

Estimating the partition function. I use MCLV-K_{dyn} to estimate the partition function $Z(\mathbf{W}) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \mathrm{e}^{-E((\mathbf{v},\mathbf{h});\mathbf{W})}$ as specified in Corollary 2 using an RBM with $n_H = 32$, so that I can easily compute the true partition function for comparison. I note that computing $Z_{\mathcal{S}}(\mathbf{W})$ is fast as it is in the order of the number of training examples (as stated earlier).

Following Corollary 2, I estimate $Z(\mathbf{W})$ with $\hat{F}^{(K_{\text{dyn}},R)}(\mathbf{W}, f_1)$, with $f_1(x) = 1$. The average tour length in this example is estimated to be close to one (see Figure 4.1). Thus, $\hat{F}^{(K_{\text{dyn}},R)}(\mathbf{W}, f_1) \approx Z_{\mathcal{S}}(\mathbf{W}) = 1.46 \times 10^{100}$ in this example. In fact, $\hat{F}^{(K_{\text{dyn}},R)}(\mathbf{W}, f_1)$ and the true partition function $Z(\mathbf{W})$ report the same value up to nearly machine precision (10-th decimal place).

4.6 Summary

This chapter proposes a Las Vegas transformation of Markov Chain Monte Carlo (MCMC) for RBMs, denoted *Markov Chain Las Vegas* (MCLV). MCLV gives statistical guarantees in exchange for random running times. My empirical results show MCLV-K is a powerful tool to learn and understand RBMs, with a gradient estimator LVS-K that can better fit RBMs to the `MNIST` dataset than standard MCMC methods such as Contrastive Divergence (CD-K).

5. CONCLUSION

This thesis proposed multiple *MCMC* based algorithms to sample k node Connected Induced Subgraphs (k-*CIS*) with the aim of estimating *CIS* statistics of an input graph. The overarching theme being that these algorithms are parsimonious in terms of memory and their per-iterate running time.

First, in Chapter 2, I introduced two estimators of k-*CIS* function averages, *RTS* and *R-PSRW*, that push the state of the art of *CIS* statistic estimation to $k = 16$ for larger graphs and $k = 25$ for smaller graphs. By introducing regeneration and weighted random walks to the *FANMOD* tree, *RTS* is able to scale function estimation to an unprecedented range of k. *R-PSRW* proposes adding a novel randomized estimator to *PSRW* [18] that allows it to scale from $k = 8$ of the original algorithm to $k = 25$ over the same graphs. Our empirical results show that *RTS* is the most consistently accurate method across all graphs, followed by *R-PSRW*, followed by other baselines (as distant 3rd places for larger k).

Then, in Chapter 3, I propose the *Ripple* estimator that uses *sequentially stratified regenerations* to control the running time of a random walk tour-based *MCMC*. *Ripple* is consistent (w.r.t. the number of random walk tours). Its time and memory complexity for the *CIS* counting problem is linear in the number of patterns of interest and polynomial in the *CIS* size. *Ripple* can accurately estimate *CIS* counts with a smaller memory footprint compared to that of the state-of-the-art Motivo [26]. Beyond my specific application, *Ripple* provides a promising way to expand the sphere of influence of regenerative simulation in finite-state reversible *MCMC*.

Finally, Chapter 4 proposes a Las Vegas transformation of Markov Chain Monte Carlo (MCMC) for RBMs, denoted *Markov Chain Las Vegas* (MCLV). MCLV gives statistical guarantees in exchange for random running times. My empirical results show MCLV-K is a powerful tool to learn and understand RBMs, with a gradient estimator LVS-K that can better fit RBMs to the `MNIST` dataset than standard MCMC methods such as Contrastive Divergence (CD-K).

REFERENCES

[1] S. Jain and C. Seshadhri, "Provably and Efficiently Approximating Near-cliques using the Turán Shadow: PEANUTS," in *WWW 2020*, 2020, pp. 1966–1976.

[2] S. Jain and C. Seshadhri, "A Fast and Provable Method for Estimating Clique Counts Using TuráN's Theorem," in *WWW*, ser. WWW '17, Perth, Australia, 2017, pp. 441–449.

[3] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks.," *Science*, 2002.

[4] F. Hormozdiari, P. Berenbrink, N. Pržulj, and S. C. Sahinalp, "Not all scale-free networks are born equal: the role of the seed graph in PPI network evolution," *PLoS computational biology*, vol. 3, no. 7, e118, 2007.

[5] K. Faust, "A puzzle concerning triads in social networks: Graph constraints and the triad census," *Social Networks*, vol. 32, no. 3, pp. 221–233, 2010.

[6] R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, "Relational pooling for graph representations," in *International Conference on Machine Learning*, 2019.

[7] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient Graphlet Kernels for Large Graph Comparison," in *JMLR Workshop and Conf.*, MIT Press, 2009.

[8] B. Bevilacqua, Y. Zhou, and B. Ribeiro, "Size-Invariant Graph Representations for Graph Classification Extrapolations," *arXiv preprint arXiv:2103.05045*, 2021.

[9] L. Cotta, C. HC Teixeira, A. Swami, and B. Ribeiro, "Unsupervised Joint k-node Graph Representations with Compositional Energy-Based Models," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[10] B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. M. Bronstein, and H. Maron, "Equivariant subgraph aggregation networks," in *International Conference on Learning Representations*, 2021.

[11] L. Cotta, C. Morris, and B. Ribeiro, "Reconstruction for powerful graph representations," *Advances in Neural Information Processing Systems*, 2021.

[12] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *arXiv preprint arXiv:2006.09252*, 2020.

[13] M. Jerrum and K. Meeks, "The parameterised complexity of counting connected subgraphs and graph motifs," *Journal of Computer and System Sciences*, 2015.

[14] A. Pinar, C. Seshadhri, and V. Vishal, "Escape: Efficiently counting all 5-vertex subgraphs," in *WWW*, 2017, pp. 1431–1440.

[15] S. K. Bera, N. Pashanasangi, and C. Seshadhri, "Linear time subgraph counting, graph degeneracy, and the chasm at size six," *arXiv preprint arXiv:1911.05896*, 2019.

[16] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs," *Bioinformatics*, vol. 20, no. 11, pp. 1746–1758, 2004.

[17] S. Wernicke, "Efficient Detection of Network Motifs," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2006.

[18] P. Wang, J. C. S. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan, "Efficiently Estimating Motif Statistics of Large Networks," *ACM TKDD*, 2014.

[19] M. Bressan, S. Leucci, and A. Panconesi, "Motivo: Fast Motif Counting via Succinct Color Coding and Adaptive Sampling," *Proc. VLDB Endow.*, 2019.

[20] X. Chen and J. C. Lui, "Mining graphlet counts in online social networks," *ACM TKDD*, vol. 12, no. 4, pp. 1–38, 2018.

[21] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan, "Guise: Uniform sampling of graphlets for large graph analysis," in *2012 IEEE 12th Int. Conf. on Data Mining*, IEEE, 2012, pp. 91–100.

[22] A. P. Iyer, Z. Liu, X. Jin, S. Venkataraman, V. Braverman, and I. Stoica, "ASAP: Fast, Approximate Graph Pattern Mining at Scale," in *OSDI*, 2018, pp. 745–761.

[23] C. Yang, M. Lyu, Y. Li, Q. Zhao, and Y. Xu, "SSRW: a scalable algorithm for estimating graphlet statistics based on random walk," in *Int. Conf. on Database Systems for Advanced Applications*, Springer, 2018, pp. 272–288.

[24] P. Wang, J. Zhao, X. Zhang, Z. Li, J. Cheng, J. C. S. Lui, D. Towsley, J. Tao, and X. Guan, "MOSS-5: A Fast Method of Approximating Counts of 5-Node Graphlets in Large Graphs," *IEEE TKDE*, 2018.

[25] M. Bressan, F. Chierichetti, R. Kumar, S. Leucci, and A. Panconesi, "Motif Counting Beyond Five Nodes," *ACM TKDD*, vol. 12, no. 4, Apr. 2018.

[26] Bressan, Marco and Leucci, Stefano and Panconesi, Alessandro, "Faster motif counting via succinct color coding and adaptive sampling," *ACM Trans. Knowl. Discov. Data*, 2021.

[27] E. Nummelin, "A splitting technique for Harris recurrent Markov chains," *magazine for ″u r probability theory and related areas*, vol. 43, no. 4, pp. 309–318, 1978.

[28] P. Bremaud, *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*, ser. Texts in Applied Mathematics. Springer New York, 2001, ISBN: 9780387985091.

[29] B. Efron, *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

[30] J. Tukey, "Bias and confidence in not quite large samples," *Ann. Math. Statist.*, vol. 29, p. 614, 1958.

[31] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. on pattern analysis and machine intelligence*, 1984.

[32] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, 2002.

[33] G. E. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," in *Neural networks: Tricks of the trade*, Springer, 2012.

[34] P. Smolensky, *Information processing in dynamical systems: Foundations of harmony theory, Tech. Rep. DTIC*, 1986.

[35] K. Avrachenkov, B. Ribeiro, and J. K. Sreedharan, "Inference in OSNs via Lightweight Partial Crawls," in *SIGMETRICS*, 2016.

[36] M. Kakodkar and B. Ribeiro, "Regenerative Tree Sampling: Efficient Subgraph Statistics via Random Walks on Trees," *Under Submission*, 2022.

[37] C. H. Teixeira, M. Kakodkar, V. Dias, W. Meira, and B. Ribeiro, "Sequential stratified regeneration: MCMC for large state spaces with an application to subgraph count estimation," *Data Mining and Knowledge Discovery*, pp. 1–34, 2022.

[38] P. H. Savarese, M. Kakodkar, and B. Ribeiro, "From Monte Carlo to Las Vegas: Improving Restricted Boltzmann Machine Training Through Stopping Sets," in *AAAI*, 2018.

[39] C. Cooper, T. Radzik, and Y. Siantos, "Fast low-cost estimation of network properties using random walks," *Internet Mathematics*, 2016.

[40] L. Massoulié, E. Le Merrer, A.-M. Kermarrec, and A. Ganesh, "Peer counting and sampling in overlay networks: random walk methods," in *PODC*, 2006.

[41] R. Matsuno and A. Gionis, "Improved mixing time for k-subgraph sampling," in *Proc. of the 2020 SIAM Int. Conf. on Data Mining*, SIAM, 2020, pp. 568–576.

[42] M. Bressan, "Efficient and near-optimal algorithms for sampling connected subgraphs," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021.

[43] D. E. Knuth, "Estimating the efficiency of backtrack programs," *Mathematics of computation*, vol. 29, no. 129, pp. 122–136, 1975.

[44] D. Aldous and J. A. Fill, *Reversible Markov Chains and Random Walks on Graphs*, 2002.

[45] P. Brémaud, *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Springer, 2013.

[46] P. Mykland, L. Tierney, and B. Yu, "Regeneration in Markov chain samplers," *Journal of the American Statistical Association*, vol. 90, no. 429, pp. 233–241, 1995.

[47] J. Hopcroft and R. Tarjan, "Algorithm 447: efficient algorithms for graph manipulation," *Communications of the ACM*, vol. 16, no. 6, pp. 372–378, 1973.

[48] K. Paramonov, D. Shemetov, and J. Sharpnack, "Estimating Graphlet Statistics via Lifting," in *SIGKDD*, 2019, pp. 587–595.

[49] G. Han and H. Sethu, "Waddling random walk: Fast and accurate mining of motif statistics in large graphs," in *ICDM*, IEEE, 2016, pp. 181–190.

[50] C. H. Teixeira, L. Cotta, B. Ribeiro, and W. Meira, "Graph Pattern Mining and Learning through User-Defined Relations," in *ICDM*, IEEE, 2018, pp. 1266–1271.

[51] S. K. Bera, N. Pashanasangi, and C. Seshadhri, "Near-Linear Time Homomorphism Counting in Bounded Degeneracy Graphs: The Barrier of Long Induced Cycles," *CoRR*, 2020.

[52] N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, and S. C. Sahinalp, "Biomolecular network motif counting and discovery by color coding," *Bioinformatics*, vol. 24, no. 13, pp. i241–i249, 2008.

[53] Z. Zhao, M. Khan, V. A. Kumar, and M. V. Marathe, "Subgraph enumeration in large social contact networks using parallel color coding and streaming," in *ICPP*, IEEE, 2010, pp. 594–603.

[54] S. Assadi, M. Kapralov, and S. Khanna, "A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling," *arXiv preprint arXiv:1811.07780*, 2018.

[55] P. Ribeiro, P. Paredes, M. E. Silva, D. Aparicio, and F. Silva, "A Survey on Subgraph Counting: Concepts, Algorithms and Applications to Network Motifs and Graphlets," *arXiv preprint arXiv:1910.13011*, 2019.

[56] P. Welke, F. Seiffarth, M. Kamp, and S. Wrobel, "HOPS: Probabilistic subtree mining for small and large graphs," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[57] G. Preti, G. De Francisci Morales, and M. Riondato, "MaNIACS: Approximate Mining of Frequent Subgraph Patterns through Sampling," in *Proc. of the 27th ACM SIGKDD Conf. on Knowledge Discovery & Data Mining*, 2021.

[58] L. D. Stefani, E. Terolli, and E. Upfal, "Tiered Sampling: An Efficient Method for Counting Sparse Motifs in Massive Graph Streams," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 5, pp. 1–52, 2021.

[59] K. B. Athreya and P. Ney, "A new approach to the limit theory of recurrent Markov chains," *Trans. of the American Mathematical Society*, vol. 245, pp. 493–501, 1978.

[60] P. E. Jacob, J. O'Leary, and Y. F. Atchadé, "Unbiased Markov chain Monte Carlo methods with couplings," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 82, no. 3, pp. 543–600, 2020.

[61] J. G. Propp and D. B. Wilson, "Exact sampling with coupled Markov chains and applications to statistical mechanics," *Random Structures & Algorithms*, 1996.

[62] K. Avrachenkov, V. S. Borkar, A. Kadavankandy, and J. K. Sreedharan, "Revisiting random walk based sampling in networks: evasion of burn-in period and frequent regenerations," *Computational social networks*, vol. 5, no. 1, pp. 1–19, 2018.

[63] R. J. Walker, "An enumerative technique for a class of combinatorial problems," in *Combinatorial Analysis, Proc. Sympos. Appl. Math*, 1960, pp. 91–94.

[64] F. Olken and D. Rotem, "Random sampling from B+ trees," in *Proc 15th VLDB Conf, Amsterdam, The Netherlands*, 1989.

[65] P. R. Rosenbaum, "Sampling the leaves of a tree with equal probabilities," *Journal of the American Statistical Association*, vol. 88, no. 424, pp. 1455–1457, 1993.

[66] B. Cloteaux and L. A. Valentin, "Counting the leaves of trees," *Congr Numerantium*, vol. 207, pp. 129–139, 2011.

[67] P. C. Chen, "Heuristic sampling: A method for predicting the performance of tree searching programs," *SIAM Journal on Computing*, vol. 21, no. 2, pp. 295–315, 1992.

[68] P. W. Purdom, "Tree Size by Partial Backtracking," *SIAM J. Comput.*, vol. 7, pp. 481–491, 1978.

[69] K. Avrachenkov, B. Ribeiro, and J. K. Sreedharan, "Inference in OSNs via Lightweight Partial Crawls," in *ACM SIGMETRICS*, Antibes Juan-les-Pins, France, 2016, pp. 165–177, ISBN: 978-1-4503-4266-7.

[70] B. Ribeiro and D. Towsley, "On the estimation accuracy of degree distributions from graph sampling," in *CDC*, 2012.

[71] D. J. Wilkinson, "Parallel bayesian computation," *Statistics Textbooks and Monographs*, vol. 184, p. 477, 2006.

[72] W. Neiswanger, C. Wang, and E. Xing, "Asymptotically exact, embarrassingly parallel MCMC," *UAI*, 2014.

[73] J. S. Rosenthal, "Minorization conditions and convergence rates for Markov chain Monte Carlo," *Journal of the American Statistical Association*, 1995.

[74] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. on Mathematical Software (TOMS)*, 1985.

[75] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford Large Network Dataset Collection*, http://snap.stanford.edu/data, Jun. 2014.

[76] G. Liu and L. Wong, "Effective pruning techniques for mining quasi-cliques," in *ECML-PKDD*, 2008.

[77] R. M. Neal, "Annealed importance sampling," *Statistics and computing*, 2001.

[78] P. W. Glynn and C.-h. Rhee, "Exact estimation for Markov chain equilibrium expectations," *Journal of Applied Probability*, vol. 51, no. A, pp. 377–389, 2014.

[79] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, 1970.

[80] A. Sinclair, "Improved bounds for mixing rates of Markov chains and multicommodity flow," *Combinatorics, probability and Computing*, vol. 1, no. 4, 1992.

[81] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, 2006.

[82] R. Salakhutdinov and G. Hinton, "Deep Boltzmann Machines," *JMLR*, 2009.

[83] Y. Bengio and O. Delalleau, "Justifying and Generalizing Contrastive Divergence," *Neural Comput.*, 2009.

[84] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *Journal of Machine Learning Research*, 2010.

[85] M. A. Carreira-Perpiñán and G. E. Hinton, "On Contrastive Divergence Learning," in *AISTATS*, 2005.

[86] I. Sutskever and T. Tieleman, "On the Convergence Properties of Contrastive Divergence," in *AISTATS*, 2010.

[87] P. Buchholz, "Exact and ordinary lumpability in finite Markov chains," *Journal of Applied Probability*, 1994.

[88] W. J. Stewart, *Introduction to the numerical solutions of Markov chains*. Princeton Univ. Press, 1994.

[89] D. Aldous and J. Fill, *Reversible Markov chains and random walks on graphs*, 2002.

[90] R. G. Gallager, *Stochastic processes: theory for applications*. Cambridge University Press, 2013.

[91] J. R. Wilson, "The inspection paradox in renewal-reward processes," *Operations Research Letters*, 1983.

[92] R. M. Neal, "Estimating Ratios of Normalizing Constants using Linked Importance Sampling," *arXiv preprint math/0511216*, 2005.

[93] R. M. Neal, "Annealed Importance Sampling," *Statistics and Computing*, vol. 11, no. 2, pp. 125–139, 2001.

[94] R. Salakhutdinov and I. Murray, "On the quantitative analysis of deep belief networks," in *ICML*, 2008.

[95] A.-M. Lyne, M. Girolami, Y. Atchadé, H. Strathmann, D. Simpson, *et al.*, "On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods," *Statistical science*, 2015.

[96] G. Montúfar and J. Morton, "Discrete restricted Boltzmann machines.," *JMLR*, 2015.

[97] A. Yuille, "The Convergence of Contrastive Divergences," in *NIPS*, 2005.

[98] H. Schulz, A. Müller, and S. Behnke, "Investigating Convergence of Restricted Boltzmann Machine Learning," in *NIPS*, 2010.

[99] A. Fischer and C. Igel, "Training restricted Boltzmann machines: An introduction," *Pattern Recognit.*, 2014.

[100] D. B. Prats, E. Romero Merino, and F. M. Castrillejo, "Stopping Criteria in Contrastive Divergence: Alternatives to the Reconstruction Error," in *ICML*, 2014.

[101] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *ICML*, New York, New York, USA, 2008.

[102] P. M. Long and R. Servedio, "Restricted Boltzmann machines are hard to approximately evaluate or simulate," in *ICML*, 2010.

[103] J. Corcoran and R. Tweedie, "Perfect sampling from independent Metropolis-Hastings chains," *J. Stat. Plan. Inference*, 2002.

[104] J. A. Fill, "An interruptible algorithm for perfect sampling via Markov chains," in *STOC*, 1997.

[105] J. A. Fill, M. Machida, D. J. Murdoch, and J. S. Rosenthal, "Extension of Fill's perfect rejection sampling algorithm to general chains," *RSA*, 2000.

[106] J. G. Propp and D. B. Wilson, "How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph," *Journal of Algorithms*, 1998.

[107] D. B. Wilson, "Layered multishift coupling for use in perfect sampling algorithms (with a primer on CFTP)," *Monte Carlo Methods*, 2000.

[108] P. H. Baxendale, "Renewal theory and computable convergence rates for geometrically ergodic Markov chains," *Ann. Appl. Probab.*, 2005.

[109] W. R. Gilks, G. O. Roberts, and S. K. Sahu, "Adaptive Markov Chain Monte Carlo through Regeneration," *J. AMSTAT*, 1998.

[110] J. P. Hobert, G. L. Jones, B. Presnell, and J. S. Rosenthal, "On the applicability of regenerative simulation in Markov chain Monte Carlo," *Biometrika*, no. 4, 2002.

[111] G. O. Roberts and R. L. Tweedie, "Bounds on regeneration times and convergence rates for Markov chains," *Stoch. Process. their Appl.*, 1999.

[112] R. Bubley and M. Dyer, "Path coupling: A technique for proving rapid mixing in Markov chains," in *FOCS*, 1997.

[113] Y. Ollivier, "Ricci curvature of Markov chains on metric spaces," *J. Funct. Anal.*, 2009.

[114] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, 2017.

[115] C. J. Geyer, "Practical markov chain monte carlo," *Statistical science*, 1992.

[116] P. Diaconis and D. Stroock, "Geometric bounds for eigenvalues of Markov chains," *The Annals of Applied Probability*, pp. 36–61, 1991.

[117] C. Robert and G. Casella, *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

[118] P. Yao, L. Zheng, Z. Zeng, Y. Huang, C. Gui, X. Liao, H. Jin, and J. Xue, "A Locality-Aware Energy-Efficient Accelerator for Graph Mining Applications," in *MICRO*, 2020.

[119] S. P. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Springer, 2012.

[120] D. Stroock, *An introduction to Markov processes*. Springer, 1995.

A. DETAILS FOR CHAPTER 2

A.1 Additional Results

Table A.1 shows statistics of the graphs used in my evaluation, obtained from the *SNAP* [75] repository.

Environment. I run all the baselines and my own algorithms on a community High Performance Computing cluster which provides an AMD Rome CPU at 2Ghz with 64 cores and 128 GB memory per process. I ensure no memory swapping via `ulimit` and provide access to 200TB of disk space on a Lustre file system. All processes run with a wall clock time limit of 3½ hours and I produce partial estimates at the 1 hour mark, which are used to evaluate convergence as described later. All methods except *Motivo* (which is written in C++) were implemented by me in `Julia` [114] (in order to speed them up for a fair comparison). My code will be made available in the final version of the publication as a `Julia` library.

In Figure A.1 we compare *Ripple* with *R-PSRW* on the 5-*CIS* pattern distribution estimation task from Section 2.6.1 and show that *R-PSRW* outperforms *Ripple* in all graphs.

Table A.1. Input graphs that were used for evaluating my method *RTS* against state of the art baselines. Each dataset is available from SNAP [75] via the identifier in the parentheses. I also report the number of vertices, edges and maximum degree for each graph (largest connected component only).

Input Graph G_{IN}	$\lvert V_{\text{IN}}\rvert$	$\lvert E_{\text{IN}}\rvert$	$\lvert\Delta_{\text{IN}}\rvert$
Amazon (com-amazon)	335K	926K	549
DBLP (com-dblp)	317K	1.05M	343
Google (web-google)	856K	4.29M	6.33K
Patents (cit-patents)	3.76M	16.5M	793
Pokec (soc-pokec)	1.63M	22.3M	14.9K
LiveJournal (com-livejournal)	4M	34.7M	14.8K
Orkut (com-orkut)	3.07M	117M	33.3K

A.2 Proofs preliminaries

Before the main proofs, I state and prove some preliminaries.

Definition A.2.1. *The simple random walk on a weighted, undirected, simple graph* $G = (V, E)$*, with a positive, finite, and symmetric edge weighting function* $\mathbf{wt}(u, v)$ *is a time*

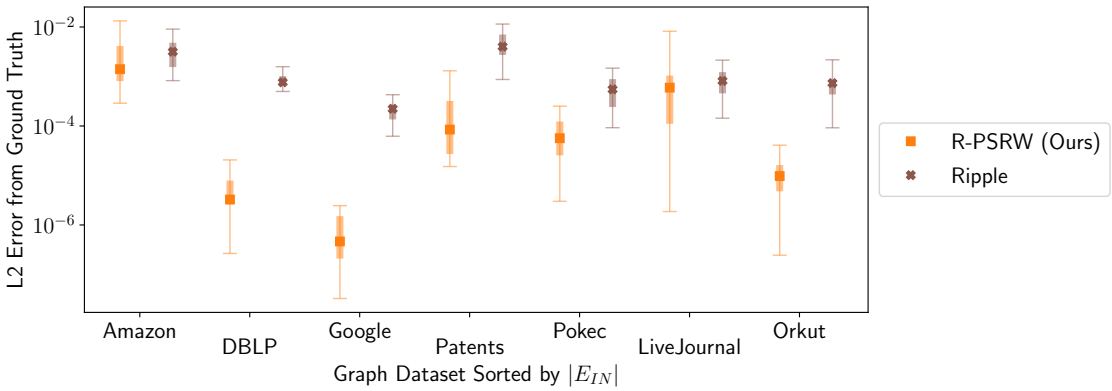


Figure A.1. *RTS* Vs *Ripple* Accuracy in the 5-*CIS* pattern distribution estimation task (Section 2.6.1). I compare *Ripple* with *R-PSRW* by running ripple with the Partition-Error-Bound parameter set to 0.003 as was done in *Ripple*'s evaluation [37]. I compare the error of *Ripple* estimates with that of *R-PSRW* estimates given the same wall clock time, and show box plots representing 10 runs. It is clear that *R-PSRW* converges to the ground truth faster on all the datasets I tested.

homogenous Markov chain, over the state space V*, with transition probability* $p_{SRW}(u,v) \propto \mathbf{wt}(u,v)$*, for all* $u,v \in V$ *and where* $\mathbf{wt}(u,v) = 0 \iff (u,v) \notin E$*.*

Lemma 5. *Let* $\mathbf{\Phi}$ *denote the simple random walk on the weighted graph* G*, defined in Definition A.2.1. Assuming* G *is connected,* $\mathbf{\Phi}$ *is irreducible. Further,* $\mathbf{\Phi}$ *is reversible and has a unique stationary distribution given by* $\boldsymbol{\pi}_{\mathbf{\Phi}}(u) = \frac{\deg(u)}{\mathrm{Vol}(G)}$*, for all* $u \in V$*, where* $\deg(u) = \sum_{v \in \mathbf{N}(u)} \mathbf{wt}(u,v)$ *and* $\mathrm{Vol}(G) = \sum_{u \in V} \deg(u)$*. Given an* RWT $X_1 = x_0, X_2, X_3, \ldots, X_\xi$*, beginning and ending at some* $x_0 \in V$ *as defined in Definition 2.3.2, and some bounded function* $g \colon V \to \mathbb{R}$*,*

$$\mathbb{E}\left[\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)\sum_{\mathrm{j}=1}^{\xi} g(X_{\mathrm{j}})\right] \overset{a.s.}{=} \mathbb{E}_{\boldsymbol{\pi}_{\mathbf{\Phi}}}[g] \triangleq \sum_{u \in V} \boldsymbol{\pi}_{\mathbf{\Phi}}(u) g(u). \tag{A.1}$$

Proof of Lemma 5. $\mathbf{\Phi}$ is irreducible when G is connected because connectivity implies that all states in V *communicate* with each other [45, Def 2.3.5]. The reversibilty and the specific

form of the stationary distribution are a direct consequence of the *detailed balance test* [45, Def 2.4.1]. That is for any $u, v \in V$

$$\boldsymbol{\pi}_{\boldsymbol{\Phi}}(u)p_{\boldsymbol{\Phi}}(u,v) = \frac{\deg(u)}{\mathrm{Vol}(G)}\frac{\mathbf{wt}(u,v)}{\deg(u)} = \boldsymbol{\pi}_{\boldsymbol{\Phi}}(v)p_{\boldsymbol{\Phi}}(v,u)\,.$$

The uniqueness of the stationary distribution is due to irreducibility.

By the Regenerative Cycle Theorem [45, Thm 2.5.11], *RWT*s are i.i.d. Further, let $Y = \sum_{\mathrm{j}=1}^{\xi} g(X_{\mathrm{j}})$. Since $\boldsymbol{\Phi}$ is finite and irreducible, it is positive recurrent [45, Thm 3.2.8]. Therefore *RWT*s have finite length, and since g is bounded, $\mathbb{E}[Y] < \infty$.

The Regenerative Cycle Theorem also stipulates that the concatenation of multiple *RWT*s is a sample path from $\boldsymbol{\Phi}$. Given a set of concatenated *RWT*s let t index this sequence. Let $N(t)$ be the counting process, which counts the number of *RWT*s completed up to time t. Then, by the Renewal Reward Theorem [45, Thm 3.3.6],

$$\frac{\mathbb{E}[Y]}{\mathbb{E}[\xi]} = \lim_{t\to\infty}\frac{\sum_{n=1}^{N(t)} Y_n}{t} = \lim_{t\to\infty}\frac{\sum_{n=1}^{N(t)} Y_n}{\sum_{n=1}^{N(t)}\xi_n}\,, \tag{A.2}$$

where Y_n and ξ_n denote the reward and length of the n-th *RWT* and the final equality is because

$$\lim_{t\to\infty}\frac{t}{\sum_{n=1}^{N(t)}\xi_n} = 1 + \lim_{t\to\infty}\frac{t - \sum_{n=1}^{N(t)}\xi_n}{\sum_{n=1}^{N(t)}\xi_n} = 1\,,$$

since positive recurrence guarantees that *RWT* lengths are finite.

By the Ergodic Theorem [45, Thm 3.3.2], Equation (A.2) yields $\mathbb{E}[Y] = \frac{\mathbb{E}_{\boldsymbol{\pi}_{\boldsymbol{\Phi}}}[g]}{\mathbb{E}[\xi]}$. The fact that $\mathbb{E}[\xi] = \frac{1}{\boldsymbol{\pi}_{\boldsymbol{\Phi}}(x_0)}$ [44, Lem 2.5] completes the proof. □

A.3 Proofs for *RTS*

Proof of Lemma 1. The probability of sampling the i-th *FANMOD* leaf $X_{k,\mathrm{i}}$, is given by $b(X_{k,\mathrm{i}})$. Consider the numerator of the Knuth-estimator under the limit $t \to \infty$,

$$\lim_{t\to\infty}\frac{1}{t}\sum_{\mathrm{i}=1}^{t}\frac{f(X_{k,\mathrm{i}})}{b(X_{\mathrm{i}})} \overset{a.s.}{=} \mathbb{E}_b\left[\frac{f(X_{k,.})}{b(X_.)}\right]\,,$$

where $\mathbb{E}_b$ is the expectation under the probability of sampling a *FANMOD* leaf using Knuth's algorithm and the almost sure convergence is due to the SLLN [45, Thm 1.4.10] and since f is bounded according to Definition 2.2.2. Moreover,

$$\mathbb{E}_b\left[\frac{f(X_{k,\cdot})}{b(X_\cdot)}\right] = \sum_{s\in\mathcal{S}^{(k)}} b(s)\frac{f(s)}{b(s)} = \sum_{s\in\mathcal{S}^{(k)}} f(s)\,.$$

The denominator similarly converges to an estimate of $|\mathcal{S}^{(k)}|$, which completes the proof. □

Proof of Proposition 2.3.1. Let s be the sampled *CIS* and $b(s)$ its sampling probability. Let $X_0, \ldots, X_k$, denote the samples from root to leaf, that led to the *CIS* $s \equiv X_k$. The highest sampling probability occurs, when the tree width (which is equal to the size of the extension set, EXT from Definition 2.2.4) is minimized for each ancestor, and vice versa. The tree width of X_0 is constant and equal to $|V_{\text{IN}}|$. The width at depth 1 can be as low as 1 and as high as Δ_{IN}, when all the neighbors of the vertex are added to EXT. The width at depth 2 can be as low as 1 and as high as $2\Delta_{\text{IN}}$, when the newly added vertex adds, Δ_{IN} new vertices to EXT, and so on. Expanding this argument recursively and taking the ratio completes the proof. □

Proof of Proposition 2.3.2. Consider the following procedure to sample transitions of the *TSMC* $\mathbf{\Phi}^{(k)}$, from a state x. With a probability $\frac{\alpha_{\text{DEPTH}(x)}}{\alpha_{\text{DEPTH}(x)}+|\mathcal{C}(x)|}$, we back-track to the parent $\mathcal{P}(x)$, and otherwise, choose a child at random. Note that all of the required information above, except $\mathcal{P}(x)$ is directly accessible via the child queries, $\mathcal{C}(\cdot)$ from Definition 2.2.4. Given the list of ancestors (at most k tree-nodes) of x, we can identify the parent tree-node is constant order time. This list can be kept updated during the walk by manipulating the head in constant time. □

Proposition A.3.1. *Given the* TSMC *from Definition 2.3.1 over the* FANMOD *tree* $\mathcal{T}^{(k)}$*, and the relative edge-weight parameters defined therein,* $\alpha_0 = 0, \alpha_1 \ldots, \alpha_k = 1$*. The* TSMC *is equivalent to the simple random walk over the weighted* FANMOD *tree from Definition A.2.1, where the edge weight between two tree-nodes* x *and* y*, where* x *is the parent of* y *is given by* $\mathbf{wt}(x,y) = \prod_{l=\text{DEPTH}(y)}^{k} \alpha_l$*. For the root-node* ρ*,* $\mathbf{wt}(\rho, y) = \prod_{l=1}^{k-1} \alpha_l$*, and edges incident on leaves have unit weight.*

Proof of proposition A.3.1. Consider the node x, defined in the statement above. For any given child $y \in \mathcal{C}(x)$, the edge weight is given by $\mathbf{wt}(x, y) = \prod_{l=\text{DEPTH}(x)+1}^{k} \alpha_l$. Similarly, $\mathbf{wt}(\mathcal{P}(x), x) = \prod_{l=\text{DEPTH}(x)}^{k} \alpha_l$. By Definition A.2.1, a simple random walk will choose a child uniformly and the parent edge with a probability $\alpha_{\text{DEPTH}(x)}$ times that of a child, making it equivalent to the *TSMC*. □

Proof of Lemma 2. By Proposition A.3.1 the *TSMC* is a simple random walk (by Definition A.2.1) on the *FANMOD* tree with weights given by Proposition A.3.1. Since $\mathcal{T}^{(k)}$ is connected the stated properties of the chain hold. By Lemma 5, for any k-depth leaf, $x \in \mathcal{L}$, the set of leaves at depth k, $\boldsymbol{\pi}_{\boldsymbol{\Phi}^{(k)}}(x) \propto \deg(x) = \alpha_k = 1$. □

Proof of Theorem 2.3.1. The numerator and denominator of Equation (2.4) are unbiased estimates of $\frac{1}{\boldsymbol{\pi}(\rho)} \sum_{s \in \mathcal{S}^{(k)}} f(s)$ and $\frac{1}{\boldsymbol{\pi}(\rho)} |\mathcal{S}^{(k)}|$, respectively due to Lemma 5, where $\boldsymbol{\pi}(\rho)$ is the stationary distribution at ρ. The ratio is however biased since it is a ratio of two random variables (estimates). This bias goes to 0 as the number of tours $\to \infty$ due to the almost sure convergence of the numerator and denominator. □

Proof of Proposition 2.3.3. The total number of tree-nodes at depth l in $\mathcal{T}^{(k)}$ is given by $\prod_{\text{i}=0}^{l-1} d_{\text{i}}$ For a tree-node x at the same depth l, the weight of any edge connecting x to $\mathcal{P}(x)$ has weight $\prod_{\text{j}=l}^{k} \alpha_{\text{j}}$. $\text{Vol}(\mathcal{T}^{(k)})$ is thus

$$\text{Vol}(\mathcal{T}^{(k)}) = \sum_{l=1}^{k} \prod_{\text{i}=0}^{l-1} d_{\text{i}} \prod_{\text{j}=l}^{k} \alpha_{\text{j}} = \sum_{l=1}^{k} w^{k-l} \prod_{\text{i}=0}^{k-1} d_{\text{i}} \,.$$

By Aldous and Fill [44, Lem 2.5], the expected length of an *RWT* is given by

$$\mathbb{E}[1/\xi] = \frac{\text{Vol}(\mathcal{T}^{(k)})}{\deg(\rho)} = \frac{\sum_{l=1}^{k-1} w^l \prod_{\text{i}=0}^{k-1} d_{\text{i}}}{w^{k-1} \prod_{\text{i}=0}^{k-1} d_{\text{i}}} = \sum_{l=1}^{k-1} w^{-l} \in \text{O}(1/w) \,.$$

The number of leaves sampled per *RWT*, using linearity of expectations and Aldous and Fill [44, Lem 2.6], is given by

$$\substack{\text{\# leaves} \\ \text{sampled}} = \frac{\deg(\text{leaf})}{\deg(\rho)} |\mathcal{L}| = \frac{1}{w^{k-1} \prod_{\text{i}=0}^{k-1} d_{\text{i}}} \prod_{\text{i}=0}^{k-1} d_{\text{i}} \in \text{O}(\frac{1}{w^{k-1}}) \,.$$

□

Proof of Proposition 2.3.4. Proposition 2.3.4 draws directly from the fact that each tree-node contains at most $|V_{\text{IN}}|$ vertices and requires as much memory. This is due to the fact that for a tree-node x, VSET_x and EXT_x as defined in Definition 2.2.3 cannot have repetitions and are mutually exclusive. Additionally, the k term is because I need to store at most k ancestors at any point of time. □

A.4 Proofs for *R-PSRW*

Proof of Proposition 2.4.1. A neighbor of a *CIS* x in $\mathcal{G}^{(k-1)}$ can be obtained by replacing a vertex in $V(x)$ with some other vertex not in $V(x)$. The number of potential neighbors of x in the *HON* is $k\,|\mathbf{N}(V(x))|$. For each potential neighbor, a graph traversal is required to check connectivity.

In Algorithm 1, Line 6 ensures that the output is indeed a $k-1$-*CIS*. If the main rejection step from Line 9 is not performed, the algorithm would sample the vertex to be added, v, with a probability which is proportional to the number of edges it shares with vertices in the original *CIS* $V(x)$. At Line 10, therefore we have v sampled uniformly from $\mathbf{N}(V(x))$ and u from $V(x)$. The connectivity check is not required if u is neither the anchor vertex a nor an articulation point of x, because a non articulating point, will not become articulating if a vertex is added, unless the anchor a is the only vertex in $V(x)$, that v is connected to. As such the rest of the procedure, trivially samples from $\mathbf{N}_{\mathcal{G}^{(k-1)}}(x)$, where $\mathbf{N}_{\mathcal{G}^{(k-1)}}(x)$ is the neighborhood of x in $\mathcal{G}^{(k-1)}$.

We require one traversal in Line 5, see Hopcroft and Tarjan [47]. The algorithm, after Line 10 proposes v from $\mathbf{N}(V(x))$ choices and a possible $u|v$ from $k-1$ choices. The procedure ends when a pair (u, v) which corresponds to a neighbor in the *HON* is chosen. The number of rejections is therefore geometrically distributed with parameter ${}^{|\mathbf{N}_{\mathcal{G}^{(k-1)}}(x)|}/_{(k-1)|\mathbf{N}(V(x))|}$, where $\mathbf{N}_{\mathcal{G}^{(k-1)}}(x)$ is the neighborhood of x in $\mathcal{G}^{(k-1)}$. The expected number of trials is thus $\mathrm{O}({}^{(k-1)|\mathbf{N}(V(x))|}/_{|\mathbf{N}_{\mathcal{G}^{(k-1)}}(x)|})$. Now note that $|\mathbf{N}_{\mathcal{G}^{(k-1)}}(x)| \geq |\mathbf{N}(V(x))|$, because for every $v \in \mathbf{N}(V(x))$, there exists at least one non-articulating $u \in V(x)$ which can be removed to yield a neighbor in $\mathbf{N}_{\mathcal{G}^{(k-1)}}(x)$. This is because assuming $k >= 3$, x with v added, is

connected and has at least 2 non-articulating points. As such, the expected number of trials is $O(k)$

Of these trials, a connectivity check (via a graph traversal) is only required if u is the anchor or articulation point. Such a u is chosen with probability at most $\frac{1+|AP_x|}{k}$, completing the proof. □

B. DETAILS FOR CHAPTER 3

B.1 Notation

The most important notations from Chapter 3 are summarized in Table B.1.

B.2 Proofs for Section 3.2

B.2.1 *MCMC* Estimates

Given a graph $\mathcal{G}$, when the $|\mathcal{E}|$ is unknown, the *MCMC* estimate of $\mu(\mathcal{E})/|\mathcal{E}|$ is given by:

Proposition B.2.1 (*MCMC* Estimate [31], [79], [115])**.** *When $\mathcal{G}$ from Definition 3.2.1 is connected, the random walk $\mathbf{\Phi}$ is reversible and positive recurrent with stationary distribution $\boldsymbol{\pi}_{\mathbf{\Phi}}(u) = \mathbf{d}(u)/2|\mathcal{E}|$. Then, the* MCMC *estimate*

$$\hat{\mu}_0\left((X_{\mathrm{i}})_{\mathrm{i}=1}^{t}\right) = \frac{1}{t-1}\sum_{\mathrm{i}=1}^{t-1} f(X_{\mathrm{i}}, X_{\mathrm{i}+1})\,,$$

computed using an arbitrarily started sample path $(X_{\mathrm{i}})_{\mathrm{i}=1}^{t}$ from $\mathbf{\Phi}$ is an asymptotically unbiased estimate of $\mu(\mathcal{E})/|\mathcal{E}|$. When $\mathcal{G}$ is non-bipartite, i.e., $\mathbf{\Phi}$ is aperiodic, and t is large, $\hat{\mu}_0$ converges to $\mu(\mathcal{E})/|\mathcal{E}|$ as

$$\left|\mathbb{E}[\hat{\mu}_0\left((X_{\mathrm{i}})_{\mathrm{i}=1}^{t}\right)] - \mu(\mathcal{E})/|\mathcal{E}|\right| \leq B\,\frac{C}{t\delta(\mathbf{\Phi})}\,,$$

where $\delta(\mathbf{\Phi})$ is the spectral gap of $\mathbf{\Phi}$ and $C \triangleq \sqrt{\frac{1-\boldsymbol{\pi}_{\mathbf{\Phi}}(X_1)}{\boldsymbol{\pi}_{\mathbf{\Phi}}(X_1)}}$ such that $f(\cdot) \leq B$.

Asymptotic unbiasedness. Because $\mathcal{G}$ is undirected, finite and connected, $\mathbf{\Phi}$ is a finite state space, irreducible, time-homogeneous Markov chain and is therefore positive recurrent [28, 3-Thm.3.3]. The reversibility and stationary distribution holds from the detailed balance test [28, 2-Cor.6.1] because

$$\boldsymbol{\pi}_{\mathbf{\Phi}}(u)\,p_{\mathbf{\Phi}}(u,v) = \boldsymbol{\pi}_{\mathbf{\Phi}}(v)\,p_{\mathbf{\Phi}}(v,u) = \frac{\mathbf{1}\{(u,v)\in\mathcal{E}\}}{2|\mathcal{E}|}\,.$$

Table B.1. Table of Notations

Symbol	Explanation
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	The graph where we have neighbor query access and whose edge sum is being computed.
$N(u)$, $\mathbf{d}(u)$	Neighborhood and degree of a vertex in $\mathcal{G}$ if no subscript is specified.
$\mu(\mathcal{E})$	The sum over edges in $\mathcal{E}$ (or some subset) of some function f.
$\mathbf{\Phi}$, $p_{\mathbf{\Phi}}(u, v)$, $\boldsymbol{\pi}_{\mathbf{\Phi}}(u)$	The random walker on $\mathcal{G}$, its transition probability and stationary distribution.
$\mathbf{X}$, ξ, $\mathcal{T}$	an *RWT* (tour), its length and a set of m *RWT*s.
$\hat{\mu}_*(\mathcal{T}; f, \mathcal{G})$	an *RWT* Estimate of $\mu(\mathcal{E})$.
δ	The spectral gap of the transition probability matrix of a chain.
$\hat{\sigma}(\cdot)^2$	Empirical variance of an *RWT* Estimate.
ζ_I, $\mathcal{G}_I$	Collapsed state and graph obtained by collapsing $I \subset \mathcal{V}$.
$q < r < t$	Strata ids always used in the same order $1 \leq q < r < t \leq R$.
$\rho \colon \mathcal{V} \to \{1, \ldots, R\}$	Stratification function.
$\mathcal{I}_r, \mathcal{J}_r$	r-th vertex and edge stratum.
$\mathcal{G}_r = (\mathcal{V}_r, \mathcal{E}_r)$	r-th graph stratum.
ζ_r, $\mathcal{T}_r$	Supernode in each stratum and a set of m_r perfectly sampled tours from ζ_r.
$\mathbf{d}(\zeta_r)$, $p_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$	Degree and transition probability out of the supernode.
$\widehat{\mathbf{d}}(\zeta_r)$, $\widehat{p}_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$	Estimated degree and transition probability out of the supernode.
$\mathcal{T}_q^\dagger$	m_q *RWT*s samples using supernode estimates.
$\widehat{\beta}_{q,r}$	The estimate of the number of edges between $\mathcal{I}_q$ and $\mathcal{I}_r$.
$\widehat{\mathbf{U}}_{q,r}$	Multiset of states visited by $\mathcal{T}_q^\dagger$ that lie in $\mathcal{I}_r$.
$\hat{\mu}_{\text{Ripple}}$, $\hat{\mu}\left(\mathcal{T}_{2:r}^\dagger; f\right)$	Overall and per-stratum *Ripple* estimate.
δ_r, ν_r, λ_r	Spectral gap and the errors in the supernode estimates in the r-th stratum.
$G = (V, E, L)$	The labelled input graph in which we want to count subgraphs.
$G(V')$	Subgraph induced by V' in G.
$\mathcal{H}$	Nonequivalent (non-isomorphic) patterns of interest.
$\mathcal{C}^{(k)} = (\mathcal{C}_H^{(k)})_{H \in \mathcal{H}}$	The subgraph pattern count vector.
$\mathcal{G}^{(k)} = (\mathcal{V}^{(k)}, \mathcal{E}^{(k)})$	The *k-HON* that provides neighborhood query access and is used to count subgraphs.
$\gamma(u, v)$	Number of edges in $\mathcal{E}^{(k-1)}$ that represent the same subgraph as (u, v).
$\textsc{dist}(u)$	Shortest path distance from $u \in V$ to any seed vertex in $V(\mathcal{I}_1)$.
V^*	The largest connected subset of $V(s)$ that constitutes an intersection between s and $\ddot{s} \in \mathcal{I}_1$.
M	Reservoir size.
Δ_G, D_G	Maximum degree in and diameter of G.
$\mathcal{A}_s$	Articulation points in s.
ϵ	Per-stratum error bound used to control tour count.

The ergodic theorem [28, 3-Cor.4.1] then applies because f is bounded and we have

$$\lim_{t\to\infty} \frac{1}{t-1} \sum_{\mathrm{i}=1}^{t-1} f(X_\mathrm{i}, X_{\mathrm{i}+1}) = \sum_{(u,v)\in\mathcal{V}\times\mathcal{V}} \boldsymbol{\pi}_{\mathbf{\Phi}}(u)\, p_{\mathbf{\Phi}}(u, v) f(u, v) = \frac{\mu(\mathcal{E})}{|\mathcal{E}|}\,.$$

□

Bias. Let the i-step transition probability of $\mathbf{\Phi}$ be given by $p^{\mathrm{i}}_{\mathbf{\Phi}}(u,v)$. The bias at the i-th step is given by

$$\begin{aligned}
\mathrm{BIAS}_{\mathrm{i}} &= \left|\mathbb{E}\left[f(X_{\mathrm{i}}, X_{\mathrm{i}+1})\right] - \sum_{(u,v)\in\mathcal{V}\times\mathcal{V}} \boldsymbol{\pi}_{\mathbf{\Phi}}(u)\, p_{\mathbf{\Phi}}(u,v) f(u,v)\right| \\
&= \left|\sum_{(u,v)\in\mathcal{V}\times\mathcal{V}} p^{\mathrm{i}}_{\mathbf{\Phi}}(X_1,u)\, p_{\mathbf{\Phi}}(u,v) f(u,v) - \sum_{(u,v)\in\mathcal{V}\times\mathcal{V}} \boldsymbol{\pi}_{\mathbf{\Phi}}(u)\, p_{\mathbf{\Phi}}(u,v) f(u,v)\right| \\
&\leq B\left|\sum_{u\in\mathcal{V}} p^{\mathrm{i}}_{\mathbf{\Phi}}(X_1,u) \sum_{v\in\mathcal{V}} p_{\mathbf{\Phi}}(u,v) - \sum_{u\in\mathcal{V}} \boldsymbol{\pi}_{\mathbf{\Phi}}(u) \sum_{v\in\mathcal{V}} p_{\mathbf{\Phi}}(u,v)\right| \\
&\leq B\left|\sum_{u\in\mathcal{V}} p^{\mathrm{i}}_{\mathbf{\Phi}}(X_1,u) - \sum_{u\in\mathcal{V}} \boldsymbol{\pi}_{\mathbf{\Phi}}(u)\right| \leq B \sum_{u\in\mathcal{V}} \left|p^{\mathrm{i}}_{\mathbf{\Phi}}(X_1,u) - \boldsymbol{\pi}_{\mathbf{\Phi}}(u)\right|,
\end{aligned}$$

where $f(\cdot) \leq B$, and the final inequality is due to Jensen's inequality. From [116, Prop-3],

$$\mathrm{BIAS}_{\mathrm{i}} \leq B\sqrt{\frac{1-\boldsymbol{\pi}_{\mathbf{\Phi}}(X_1)}{\boldsymbol{\pi}_{\mathbf{\Phi}}(X_1)}}\beta^{\mathrm{i}}_*,$$

where $\beta_* = 1-\delta(\mathbf{\Phi})$ is the SLEM of $\mathbf{\Phi}$. Because of Jensen's inequality and by summing a GP,

$$\left|\mathbb{E}[\hat{\mu}_0\left((X_{\mathrm{i}})_{\mathrm{i}=1}^{t}\right)] - \frac{\mu(\mathcal{E})}{|\mathcal{E}|}\right| \leq \frac{1}{t-1}\sum_{\mathrm{i}=1}^{t-1}\mathrm{BIAS}_{\mathrm{i}} \leq \frac{B}{t-1}\sqrt{\frac{1-\boldsymbol{\pi}_{\mathbf{\Phi}}(X_1)}{\boldsymbol{\pi}_{\mathbf{\Phi}}(X_1)}}\frac{1-\beta_*^t}{1-\beta_*}.$$

Assuming that $\beta_*^t \approx 0$ and $t-1 \approx t$ when t is sufficiently large completes the proof. □

Lemma 6 ([69])**.** *Let $\mathbf{\Phi}$ be a finite state space, irreducible, time-homogeneous Markov chain, and let ξ denote the return time of* RWT *started from some $x_0 \in \mathcal{S}$ as defined in Definition 3.2.2. If $\mathbf{\Phi}$ is reversible, then*

$$\mathbb{E}\left[\xi^2\right] \leq \frac{3}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)^2\delta(\mathbf{\Phi})}, \tag{B.1}$$

where $\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)$ is the stationary distribution of x_0, and $\delta(\mathbf{\Phi})$ is the spectral gap of $\mathbf{\Phi}$. When $\mathbf{\Phi}$ is not reversible, the second moment of return times is given by Equation (B.2).

Proof. Using [44, Eq 2.21], we have

$$\mathbb{E}\left[\xi^2\right] = \frac{1 + 2\mathbb{E}_{\pi_{\mathbf{\Phi}}}(T_{x_0})}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)}\,, \tag{B.2}$$

where $\mathbb{E}_{\pi_{\mathbf{\Phi}}}(T_{x_0})$ is the expected hitting time of x_0 from the steady state. Combining [44, Lemma 2.11 & Eq 3.41] and accounting for continuization yields

$$\mathbb{E}_{\pi_{\mathbf{\Phi}}}(T_{x_0}) \leq \frac{1}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)\delta(\mathbf{\Phi})}\,. \text{ and therefore, } \mathbb{E}\left[\xi^2\right] \leq \frac{1 + \frac{2}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)\delta(\mathbf{\Phi})}}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)} < \frac{3}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)^2\delta(\mathbf{\Phi})}\,,$$

because $\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)$ and $\delta(\mathbf{\Phi})$ lie in the interval $(0, 1)$. □

Proposition B.2.2. *Given a positive recurrent Markov chain $\mathbf{\Phi}$ over state space $\mathcal{S}$ and a set of m* RWT*s $\mathcal{T}$ and assuming an arbitrary ordering over $\mathcal{T}$, where $\mathbf{X}^{(\mathrm{i})}$ is the* i*th* RWT *in $\mathcal{T}$, $\mathbf{X}^{(\mathrm{i})}$ and $|\mathbf{X}^{(\mathrm{i})}|$ are i.i.d. processes such that $\mathbb{E}[|\mathbf{X}^{(\mathrm{i})}|] < \infty$, and when the tours are stitched together as defined next, the sample path is governed by $\mathbf{\Phi}$. For $t \geq 1$, define $\Phi_t = X^{N_t}_{t-R_{N_t}}$, where $R_{\mathrm{i}} = \sum_{\mathrm{i}'=1}^{\mathrm{i}-1} |\mathbf{X}^{\mathrm{i}}|$ when $\mathrm{i} > 1$ and $R_1 = 0$ and $N_t = \max\{\mathrm{i}\colon R_{\mathrm{i}} < t\}$.*

Proof. R_{i} is a sequence of stopping times. Therefore, the strong Markov property [28, 2-Thm.7.1] states that sample paths before and after R_{i} are independent and are governed by $\mathbf{\Phi}$. Because $\mathbf{\Phi}$ is positive recurrent and x_0 is visited i.o., the regenerative cycle theorem [28, 2-Thm.7.4] states that these trajectories are identically distributed and are equivalent to the tours $\mathcal{T}$ sampled according to Definition 3.2.2. $\mathbb{E}[|\mathbf{X}^{(}\mathrm{i})|] < \infty$ due to positive recurrence. □

B.2.2 Proof of Lemma 3

Unbiasedness and Consistency. Because $\mathcal{G}$ is connected, $\mathbf{\Phi}$ is positive recurrent with steady state $\boldsymbol{\pi}_{\mathbf{\Phi}}(u) \propto \mathbf{d}(u)$ due to Proposition B.2.1. Consider the reward process $F^{(\mathrm{i})} = \sum_{\mathrm{j}=1}^{|\mathbf{X}^{(\mathrm{i})}|} f(X^{(\mathrm{i})}_{\mathrm{j}}, X^{(\mathrm{i})}_{\mathrm{j}+1})$, $\mathrm{i} \geq 1$. From Proposition B.2.2, $F^{(\mathrm{i})}$ and $|\mathbf{X}^{(\mathrm{i})}|$ are i.i.d. sequences with finite first moments, because $F^{(\mathrm{i})} \leq B|\mathbf{X}^{(\mathrm{i})}|$. Let N_t and R_{i} be as defined in Proposition B.2.2.

Therefore, from the renewal reward theorem [28, 3-Thm.4.2], we have

$$\frac{\mathbb{E}[F^{(\mathrm{i})}]}{\mathbb{E}[|\mathbf{X}^{(\mathrm{i})}|]} = \lim_{t\to\infty} \frac{\sum_{\mathrm{i}=1}^{N_t} F^{(\mathrm{i})}}{t} = \lim_{t\to\infty} \frac{\sum_{\mathrm{i}=1}^{N_t} F^{(\mathrm{i})}}{R_{N_t}} \cdot \frac{R_{N_t}}{t} = \frac{\sum_{\mathrm{i}=1}^{N_t} F^{(\mathrm{i})}}{R_{N_t}},$$

where the final equality holds because $\lim_{t\to\infty} \frac{R_{N_t}}{t} = 1 - \lim_{t\to\infty} \frac{t-R_{N_t}}{t}$, and $\lim_{t\to\infty} \frac{t-R_{N_t}}{t}$ converges to 0 as $t \to \infty$ because $|\mathbf{X}^{(\cdot)}| < \infty$ w.p. 1 because $\mathbf{\Phi}$ is positive recurrent.

From Proposition B.2.2 and the definition of $F^{(\mathrm{i})}$, $\sum_{\mathrm{i}=1}^{N_t} F^{(\mathrm{i})} = \sum_{\mathrm{j}=1}^{R_{N_t}} f(\Phi_{\mathrm{j}}, \Phi_{\mathrm{j}+1})$, and because f and $\boldsymbol{\pi}_{\mathbf{\Phi}}$ are bounded, we have from the ergodic theorem [28, 3-Cor.4.1],

$$\frac{\mathbb{E}[F^{(\mathrm{i})}]}{\mathbb{E}[|\mathbf{X}^{(\mathrm{i})}|]} = \lim_{t\to\infty} \frac{\sum_{\mathrm{j}=1}^{R_{N_t}} f(\Phi_{\mathrm{j}}, \Phi_{\mathrm{j}+1})}{R_{N_t}} \overset{a.s.}{=} \sum_{(u,v)\in\mathcal{V}\times\mathcal{V}} \boldsymbol{\pi}_{\mathbf{\Phi}}(u) p_{\mathbf{\Phi}}(u,v) g(u,v) = \frac{2\mu(\mathcal{E})}{2|\mathcal{E}|}.$$

From Kac's formula [44, Cor.2.24], $1/\mathbb{E}[|\mathbf{X}^{(\mathrm{i})}|] = \boldsymbol{\pi}_{\mathbf{\Phi}}(x_0) = \frac{\mathbf{d}(x_0)}{2|\mathcal{E}|}$, and

$$\mathbb{E}\left[\frac{\mathbf{d}(x_0)}{2} F^{(\mathrm{i})}\right] \overset{a.s.}{=} \mu(\mathcal{E}).$$

$\hat{\mu}_*(\mathcal{T}; f, \mathcal{G})$ is unbiased by linearity of expectations on the summation over $\mathcal{T}$, and consistency is a consequence of Kolmogorov's SLLN [28, 1-Thm.8.3]. □

Running Time. From Kac's formula [44, Cor.2.24], $\mathbb{E}[|\mathbf{X}^{(\mathrm{i})}|] = \frac{2|\mathcal{E}|}{\mathbf{d}(x_0)}$. From Proposition B.2.2, tours can be sampled independently and thus parallelly. All cores will sample an equal number of tours in expectation, yielding the running time bound. □

Variance. Because $f(\cdot) < B$, and tours are i.i.d., the variance is given by

$$\mathrm{Var}\left(\hat{\mu}_*(\mathcal{T})\right) = \mathrm{Var}\left(\frac{\mathbf{d}(x_0)}{2m} \sum_{\mathbf{X}\in\mathcal{T}} \sum_{\mathrm{j}=1}^{|\mathbf{X}|} f(X_{\mathrm{j}}, X_{\mathrm{j}+1})\right) \leq \frac{\mathbf{d}(x_0)^2 B^2}{4m} \mathrm{Var}\left(|\mathbf{X}|\right).$$

From Lemma 6 and Kac's formula [44, Cor.2.24], $\mathrm{Var}\left(|\mathbf{X}|\right)$ is given by

$$\mathrm{Var}\left(|\mathbf{X}|\right) \leq \frac{3}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)^2 \delta(\mathbf{\Phi})} - \frac{1}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)^2} \leq \frac{3}{\boldsymbol{\pi}_{\mathbf{\Phi}}(x_0)^2 \delta(\mathbf{\Phi})} = \frac{12|\mathcal{E}|^2}{\mathbf{d}(x_0)^2 \delta(\mathbf{\Phi})}.$$

□

B.3 Proofs for Section 3.3

Assumption 2. *For each $\mathcal{G}_r$, $1 < r \leq R$ from Definition 3.3.2, assume $\mathbf{d}(\zeta_r)$ is known and that $p_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$ can be sampled from.*

Proposition B.3.1 (*RWT*s in $\mathbf{\Phi}_r$)**.** *Under Assumption 2, given access only to the original chain $\mathbf{\Phi}$ and stratifying function ρ, let $\mathbf{\Phi}_r$ be the random walk in the graph stratum $\mathcal{G}_r$ from Definition 3.3.2. To sample an* RWT $(X_{\mathrm{i}})_{\mathrm{i}=1}^{\xi}$ *over $\mathbf{\Phi}_r$ from the supernode ζ_r, we set $X_1 = \zeta_r$, sample $X_2 \sim p_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$, and then, until $\rho(X_{\xi+1}) < r$, we sample*

$$X_{\mathrm{i}+1} \sim \textsc{unif}\left(\mathbf{N}_{G_r}\left(X_{\mathrm{i}}\right)\right) \equiv \begin{cases} \textsc{unif}(\mathbf{N}_{\mathcal{G}}(X_{\mathrm{i}})) & \text{if } \rho(X_{\mathrm{i}}) = r \\ \textsc{unif}(\mathbf{N}_{\mathcal{G}}(X_{\mathrm{i}}) \cap \mathcal{I}_r) & \text{if } \rho(X_{\mathrm{i}}) > r \end{cases}.$$

Proof. The proof is a direct consequence of Definition 3.3.2 and Definition 3.2.1. □

Proposition B.3.2 (Perfectly Stratified Estimate)**.** *Under Assumption 2, given the* EPS *(Definition 3.3.3) stratum $\mathcal{G}_r$ (Definition 3.3.2), bounded $f\colon \mathcal{E} \to \mathbb{R}$ and a set of m* RWT*s $\mathcal{T}_r$ over $\mathbf{\Phi}_r$ from ζ_r from Proposition B.3.1, the per stratum estimate is given by*

$$\hat{\mu}(\mathcal{T}_r; f, \mathcal{G}_r) = \frac{\mathbf{d}(\zeta_r)}{2m} \sum_{\mathbf{X} \in \mathcal{T}_r} \sum_{\mathrm{j}=2}^{|\mathbf{X}|-1} f(X_{\mathrm{j}}, X_{\mathrm{j}+1}), \tag{B.3}$$

where X_{j} is the jth state visited in the RWT *$\mathbf{X} \in \mathcal{T}_r$. For all $r > 1$, $\hat{\mu}(\mathcal{T}_r; f, \mathcal{G}_r)$ is an unbiased and consistent estimator of $\mu(\mathcal{J}_r) = \sum_{(u,v)\in\mathcal{J}_r} f(u,v)$, where $\mathcal{J}_r$ is the r-th edge stratum defined in Definition 3.3.1.*

Proof. Define $f'\colon \mathcal{E}_r \to \mathbb{R}$ as $f'(u,v) \triangleq \mathbf{1}\{u, v \neq \zeta_r\} f(u,v)$. By Definition 3.2.2, in each *RWT* $\mathbf{X} \in \mathcal{T}_r$, $f'(X_1, X_2) = f'(X_{|\mathbf{X}|}, X_{|\mathbf{X}|+1}) = 0$, and therefore, $\hat{\mu}(\mathcal{T}_r; f, \mathcal{G}_r) = \hat{\mu}_*(\mathcal{T}; f', \mathcal{G}_r)$, where $\hat{\mu}_*$ is the *RWT* Estimate from Lemma 3. Moreover, because $\mathcal{G}_r$ is connected,

$$\mathbb{E}\left[\hat{\mu}(\mathcal{T}_r; f, \mathcal{G}_r)\right] = \mathbb{E}\left[\hat{\mu}_*(\mathcal{T}; f', \mathcal{G}_r)\right] = \sum_{(u,v)\in\mathcal{E}_r} f'(u,v) = \sum_{(u,v)\in\mathcal{J}_r} f(u,v),$$

where the final equality holds because $\mathcal{E}_r$ is the union of $\mathcal{J}_r$ and edges incident on the supernode. Consistency is also due to Lemma 3. □

B.3.1 Proof of Proposition 3.3.1

Proof. Proposition 3.3.1 (a) is necessary because when Proposition 3.3.1 (a) does not hold, there exists a component such that the minimum value of ρ in that component is $\ddot{r} > 0$ such that in $\mathcal{G}_{\ddot{r}}$ (Definition 3.3.2), and the supernode $\zeta_{\ddot{r}}$ will be disconnected from all vertices. If Proposition 3.3.1 (b) is violated, a vertex $\ddot{u}$ exists that is disconnected in $\mathcal{G}_{\rho(\ddot{u})}$, and if Proposition 3.3.1 (c) is violated, the supernode is disconnected. Finally, it is easily seen that these conditions sufficiently guarantee that each stratum is connected, and the stratification is an *EPS*. □

B.3.2 Proof of Theorem 3.3.1

We begin by defining the multi-set containing the end points of edges between vertex strata.

Definition B.3.1. *Given $\mathcal{G}$ stratified into R strata, $\forall 1 \leq q < t \leq R$ define border multi-sets as*

$$\mathcal{B}_{q,t} \triangleq \{v \,\forall (u,v) \in \mathcal{E} \colon u \in \mathcal{I}_q \,\text{and}\, v \in \mathcal{I}_t\}\,.$$

The degree of the supernode in $\mathcal{G}_r$ (Definition 3.3.2) is then given by $\mathbf{d}(\zeta_r) = \sum_{q=1}^{r-1} |\mathcal{B}_{q,r}|$, and transitions out of ζ_r can be sampled by sampling $q \in \{1, \ldots, r-1\}$ w.p. $\propto |\mathcal{B}_{q,r}|$ and then by uniformly sampling from $\mathcal{B}_{q,r}$.

Proposition B.3.3. *Given the setting in Definitions 3.3.5 and 3.3.6, for all $1 \leq r < t \leq R$,*

$$\lim_{|\mathcal{T}_2^\dagger| \to \infty} \ldots \lim_{|\mathcal{T}_r^\dagger| \to \infty} \widehat{\beta}_{r,t} \overset{a.s.}{=} |\mathcal{B}_{r,t}|\,, \tag{B.4}$$

$$\lim_{|\mathcal{T}_2^\dagger| \to \infty} \ldots \lim_{|\mathcal{T}_r^\dagger| \to \infty} \widehat{\mathbf{U}}_{r,t} \sim \textsc{unif}(\mathcal{B}_{r,t})\,, \tag{B.5}$$

$$\lim_{|\mathcal{T}_2^\dagger| \to \infty} \ldots \lim_{|\mathcal{T}_r^\dagger| \to \infty} p_{\widehat{\mathbf{\Phi}}_r}(\mathbf{X}) = p_{\mathbf{\Phi}_r}(\mathbf{X})\,, \quad \forall \mathbf{X} \in \mathcal{T}_r^\dagger\,. \tag{B.6}$$

i.e., each tour in $\mathcal{T}_r^\dagger$ is perfectly sampled from $\mathbf{\Phi}_r$.

By Strong Induction. The base case for $r = 1$ holds by the base case in Definition 3.3.5. Now assume that Proposition B.3.3 holds for all strata up to and including $r - 1$. Because of the inductive claim and by Definition B.3.1,

$$\lim_{|\mathcal{T}_2^\dagger|\to\infty} \dots \lim_{|\mathcal{T}_{r-1}^\dagger|\to\infty} \widehat{\mathbf{d}}(\zeta_r) = \sum_{q=1}^{r-1} \widehat{\beta}_{q,r} \overset{a.s.}{=} \sum_{q=1}^{r-1} |\mathcal{B}_{q,r}| = \mathbf{d}(\zeta_r)\,,$$

$$\text{and similarly,} \quad \lim_{|\mathcal{T}_2^\dagger|\to\infty} \dots \lim_{|\mathcal{T}_{r-1}^\dagger|\to\infty} \widehat{p}_{\mathbf{\Phi}_r}(\zeta_r, \cdot) \equiv p_{\mathbf{\Phi}_r}(\zeta_r, \cdot)$$

because the inductive claim makes the procedure of sampling transitions out of ζ_r in Definition 3.3.5 equivalent to Definition B.3.1. Equation (B.6) holds because transition probabilities at all states other than ζ_r are equivalent in $\mathbf{\Phi}_r$ and $\widehat{\mathbf{\Phi}}_r$ according to Definition 3.3.4. Now recall that

$$\widehat{\beta}_{r,t} = \frac{\widehat{\mathbf{d}}(\zeta_r)}{|\mathcal{T}_r^\dagger|} \sum_{\mathbf{X}\in\mathcal{T}_r^\dagger} \sum_{\mathrm{j}=2}^{|\mathbf{X}|} \mathbf{1}\{\rho(X_\mathrm{j}) = t\}\,.$$

Because $\widehat{\mathbf{d}}(\zeta_r) = \mathbf{d}(\zeta_r)$ and the tours are sampled perfectly,

$$\lim_{|\mathcal{T}_2^\dagger|\to\infty} \dots \lim_{|\mathcal{T}_{r-1}^\dagger|\to\infty} \widehat{\beta}_{r,t} = \hat{\mu}_* \left(\mathcal{T}_r^\dagger; f'\right)\,,$$

where $f'(u, v) = \mathbf{1}\{\rho(v) = t\}$ and $\hat{\mu}_*$ is from Lemma 3, from which we also use the consistency guarantee to show that under an *EPS*, Equation (B.4) holds as

$$\lim_{|\mathcal{T}_2^\dagger|\to\infty} \dots \lim_{|\mathcal{T}_r^\dagger|\to\infty} \widehat{\beta}_{r,t} \overset{a.s.}{=} \sum_{(u,v)\in\mathcal{E}_r} f'(u, v) = |\mathcal{B}_{r,t}|\,.$$

Because of Proposition B.2.2, concatenating tours $\mathbf{X} \in \mathcal{T}_q^\dagger$ yields a sample path from $\mathbf{\Phi}_r$, and these samples are distributed according to $\boldsymbol{\pi}_{\mathbf{\Phi}_r}$ as $|\mathcal{T}_{r'}^\dagger| \to \infty$, $r' \le r$. Therefore,

$$\lim_{|\mathcal{T}_2^\dagger|\to\infty} \dots \lim_{|\mathcal{T}_r^\dagger|\to\infty} \uplus_{\mathbf{X}\in\mathcal{T}_q^\dagger} \uplus_{\mathrm{j}=2}^{|\mathbf{X}|} \{X_\mathrm{j} \colon \rho(X_\mathrm{j}) = t\} \sim \boldsymbol{\pi}'_{\mathbf{\Phi}_r}\,,$$

where $\boldsymbol{\pi}'_{\mathbf{\Phi}_r}(u) \propto \mathbf{1}\{\rho(u) = t\}\mathbf{d}_{\mathcal{G}_r}(u)$, which is equivalent to $\textsc{unif}(\mathcal{B}_{r,t})$ by Definitions B.3.1 and 3.3.2, thus proving Equation (B.5). □

Main Theorem. Combining Proposition B.3.3 and Proposition B.3.2 proves Theorem 3.3.1. □

B.3.3 Proof of Theorem 3.3.2

Definition B.3.2 (L^2 Distance between $\widehat{\boldsymbol{\pi}}$ and $\boldsymbol{\pi}$ [44])**.** *The L^2 distance between discrete probability distribution $\widehat{\boldsymbol{\pi}}$ and reference distribution $\boldsymbol{\pi}$ with sample space Ω is given by* $\|\widehat{\boldsymbol{\pi}} - \boldsymbol{\pi}\|_2 = \sum_{\mathrm{i}\in\Omega} \frac{(\widehat{\pi}(\mathrm{i})-\pi(\mathrm{i}))^2}{\pi(\mathrm{i})}$.

Definition B.3.3 (Distorted chain)**.** *Given a Markov chain $\boldsymbol{\Phi}$ over finite state space $\mathcal{S}$ and an arbitrary $x_0 \in \mathcal{S}$, let $\widehat{\boldsymbol{\Phi}}$ be the distorted chain such that $\forall\, u \neq x_0$, $p_{\widehat{\boldsymbol{\Phi}}}(u,\cdot) = p_{\boldsymbol{\Phi}}(u,\cdot)$, and $p_{\widehat{\boldsymbol{\Phi}}}(x_0,\cdot)$ is an arbitrary distribution with support* $\mathrm{supp}(p_{\widehat{\boldsymbol{\Phi}}}(x_0,\cdot)) \subseteq \mathrm{supp}(p_{\boldsymbol{\Phi}}(x_0,\cdot))$*. The distortion is given by $\|p_{\widehat{\boldsymbol{\Phi}}}(x_0,\cdot) - p_{\boldsymbol{\Phi}}(x_0,\cdot)\|$ as defined in Definition B.3.2.*

Lemma 7. *Given a finite state, positive recurrent Markov chain $\boldsymbol{\Phi}$ over state space $\mathcal{S}$, let $\widehat{\boldsymbol{\Phi}}$ be the chain distorted at some $x_0 \in \mathcal{S}$ from Definition B.3.3. Let*

$$\mathcal{X} = \{(X_1,\ldots,X_\xi)\colon X_1 = x_0\,,\, \xi = \min\{t>0\colon X_{t+1} = x_0\}\,,\, p_{\boldsymbol{\Phi}}(X_1,\ldots,X_\xi) > 0\}\,,$$

denote the set of all possible arbitrary lengths RWT*s that begin and end at x_0 from Definition 3.2.2. Given a tour $\mathbf{Y} \in \mathcal{X}$ sampled from $\boldsymbol{\Phi}$ and a bounded function $F\colon \mathcal{X} \to \mathbb{R}$,*

$$\mathbb{E}_{\boldsymbol{\Phi}}\left[\frac{p_{\widehat{\boldsymbol{\Phi}}}(Y_1,Y_2)}{p_{\boldsymbol{\Phi}}(Y_1,Y_2)}F(\mathbf{Y})\right] = \mathbb{E}_{\widehat{\boldsymbol{\Phi}}}\left[F(\mathbf{Y})\right]\,, \tag{B.7}$$

where $\mathbb{E}_{\boldsymbol{\Phi}}$ and $\mathbb{E}_{\widehat{\boldsymbol{\Phi}}}$ are expectations under the distribution of tours sampled from $\boldsymbol{\Phi}$ and $\widehat{\boldsymbol{\Phi}}$.

Proof. All tours in $\mathcal{X}$ are of finite length because of the positive recurrence of $\boldsymbol{\Phi}$. The ratio of the probability of sampling the tour $\mathbf{Y} = (Y_1,\ldots,Y_{\xi'})$ from the chain $\widehat{\boldsymbol{\Phi}}$ to $\boldsymbol{\Phi}$ is given by

$$\frac{p_{\widehat{\boldsymbol{\Phi}}}(\mathbf{Y})}{p_{\boldsymbol{\Phi}}(\mathbf{Y})} = \frac{\prod_{\mathrm{j}=1}^{\xi'} p_{\widehat{\boldsymbol{\Phi}}}(Y_\mathrm{j},Y_{\mathrm{j}+1})}{\prod_{\mathrm{j}=1}^{\xi'} p_{\boldsymbol{\Phi}}(Y_\mathrm{j},Y_{\mathrm{j}+1})} = \frac{p_{\widehat{\boldsymbol{\Phi}}}(Y_1,Y_2)}{p_{\boldsymbol{\Phi}}(Y_1,Y_2)}\,, \tag{B.8}$$

because $p_{\boldsymbol{\Phi}}(Y_\mathrm{j},\cdot) = p_{\widehat{\boldsymbol{\Phi}}}(Y_\mathrm{j},\cdot)$, $\forall 1 < \mathrm{j} \leq \xi'$ because $Y_\mathrm{j} \neq x_0$ by the definitions of $\mathcal{X}$ and $\widehat{\boldsymbol{\Phi}}$. Because $\mathrm{supp}(p_{\widehat{\boldsymbol{\Phi}}}(x_0,\cdot)) \subseteq \mathrm{supp}(p_{\boldsymbol{\Phi}}(x_0,\cdot))$, $\mathrm{supp}(p_{\widehat{\boldsymbol{\Phi}}}(\mathbf{Y})) \subseteq \mathrm{supp}(p_{\boldsymbol{\Phi}}(\mathbf{Y}))$. The theorem

statement therefore directly draws from the definition of importance sampling [117, Def 3.9] with the importance weights derived in Equation (B.8). □

Lemma 8. *Given a simple random walk $\mathbf{\Phi}$ on the connected non-bipartite graph $\mathcal{G}$ from Definition 3.2.1, let $\widehat{\mathbf{\Phi}}$ be the chain distorted at some $x_0 \in \mathcal{S}$ from with distortion ν Definition B.3.3. Let $\lambda = \widehat{\mathbf{d}}(x_0)/\mathbf{d}(x_0)$. Let $f\colon \mathcal{E} \to \mathbb{R}$ bounded by B, and $F(\mathbf{X}) = \sum_{\mathrm{j}=1}^{|\mathbf{X}|} f(X_{\mathrm{j}}, X_{\mathrm{j}+1})$, where $\mathbf{X}$ is an* RWT *as defined in Section B.2.2. The bias of an* RWT *Estimate (Equation (3.2)) computed using tours sampled over $\widehat{\mathbf{\Phi}}$ and using $\widehat{\mathbf{d}}(x_0)$ as the degree is given by*

$$\text{BIAS} = \left| \mathbb{E}_{\widehat{\mathbf{\Phi}}} \left[\frac{\widehat{\mathbf{d}}(x_0)}{2} F(\mathbf{X}) \right] - \mu(\mathcal{E}) \right| \leq (\lambda\nu + |1-\lambda|) \frac{\sqrt{3} B |\mathcal{E}|}{\sqrt{\delta}} ,$$

where δ is the spectral gap of $\mathbf{\Phi}$, and B is the upper bound of f.

Proof. From Lemma 7 and Lemma 3 we have, respectively,

$$\mathbb{E}_{\widehat{\mathbf{\Phi}}} \left[\frac{\widehat{\mathbf{d}}(x_0)}{2} F(\mathbf{X}) \right] = \mathbb{E}_{\mathbf{\Phi}} \left[\frac{\widehat{\mathbf{d}}(x_0)}{2} \frac{p_{\widehat{\mathbf{\Phi}}}(X_1, X_2)}{p_{\mathbf{\Phi}}(X_1, X_2)} F(\mathbf{X}) \right] ,$$
$$\mu(\mathcal{E}) = \mathbb{E}_{\mathbf{\Phi}} \left[\frac{\mathbf{d}(x_0)}{2} F(\mathbf{X}) \right] .$$

Subtracting the two, squaring both sides and using the Cauchy-Schwarz inequality decomposes the squared bias into

$$\text{BIAS} = \left| \mathbb{E}_{\mathbf{\Phi}} \left[\left(\frac{\widehat{\mathbf{d}}(x_0)}{\mathbf{d}(x_0)} \frac{p_{\widehat{\mathbf{\Phi}}}(X_1, X_2)}{p_{\mathbf{\Phi}}(X_1, X_2)} - 1 \right) \frac{\mathbf{d}(x_0)}{2} F(\mathbf{X}) \right] \right| .$$
$$\text{BIAS}^2 \leq \underbrace{\mathbb{E} \left[\left(\frac{\widehat{\mathbf{d}}(x_0)}{\mathbf{d}(x_0)} \frac{p_{\widehat{\mathbf{\Phi}}}(x_0, X_2)}{p_{\widehat{\mathbf{\Phi}}}(x_0, X_2)} - 1 \right)^2 \right]}_{\text{BIAS}_{\text{dist}}} \underbrace{\mathbb{E} \left[\left(\frac{\mathbf{d}(x_0)}{2} F(\mathbf{X}) \right)^2 \right]}_{\text{BIAS}_{\text{spectral}}} ,$$

where the expectation is under $\mathbf{\Phi}$. Using definitions from the theorem statement,

$$\begin{aligned} \text{BIAS}_{\text{dist}} =& \frac{\widehat{\mathbf{d}}(x_0)^2}{\mathbf{d}(x_0)^2} \mathbb{E} \left[\left(\frac{p_{\widehat{\mathbf{\Phi}}}(x_0, X_2)}{p_{\mathbf{\Phi}}(x_0, X_2)} \right)^2 \right] + 1 - 2 \frac{\widetilde{\mathbf{d}}(x_0)}{\mathbf{d}(x_0)} \mathbb{E} \left[\frac{p_{\widehat{\mathbf{\Phi}}}(x_0, X_2)}{p_{\mathbf{\Phi}}(x_0, X_2)} \right] \\ =& \lambda^2 (1 + \nu^2) + 1 - 2\lambda = \lambda^2 + \lambda^2 \nu^2 + 1 - 2\lambda \\ =& \lambda^2 \nu^2 + (1-\lambda)^2 \leq (\lambda\nu + |1-\lambda|)^2 . \end{aligned}$$

Because $F(\mathbf{X}) \leq B\xi$, the tour length, from Lemma 6, we see that

$$\text{BIAS}_{\text{spectral}} \leq \frac{\mathbf{d}(x_0)^2 B^2}{4} \frac{3}{\boldsymbol{\pi}_{\boldsymbol{\Phi}}(x_0)^2 \delta} = \frac{3B^2|\mathcal{E}|^2}{\delta},$$

and combining both biases completes the proof for BIAS. □

Main Theorem. Note that by linearity of expectations

$$\begin{aligned}\mathbb{E}\left[\hat{\mu}\left(\mathcal{T}^{\dagger}_{2:r}; f\right) \middle| \mathcal{T}^{\dagger}_{2:r-1}\right] =& \mathbb{E}\left[\frac{\widehat{\mathbf{d}}(\zeta_r)}{2|\mathcal{T}^{\dagger}_r|} \sum_{\mathbf{X} \in \mathcal{T}^{\dagger}_r} \sum_{\mathrm{j}=2}^{|\mathbf{X}|-1} f(X_{\mathrm{j}}, X_{\mathrm{j}+1})\right], \\ =& \mathbb{E}_{\mathbf{X} \sim \widehat{\boldsymbol{\Phi}}_r}\left[\frac{\widehat{\mathbf{d}}(\zeta_r)}{2} \sum_{\mathrm{j}=1}^{|\mathbf{X}|} f'(X_{\mathrm{j}}, X_{\mathrm{j}+1})\right],\end{aligned}$$

where $\mathbf{X}$ is an *RWT* on $\widehat{\boldsymbol{\Phi}}_r$ that depends on $\mathcal{T}^{\dagger}_{2:r-1}$ and $f'(u,v) \triangleq \mathbf{1}\{u, v \neq \zeta_r\} f(u,v)$. Applying Lemma 8 completes the proof because $\widehat{\boldsymbol{\Phi}}_r$ is a distorted chain by Definition B.3.3. □

B.4 Proofs for Section 3.4

B.4.1 Proof of Proposition 3.4.1

Proof. From [18, Thm-3.1], we know that each disconnected component of G leads to a disconnected component in $\mathcal{G}^{(k-1)}$, and if $\mathcal{I}_1$ contains a subgraph in each connected component, Proposition 3.3.1 (a) is satisfied. We now prove that $\forall\, s \in \mathcal{V}^{(k-1)}$, if $\rho(s) = r > 1$, $\exists\, s' \in \mathbf{N}(s)\colon \rho(s') < r$ which simultaneously satisfies Proposition 3.3.1 (b) and Proposition 3.3.1 (c).

W.l.o.g. let the vertex with the smallest distance from the seed vertices be denoted by $\hat{u} = \text{argmin}_{u \in V(s)} \text{DIST}(u)$. When $\text{DIST}(\hat{u}) > 0$, there exists $v \in \mathbf{N}_G(\hat{u})$ such that $\text{DIST}(v) < \text{DIST}(\hat{u})$ by the definition of DIST. More concretely, v would be the penultimate vertex in the shortest path from the seed vertices to $\hat{u}$. Let $v' \neq \hat{u}$ be a non-articulating vertex of s, which is possible because any connected graph has at least 2 non-articulating vertices. Let $s_1 = G\left(V(s) \backslash \{v'\} \cup \{v\}\right) \in \mathcal{V}^{(k-1)}$. Now, $\rho(s_1) < \rho(s)$ because v' has been replaced with a vertex at necessarily a smaller distance and because the indicator in the definition of *rho*

will always be 0 in this case. Moreover, $s_1 \circ s = G\left(V(s) \cup \{v\}\right) \in \mathcal{V}^{(k)}$, and hence an edge exists between the two.

When $\textsc{dist}(\hat{u}) = 0$, there exists $v \in \mathbf{N}_G(\hat{u})$ such that $\textsc{dist}(v) = 0$. There exists a non-articulating $v' \in V(s) \backslash V^*$ because otherwise V^* would have been disconnected. Observing that $\textsc{dist}(v') + \mathbf{1}\{v' \in V(\mathcal{I}_1) \backslash V^*)\} > 0$ completes the proof of ergodicity. □

B.4.2 Proof of Proposition 3.4.2

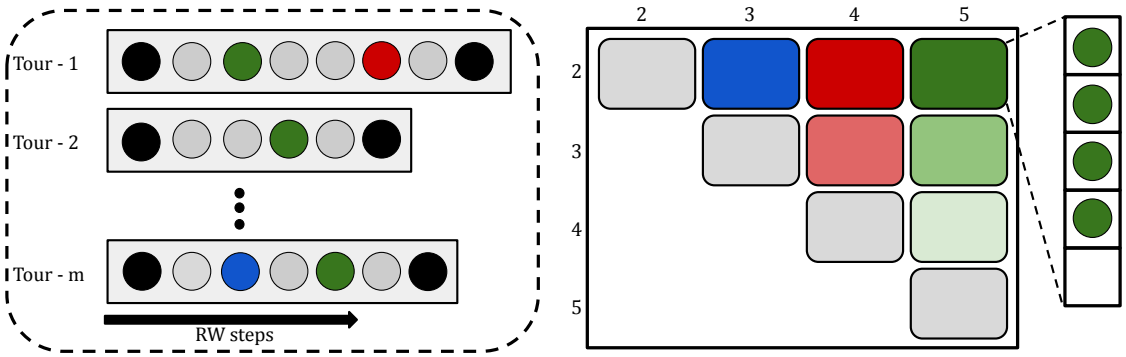
Sampling Probability. Consider the lines Lines 3 to 5. The probability of sampling the pair (u, v) from $V(s) \times \mathbf{N}_G\left(V(s)\right)$ is given by

$$\begin{aligned} P(u,v) &= \sum_{a \in V(s) \backslash \{u\}} P(v|a,u) P(a|u) P(u) \\ &= \sum_{a \in V(s) \backslash \{u\}} \frac{\mathbf{1}\{v \in \mathbf{N}(a)\}}{\mathbf{d}(a)} \frac{\mathbf{d}(a)}{\deg_s - \mathbf{d}(u)} \frac{\deg_s - \mathbf{d}(u)}{(k-1-1)\deg_s} \\ &\propto \sum_{a \in V(s) \backslash \{u\}} \mathbf{1}\{v \in \mathbf{N}(a)\} = |N(v) \cap V(s) \backslash \{u\}| = \textsc{bias}\,, \end{aligned}$$

where BIAS is defined in Line 6 and corrected for in Line 7. After the rejection, therefore, $(u, v) \sim \textsc{unif}(V(s) \times \mathbf{N}_G\left(V(s)\right))$.

Line 9 constitutes an importance sampling with unit weight for pairs (u, v), where removing u from and adding v to $V(s)$ produces a $k-1$-*CIS* and zero otherwise. In Line 9, because removing a non-articulating vertex and adding another vertex to s cannot lead to a disconnected subgraph, we can avoid a DFS when $u \notin \mathcal{A}_s$. This completes the proof. □

Time Complexity. Assuming access to a precomputed vector of degrees, the part up to Line 1 is $\mathrm{O}(k - 1^2)$. In each proposal, Lines 3 and 4 are $\mathrm{O}(k-1)$, and Line 5 is $\mathrm{O}(\Delta_s)$. Line 6 is $\mathrm{O}(k-1)$, and the expected complexity of Line 9 is $\mathrm{O}(k - 1^2\, {}^{|\mathcal{A}_s|}/_{k-1})$ because in expectation only ${}^{|\mathcal{A}_s|}/_{k-1}$ graph traversals will be required. The acceptance probability is $\geq {}^{1}/_{k-1}$ is Line 7 and $\geq \frac{k-1-|\mathcal{A}_s|}{k-1}$. The expected number of proposals is therefore $\leq \frac{k-1^2}{k-1-|\mathcal{A}_s|}$. As such, the expected time complexity is $\mathrm{O}(k - 1^2(1 + \frac{\Delta_s + k - 1|\mathcal{A}_s|}{k-1-|\mathcal{A}_s|}))$. □



(a) $\mathcal{T}_2$, the set of *RWT*s sampled in $\mathcal{G}_2$.

(b) The reservoir matrix, $\widehat{\mathbf{U}}_{r,t}$ for $2 \leq r < t \leq R$.

Figure B.1. Parallel *RWT*s and Reservoirs: Figure B.1a shows the set of m *RWT*s sampled on $\mathcal{G}_2$ in parallel, where the supernode ζ_2 is colored black. The gray, blue, red and green colors represent states in stratum 2–5, respectively. Figure B.1b shows the upper triangular reservoir matrix in which the cell in the r-th row and t-th column contains samples from $\widehat{\mathbf{U}}_{r,t}$.

B.5 Additional Implementation Details

B.5.1 Parallel Sampling with a Reservoir Matrix.

Given a reasonably large M and the number of strata R, we initialize an upper triangular matrix of empty reservoirs $[\widehat{\mathbf{U}}_{r,t}]_{2\leq r<t\leq R}$ and a matrix of atomic counters $[\hat{\mathrm{M}}_{q,r}]_{2\leq r<t\leq R}$ initialized to 0. In each stratum r, while being sampled in parallel whenever a tour enters the t-th stratum, $\hat{\mathrm{M}}_{r,t}$ is incremented, and with a probability $\min(1, \mathrm{M}/\hat{\mathrm{M}}_{r,t})$, the state is inserted into a random position in the reservoir $\widehat{\mathbf{U}}_{r,t}$ and rejected otherwise. The only contention between threads in this scheme is at the atomic counter and in the rare case where two threads choose the same location to overwrite, wherein ties are broken based on the value of the atomic counter at the insertion time, guaranteeing thread safety. The space complexity of a reservoir matrix is therefore $O(R^2\mathrm{M})$.

A toy example of this matrix is presented in Figure B.1, where $R = 5$, and the *RWT*s are being sampled on the graph stratum $\mathcal{G}_2$. Whenever (non-gray) states in $\mathcal{I}_{3:5}$ are visited, they are inserted into the corresponding reservoirs–$\widehat{\mathbf{U}}_{2,5}$ is depicted in detail.

B.5.2 *PSRW* Neighborhood

The neighborhood of a k-*CIS* s in $\mathcal{G}^{(k)}$ is the set of all vertices $u, v \in V$ such that replacing u with v in s yields a k-*CIS*. Formally,

$$\mathbf{N}_{\mathcal{G}^{(k)}}(s) \equiv \left\{(u,v) \in V(s) \times \mathbf{N}_G\left(V(s)\right) : G\left(V(s) \cup \{v\} \backslash \{u\}\right) \in \mathcal{V}^{(k)}\right\}, \tag{B.9}$$

where $\mathbf{N}_G(V(s)) = \cup_{x \in V(s)} \mathbf{N}_G(x)$ is the union of the neighborhood of each vertex in s. The size of the neighborhood is then $\mathrm{O}(k\,\mathbf{N}_G(V(s))) \in \mathrm{O}(k^2\Delta_G)$ because $\mathbf{N}_G(V(s)) \in \mathrm{O}(k\Delta_G)$, where Δ_G is the maximum degree in G. Each potential neighbor further requires a connectivity check in the form of a BFS or DFS, which implies that the naive neighborhood sampling algorithm requires $\mathrm{O}(k^4\Delta_G)$ time.

Articulation Points

Apart from the rejection sampling algorithm from Algorithm 2, we use articulation points to efficiently compute the subgraph bias γ from Equation (3.9). Specifically, given the $k-1$-*CIS*, s, $\gamma(s) = \binom{\kappa - \mathcal{A}_s}{2}$, $\mathcal{A}_s$ is the set of articulation points of s. This draws directly from [18, Sec-3.3] and the definition of articulation points. [47] showed that for any simple graph s the set of articulation points can be computed in $O(|V(s)| + |E(s)|)$ time.

B.5.3 Proof of Proposition 3.4.3

Proposition B.5.1 (Extended Version of Proposition 3.4.3)**.** *We assume a constant number of tours m in each stratum and ignore graph loading. The* Ripple *estimator of k-*CIS *counts described in Algorithm 3 has space complexity in*

$$\mathrm{O}(k^3 D_G^2 \mathrm{M} + |\mathcal{H}|) \equiv \widehat{\mathrm{O}}(k^3 + |\mathcal{H}|)\,,$$

where $\widehat{\mathrm{O}}$ ignores all factors other than k and $|\mathcal{H}|$, M is the size of the reservoir from Section 3.4.3, D_G is the diameter of G, and $|\mathcal{H}|$ is the number of patterns of interest.

Algorithm 3: *Ripple* for Subgraph Counting

Input: Input graph G, Order k, Set of subgraph patterns $\mathcal{H}$ of interest
Input: Initial vertex stratum $\mathcal{I}_1$, Reservoir Size M and Error Bound ϵ
Output: $\hat{\mu}$, an asymptotically unbiased estimate of $\mathcal{C}^{(k)}$

`/* Initialization */`
1 $\hat{\mu} = 0$, $\widehat{\beta}_{q,t} = 0$, $\widehat{\mathbf{U}}_{q,t} = \emptyset$, $\forall 1 \le q < t \le R$;
2 Run BFS for stratification $\rho\colon \mathcal{V}^{(k-1)} \to \{1, \dots, R\}$, with $\mathcal{I}_1$ (Proposition 3.4.1)
`/* Exact computation in the first stratum */`
3 **foreach** $u \in \mathcal{I}_1, v \in \mathbf{N}_{\mathcal{G}^{(k-1)}}(u)$ **do**
4 Update $\widehat{\beta}_{1,\rho(v)} \mathrel{+}= 1$, $\widehat{\mathbf{U}}_{1,\rho(v)} \cup\!= v$
5 Update $\hat{\mu} \mathrel{+}= \left(\frac{\mathbf{1}\{u \circ v \sim H\}}{\gamma(u \circ v)}\right)_{H \in \mathcal{H}}$; `// Equation (3.9)`
`/* Estimate remaining strata */`
6 **for** $r \in 2, \dots, R$ **do**
7 Initialize $\hat{\mu}_r = 0$, $m_r = 0$
8 **parallel while** *Equation (3.10) is not satisfied* **do**
9 Sample q from $\{1, \dots, r-1\}$ w.p. $\widehat{\beta}_{q,r}$
10 Sample u from $\widehat{\mathbf{U}}_{q,r}$; `// Equation (3.6)`
11 Sample $v \sim \text{UNIF}(\mathbf{N}_{\mathcal{G}^{(k-1)}}(u))$; `// Algorithm 2`
12 **while** $\rho(v) \ge r$ **do**
13 Update $\hat{\mu}_r \mathrel{+}= \left(\frac{\mathbf{1}\{u \circ v \sim H\}}{\gamma(u \circ v)}\right)_{H \in \mathcal{H}}$; `// Equation (3.8)`
14 **if** $\rho(v) > r$ **then**
15 Update $\widehat{\beta}_{r,\rho(v)} \mathrel{+}= 1$; `// Equation (3.4)`
16 Update $\widehat{\mathbf{U}}_{r,\rho(v)} \cup\!= v$; `// Equation (3.6)`
17 $u := v$
`/* Proposition B.3.1 and Algorithm 2 */`
18 **if** $\rho(u) = r$ **then**
19 Sample $v \sim \text{UNIF}(\mathbf{N}_{\mathcal{G}^{(k-1)}}(u))$
20 **else**
21 **while** $\rho(v) \ne r$ **do**
22 Sample $v \sim \text{UNIF}(\mathbf{N}_{\mathcal{G}^{(k-1)}}(u))$
23 $m_r \mathrel{+}= 1$
24 Compute $\widehat{\deg}_r = \sum_{q=1}^{r-1} \widehat{\beta}_{q,r}$; `// Equation (3.5)`
25 $\hat{\mu} \mathrel{+}= \frac{\widehat{\deg}_r}{2\, m_r} \hat{\mu}_r$; `// Equations (3.7) and (3.8)`
26 Update $\widehat{\beta}_{r,t} \mathrel{*}= \frac{\widehat{\deg}_r}{m_r}$, $\forall t > r$; `// Equation (3.4)`
27 **return** $\hat{\mu}$

The total number of random walk steps is given by $\mathrm{O}(k^3 m D_G \Delta_G C_{\text{REJ}})$*, where* C_{REJ} *is the number of rejections in Line 21 of Algorithm 3,* Δ_G *is the largest degree in* G*, and the total time complexity is* $\widehat{\mathrm{O}}(k^7 + |\mathcal{H}|)$.

Remark 2. *In practice, we adapt the proposals in Algorithm 2 to minimize* C_{REJ} *using heuristics over the values of* $\text{DIST}(\cdot)$ *from Proposition 3.4.1.*

Lemma 9. *Given a graph stratum* $\mathcal{G}_r$ *from Definition 3.3.2, for some* $r > 1$*, define* $\alpha_r = {}^{|\{u \in \mathcal{I}_r \colon \mathbf{N}(u) \cap \mathcal{I}_{1:r-1} \neq \emptyset\}|}/_{|\mathcal{I}_r|}$ *as the fraction of vertices in the* r*-th vertex stratum that share an edge with a previous stratum. The return time* ξ_r *of the chain* $\mathbf{\Phi}_r$ *to the supernode* $\zeta_r \in \mathcal{V}_r$ *follows* $\mathbb{E}_{\mathbf{\Phi}_r}[\xi_r] \leq \frac{2\bar{\mathbf{d}}_r}{\alpha_r}$*, where* $\bar{\mathbf{d}}_r$ *is the average degree in* $\mathcal{G}$ *of all vertices in* $\mathcal{I}_r$*.*

Proof. Because $\alpha_r \mathcal{I}_r$ vertices have at least one edge incident on ζ_r, $\mathbf{d}_{\mathcal{G}_r}(\zeta_r) \geq \alpha_r \mathcal{I}_r$. From Definition 3.3.2, because all edges not incident on $\mathcal{I}_r$ are removed from $\mathcal{G}_r$, $\text{Vol}(\mathcal{G}_r) \leq 2 \sum_{u \in \mathcal{I}_r} \mathbf{d}_{\mathcal{G}}(u)$. Therefore, from Lemma 3,

$$\mathbb{E}_{\mathbf{\Phi}_r}[\xi_r] = \frac{\text{Vol}(\mathcal{G}_r)}{\mathbf{d}(\zeta_r)} \leq \frac{2 \sum_{u \in \mathcal{I}_r} \mathbf{d}_{\mathcal{G}}(u)}{\alpha_r \mathcal{I}_r} = \frac{2\bar{\mathbf{d}}_r}{\alpha_r}\,.$$

□

Proposition B.5.2. *The Ergodicity-Preserving Stratification from Proposition 3.4.1 is such that* $\alpha_r = 1$ *for all* $r > 1$ *as defined in Lemma 9, and consequently, the diameter of each graph stratum is* ≤ 4*. The total number of strata* $R \in \text{O}(k\, D_G)$*, where* D_G *is the diameter of* G*.*

Proof. We show in Section B.4.1 that for each vertex $s \in \mathcal{V}^{(k-1)}$, if $\rho(s) = r > 1$, there exists $s' \in \mathbf{N}(s)$ such that $\rho(s') < r$. This implies that $\alpha_r = 1$. In $\mathcal{G}_r$, therefore, from ζ_r, all vertices in $\mathcal{I}_r$ are at unit distance from ζ_r, and vertices in $\mathbf{N}(\mathcal{I}_r) \backslash \mathcal{I}_r$ are at a distance of 2 from ζ_r. Because no other vertices are present in $\mathcal{G}_r$, this completes the proof of the first part. Trivially, $R \leq (k-1) \cdot \max_{u \in V} \text{DIST}(u) \in \text{O}(k \cdot D_G)$. □

Memory Complexity. From Algorithm 3, we compute a single count estimate per stratum and maintain reservoirs and inter-partition edge count estimates for each $2 \leq q < t \leq R$. Because a reservoir $\widehat{\mathbf{U}}_{q,t}$ needs $\text{O}(k\text{M})$ space (Section B.5.1), the total memory requirement is $\text{O}(R^2\, k\text{M})$, where R is the number of strata. From Proposition B.5.2, plugging $R \in \text{O}(kD_G)$, and because storing the output $\hat{\mu}$ requires $\text{O}(|\mathcal{H}|)$ memory the proof is completed. □

Time Complexity. The stratification requires a single BFS $\in \mathrm{O}(|V|+|E|)$ from Section 3.4.2. In Line 3, the estimation phase starts by iterating over the entire higher-order neighborhood of each subgraphs in $\mathcal{I}_1$. Based on Section B.5.2, Line 5 is in $\mathrm{O}(k^2)$. Because the size of the higher-order neighborhood of each subgraph is $\mathrm{O}(k^2\Delta_G)$ from Section B.5.2, the initial estimation phase will require $\mathrm{O}(|\mathcal{I}_1|\, k^4\Delta_G)$ time.

In all other strata $r = 2, \ldots, R$, we assume that m tours are sampled in Line 8. Starting each tour (Lines 9 to 11) requires order of magnitude R time, leading to a total time of $\mathrm{O}(m\, R^2) \in \mathrm{O}(mk^2 D_G^2)$ because $R \in \mathrm{O}(kD_G)$ from Proposition B.5.2. The total time for these ancillary procedures is $\mathrm{O}(mk^2 D_G^2 + |\mathcal{I}_1|\, k^4\Delta_G)$

Therefore, the time complexity of bookkeeping and setup is $\mathrm{O}(mk^2 D_G^2 + |\mathcal{I}_1|\, k^4\Delta_G + |V| + |E|) \in \widehat{\mathrm{O}}(k^4)$. The time complexity at each random walk step is $\mathrm{O}(k-1^2\Delta_G + k-1^4) \in \widehat{\mathrm{O}}(k^4)$ from Section B.4.2 and Section B.5.2. We assume that the expected number of rejections in Line 21 is given by C_{REJ}. The total number of random walk steps is given by $\mathrm{O}(R\, m\, C_{\text{REJ}})$ times the expected tour length. By Lemma 9 and proposition B.5.2, the expected tour length is $\mathrm{O}(\Delta_{\mathcal{G}^{(k-1)}}) \equiv \mathrm{O}(k^2\Delta_G)$. Therefore, the total number of random walk steps is $\mathrm{O}(k^3 m D_G \Delta_G C_{\text{REJ}})$.

$\mathrm{O}(|\mathcal{H}|)$ time is to print the output $\hat{\mu}$. We assume that updating $\hat{\mu}$ is amortized in constant order if we use a hashmap to store elements of the vector, and because updating a single key in said hashmap is by Equation (3.9) increments, the proof is completed. □

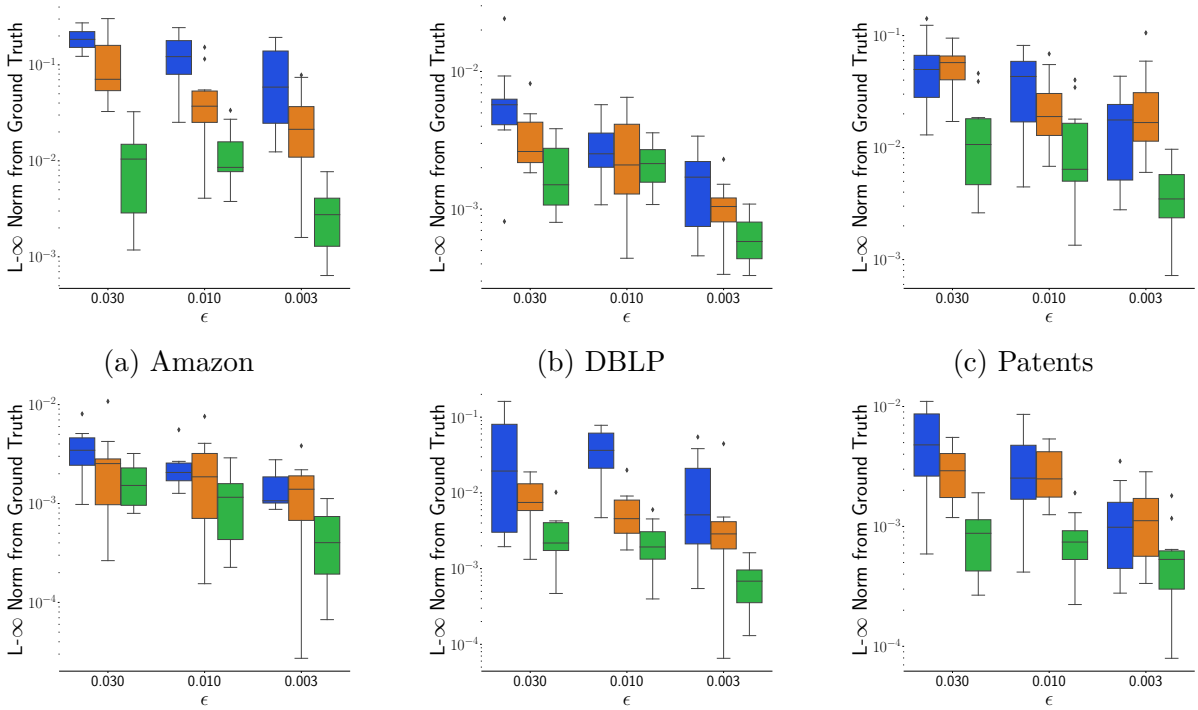
B.6 Additional Results

I now present the results of additional experiments performed on *Ripple*. Table B.2 shows the dispersion, $\frac{\max - \min}{\text{mean}}$, of the estimates that were used to measure the running time and space utilization of *Ripple* in Section 3.5.1. Figure B.2 shows the L-∞ norm from the ground truth for $k = 5$ with $\textsc{m} = 10^7$ while ϵ and $\mathcal{I}_1$ vary.

Trade-off between Convergence and Reservoir size. Next, I measure the effect of the reservoir capacity $\textsc{m}$ on accuracy, as discussed in Section 3.4.3. I vary $\textsc{m}$ from 50000 to 10^7 while keeping the other parameters fixed as $\epsilon = 0.003$ and $|\mathcal{I}_1| = 10^4$ and measure the L2-norm between the *Ripple* estimate and the exact value of the count vector $\mathcal{C}^{(5)}$, such as

Table B.2. Dispersion, $\frac{\max - \min}{\text{mean}}$, of *Ripple*'s estimates of $|\mathcal{V}^{(k)}|$ computed using $\epsilon = 0.003$, $|\mathcal{I}_1| = 10^4$ and $\text{M} = 10^7$ for $k = 6, 8, 10, 12$ used in the analysis in Section 3.5.1. The selected hyper-parameters provide reasonably similar estimates over 10 independent runs. Orkut for $k = 8$ exhibits the largest dispersion because of the presence of a single outlier.

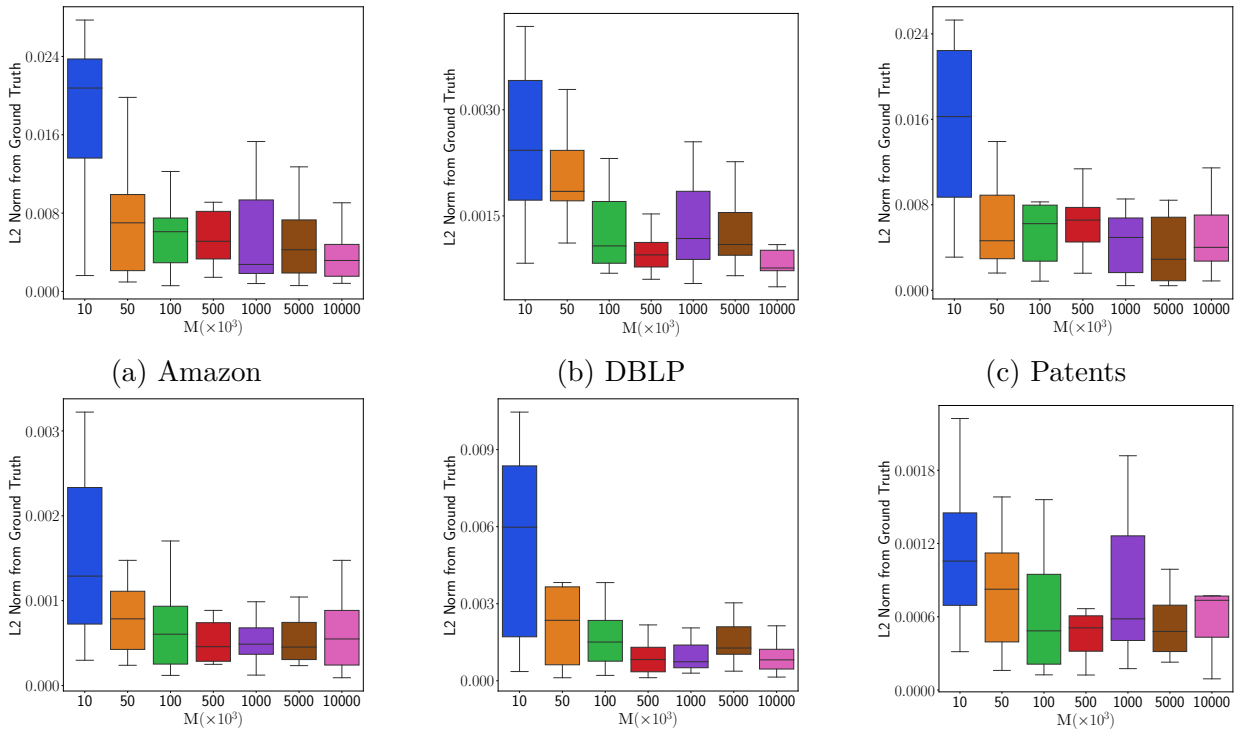
Graph	**Rel. dispersion of estimates**			
	6	8	10	12
Amazon	0.203	0.241	0.285	0.268
DBLP	0.023	0.023	0.041	0.054
Patents	0.037	0.083	0.093	0.123
Pokec	0.065	0.044	0.037	0.046
LiveJ.	0.050	0.060	0.033	0.066
Orkut	0.021	0.761	0.053	0.031



(a) Amazon (b) DBLP (c) Patents

(d) Pokec (e) Live Journal (f) Orkut

Figure B.2. Accuracy and convergence analysis for 5-*CIS* s using L-∞ norm. The y-axes show the L-∞ norm between the *Ripple* estimate and the ground truth vector $\mathcal{C}^{(5)}$, containing counts of all possible non-isomorphic subgraph patterns for various settings of ϵ and $\mathcal{I}_1$. As expected, the accuracy improves as ϵ decreases and $|\mathcal{I}_1|$ increases. Each box and whisker represents 10 runs.



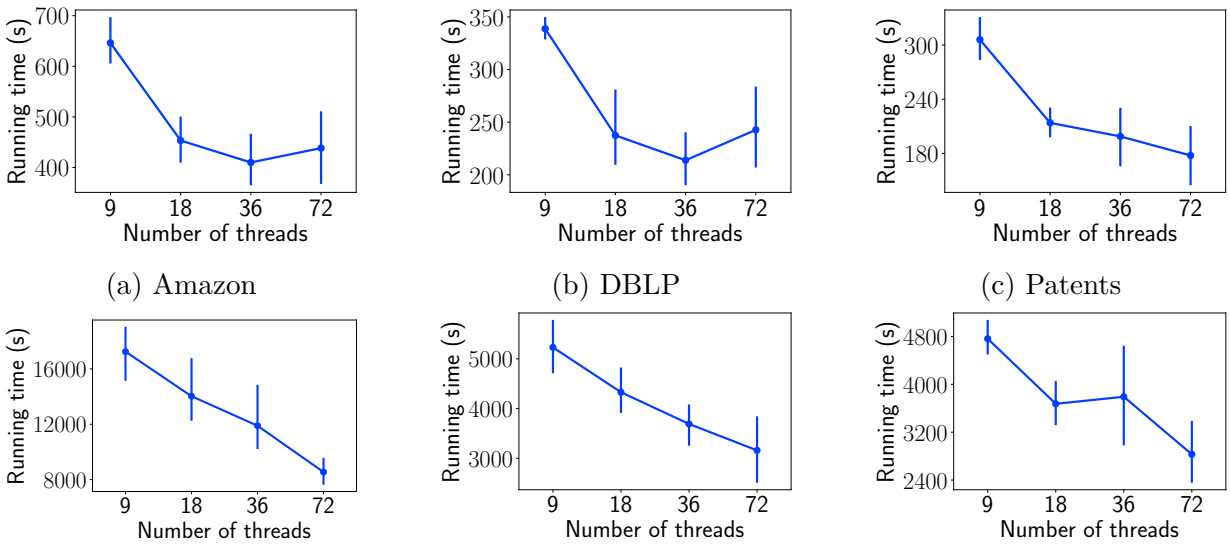
(a) Amazon (b) DBLP (c) Patents

(d) Pokec (e) Live Journal (f) Orkut

Figure B.3. Sensitivity of *Ripple* to the reservoir capacity M for $k = 5$. I verify that a larger reservoir improves the accuracy of *Ripple* estimates in all graphs because it reduces oversampling bias. Each box and whisker plot represents 10 runs.

in Section 3.5.2. We see that larger reservoirs reduce oversampling bias and improve the convergence and accuracy in all datasets.

Scalability on Number of Threads (Figure B.4). In this experiment, because of Equation (3.10), I set $\epsilon = 0.001$ to force a larger number of tours, thereby increasing the load per core and ensuring a sufficient workload. Further, I fix $k = 5$, set $|\mathcal{I}_1| = 10^4$, $\text{M} = 10^7$ and compute running times over 10 executions while excluding the graph read time, which is not parallel. We observe that my implementation does not scale linearly: as we double the number of cores, the running time decreases by $\approx 1/4$ rather than $1/2$. Local profiling using hardware performance counters (Linux's *perf*) suggests that this overhead is an outcome of increased random-access patterns of in-memory graph data, which limits the overall use of the underlying processing pipeline. Indeed, sub-optimal access patterns of graph data are a known issue that is currently handled by dedicated accelerator hardware deploying optimized



(a) Amazon (b) DBLP (c) Patents

(d) Pokec (e) Live Journal (f) Orkut

Figure B.4. Scalability w.r.t. the Number of Threads for 5-*CIS*. Despite a noticeable reduction in running time, the scalability is not linear. In fact, as we double the number of cores, the running time decreases by approximately a quarter instead of half, which may be due to the memory bandwidth limit coupled with the lack of memory locality, which is a ubiquitous problem in graph mining algorithms.

and specific caching mechanisms and memory access policies for workloads dominated by subgraph enumeration [118].

C. DETAILS FOR CHAPTER 4

C.1 Proof of Corollary 1

Proof. Collapse the states of $\mathcal{S}$ into a single state $\mathcal{S}$ to form a state-collapsed MC $\Phi'(\mathbf{W})$, with transition probabilities given by Definition 4.2.1.

Let $(\mathcal{S}, \mathbf{X}^{(\cdot)}(2), \ldots, \mathbf{X}^{(\cdot)}(\xi^{(\cdot)}))$ be a sequence of discrete states of the r-th tour of the state-collapsed MC $\Phi'(\mathbf{W})$. Note that $\mathcal{S}$ is the renewal state of the tour $\Xi^{(\cdot)}$, i.e., $\mathbf{X}^{(\cdot)}(1) = \mathcal{S}$.

The time reversibility of $\Phi(\mathbf{W})$ implies that $p(\mathbf{x}; \mathbf{W})p_\Phi(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}; \mathbf{W})p_\Phi(\mathbf{y}, \mathbf{x})$, where p_a indicates the probability transition matrix of MC a. Let $Z_\mathcal{S}(\mathbf{W}) = \sum_{\mathbf{y}\in\mathcal{S}} \mathrm{e}^{-E(\mathbf{y};\mathbf{W})}$. We now show that $\Phi'(\mathbf{W})$ is time-reversible using the fact that the steady state distribution of $\Phi(\mathbf{W})$ is known up to a constant factor. Thus, we "guess" the steady state distribution in $\Phi'(\mathbf{W})$ of $\mathcal{S}$ as $p(\mathcal{S}; \mathbf{W}) = Z_\mathcal{S}(\mathbf{W})/Z(\mathbf{W})$ and verify that, because $\mathcal{S}$ is a proper subset of Ω, the balance equations of $\Phi'(\mathbf{W})$ are time reversible:

$$
\begin{aligned}
p(\mathcal{S}; \mathbf{W})p_{\Phi'}(\mathcal{S}, \mathbf{x}) &\coloneqq \frac{Z_\mathcal{S}(\mathbf{W})}{Z(\mathbf{W})} \sum_{\mathbf{y}\in\mathcal{S}} \frac{\mathrm{e}^{-E(\mathbf{y};\mathbf{W})}}{Z_\mathcal{S}(\mathbf{W})} p_\Phi(\mathbf{y}, \mathbf{x}) \\
&= \sum_{\mathbf{y}\in\mathcal{S}} p(\mathbf{y}; \mathbf{W})p_\Phi(\mathbf{y}, \mathbf{x}) \\
&= \sum_{\mathbf{y}\in\mathcal{S}} p(\mathbf{x}; \mathbf{W})p_\Phi(\mathbf{x}, \mathbf{y}) \qquad \text{see}^\dagger \\
&= p(\mathbf{x}; \mathbf{W})p_{\Phi'}(\mathbf{x}, \mathcal{S}),
\end{aligned}
$$

†from the time reversibility of $\Phi(\mathbf{W})$. Thus, all states $\mathbf{x} \in \Omega'\backslash\{\mathcal{S}\}$ in $\Phi'(\mathbf{W})$ have the same steady state distribution as in $\Phi(\mathbf{W})$: $p(\mathbf{x}; \mathbf{W})$. □

C.2 Proof of Lemma 4

Lemma (Perfect sampling of tours)**.** *Let*

$$
\mathcal{C}_k = \{(\mathbf{x}, \mathbf{X}^{(\mathrm{i})}(2), \ldots, \mathbf{X}^{(\mathrm{i})}(k))\}_\mathrm{i}
$$

be a set of tours of length $k \leq K$, with $\mathbf{x}$ sampled from $\mathcal{S}$ according to some distribution.

Then, there exists a distribution G_k such that the random variables

$$\mathcal{G}_k \coloneqq \{g(\sigma) : \forall \sigma \in \mathcal{C}_k\} \tag{C.1}$$

are i.i.d. samples of G_k, with g defined over the appropriate σ-algebra (e.g., k RBM states) with $\|g(\cdot)\|_1 \leq \infty$.

Moreover, if we perform M tours, these tours finish in finite time and $\{\xi^{(r)}\}_{r=1}^{M}$ is an i.i.d. sequence with a well-defined probability distribution $p(\xi^{(\cdot)} = k)$.

Proof. Consider an infinite run of the MCMC $\Phi'(\mathbf{W})$: $\mathbf{X}(1), \mathbf{X}(2), \ldots$, starting at state $\mathbf{X}(1) = \mathcal{S}$. Divide this infinite run into tours, the longest segments of consecutive states that start at state $\mathcal{S}$ but do not contain $\mathcal{S}$ in any other states in the segment. Let $\xi^{(r)}$ be the length of the r-th tour. Because $\Phi'(\mathbf{W})$ is an irreducible Markov chain, it is positive recurrent [90, Theorem 6.3.8], and we can use Kac's theorem [119, Theorem 10.2.2] to assert that $\mathbb{E}[\xi^{(\cdot)}] < \infty$, which also implies $\xi^{(\cdot)} < \infty$ almost surely (i.e., except for a set of measure zero). Define $R_{r+1} = R_r + \xi^{(r+1)}$, with $R_0 = 0$. Define

$$\mathcal{G}_k = \{g(\mathbf{X}(R_{r-1}), \ldots, \mathbf{X}(R_r - 1)) : \quad r = 1, \ldots, M, \xi^{(r)} = k\},$$

with $M > 1$. By the strong Markov property, there exists a distribution G_k such that $\mathcal{G}_k$ is an iid sequence from G_k. Note that $\{\xi^{(r)}\}_{r=1}^{M}$ is also iid. Further, note that by Corollary 1 we can equivalently consider the MC $\Phi(\mathbf{W})$, starting at state $\mathbf{x}$ sampled from the stopping set $\mathcal{S}$, which concludes the proof. □

C.3 Proof of Theorems 4.2.1 and 4.2.3

Proof. For simplicity, in what follows we combine the proofs of Theorems 4.2.1 and 4.2.3, specializing on each case when necessary. Define for all $r \geq 0$, $R_{r+1} = R_r + \xi^{(r+1)}$ and for

$t \geq 0$, $N(t) = \text{argmax}_r \mathbf{1}_{\{R_{r-1}<t\}}$, with $R_0 = 0$. $N(t)$ counts how many of the tours in the sequence $\{\xi^{(r)}\}_{r\geq 1}$ are needed to add up to the largest number smaller than t. Let

$$Y_K^{(r)} = \mathbf{1}\{\xi^{(r)} \leq K\} \sum_{t=R_{r-1}+1}^{R_r} f(\mathbf{X}^{(r)}(t - R_{r-1})).$$

By Lemma 4, both $\{Y_K^{(r)}\}_{r\geq 1}$ and $\{\xi^{(r)}\}_{r\geq 1}$ are iid sequences. Also, even in the case $K \to \infty$,

$$\mathbb{E}[\|Y_K^{(\cdot)}\|_1] \leq \mathbb{E}[\sup_{\mathbf{x}} \xi^{(\cdot)} \|f(\mathbf{x})\|_1] = \mathbb{E}[\xi^{(\cdot)}] \sup_{\mathbf{x}} \|f(\mathbf{x})\|_1 < \infty,$$

as by definition $\|f(\cdot)\|_1 < \infty$ and we know $\mathbb{E}[\xi^{(\cdot)}] < \infty$ (see Lemma 4). The Renewal-Reward Theorem [28, Chap 3-Thm 4.2] yields, $r \geq 1$

$$\lim_{t\to\infty} \frac{\sum_{r=1}^{N(t)} Y_K^{(r)}}{t} = \frac{\mathbb{E}\left[\mathbf{1}_{\{\xi^{(\cdot)}\leq K\}} \sum_{k=1}^{\xi^{(\cdot)}} f(\mathbf{X}^{(\cdot)}(k))\right]}{\mathbb{E}[\xi^{(\cdot)}]}. \tag{C.2}$$

Note that,

$$\begin{aligned}\frac{\sum_{r=1}^{N(t)} Y_K^{(r)}}{t} =& \frac{\mathbf{1}_{\{\xi^{(N(t'))}\leq K\}} \sum_{t'=1}^{R_{N(t)}} f(\mathbf{X}^{(N(t'))}(t' - R_{N(t')-1}))}{t} \\ =& \frac{\sum_{t'=1}^{R_{N(t)}} \mathbf{1}_{\{\xi^{(N(t'))}\leq K\}} f(\mathbf{X}^{(N(t'))}(t' - R_{N(t')-1}))}{R_{N(t)}} \cdot \frac{R_{N(t)}}{t}.\end{aligned}$$

Most importantly, $\lim_{t\to\infty} \frac{R_{N(t)}}{t} = 1$, because by definition $R_{N(t)} + \xi^{(N(t)+1)} > t$, and further, $\lim_{t\to\infty} \frac{\xi^{(N(t)+1)}}{t} = 0$, otherwise an infinitely large $\xi^{(N(t)+1)}$ would have non-zero measure, thus contradicting $\mathbb{E}[\xi^{(N(t)+1)}] < \infty$. This yields,

$$\lim_{t\to\infty} \frac{\sum_{r=1}^{N(t)} Y_K^{(r)}}{t} = \lim_{t\to\infty} \frac{\sum_{t'=1}^{R_{N(t)}} \mathbf{1}_{\{\xi^{(N(t'))}\leq K\}} f(\mathbf{X}^{(N(t'))}(t' - R_{N(t')-1}))}{R_{N(t)}}.$$

(Theorem 4.2.3) The case of K_{dyn}: As K_{dyn} is finite almost surely (see proof of Lemma 4), it makes the condition $\mathbf{1}_{\{\xi^{(N(t'))}\leq K_{\text{dyn}}\}} := 1$. Note that the sequence $\{\mathbf{X}^{(N(t'))}(t' - R_{N(t')-1})\}_{t'=1}^{R_{N(t)}}$ is just a single sample path of our MC starting at state $\mathbf{x}'$, taking $R_{N(t)}$ steps.

As our MC is irreducible and time-reversible, there is solution and the solution is unique [90, Theorem 6.3.8], and thus we can use the ergodic theorem to show

$$\lim_{t\to\infty} \frac{\sum_{t'=1}^{R_{N(t)}} f(\mathbf{X}^{(N(t'))}(t' - R_{N(t')-1}))}{R_{N(t)}} = \sum_{\mathbf{x}} f(\mathbf{x})p(\mathbf{x};\mathbf{W}),$$

and substituting the above equation in (C.2), yields

$$\mathbb{E}\left[\sum_{k=1}^{\xi^{(\cdot)}} f(\mathbf{X}^{(\cdot)}(k))\right] = \mathbb{E}[\xi^{(\cdot)}] \sum_{\mathbf{x}} f(\mathbf{x})p(\mathbf{x};\mathbf{W}), \quad r \geq 1. \tag{C.3}$$

Finally, by Kac's theorem [119, Theorem 10.2.2],

$$\mathbb{E}[\xi^{(\cdot)}] = \frac{1}{p(\mathbf{x}';\mathbf{W})} = \frac{Z(\mathbf{W})}{\mathrm{e}^{-E(\mathbf{x}';\mathbf{W})}}, \tag{C.4}$$

as $p(\mathbf{x}';\mathbf{W})$ is the steady state probability of visiting state $\mathbf{x}'$. Replacing (C.4) into (C.3) and multiplying it by $\mathrm{e}^{-E(\mathbf{x}';\mathbf{W})}$ on both sides concludes the unbiasedness proof. Thus, if $\hat{F}_r^{(K_{\mathrm{dyn}})}(\mathbf{W}, f)$ denotes the estimator $\hat{F}$ in eq.(4.5) applied to only a single tour $r = 1, \ldots, R$. Then, $\mathbb{E}[F_r^{(K_{\mathrm{dyn}})}(\mathbf{W}, f)] = F(\mathbf{W}, f)$ and the sequence $\{F_r^{(K_{\mathrm{dyn}})}(\mathbf{W}, f)\}_{r\geq 1}$ is trivially iid by the strong Markov property. This iid sequence guarantees the following convergence properties.

Error bound: Note that $\sum_{k=1}^{\xi^{(r)}} \frac{\partial E(\mathbf{X}^{(r)}(k);\mathbf{W})}{\partial \mathbf{W}}$ is upper bounded by $\xi^{(r)}B$. As $\Phi(\mathbf{W})$ is time-reversible, it is equivalent to a random walk on a weighted graph. Thus, Lemma 2(i) of Avrachenkov, Ribeiro, and Sreedharan [35] applies with $Z(\mathbf{W}) = 2d_{\mathrm{tot}}$, $Z_{\mathcal{S}}(\mathbf{W}) = d_{\mathcal{S}_n}$, and we have

$$\mathrm{var}(\hat{F}_1^{(K_{\mathrm{dyn}})}(\mathbf{W})) \leq B^2 \left((Z(\mathbf{W}))^2/(Z_{\mathcal{S}}(\mathbf{W})\delta) + 1\right).$$

By the strong Markov property the tours are independent, thus, $\mathrm{var}(\hat{F}^{(K_{\mathrm{dyn}},R)}(\mathbf{W})) = \mathrm{var}(\hat{F}_1^{(K_{\mathrm{dyn}})}(\mathbf{W}))/R$ by the Bienaymé formula. And we have already shown that the estimate of $\hat{F}_1^{(K_{\mathrm{dyn}})}(\mathbf{W})$ is unbiased. Finally, we obtain the bound through the application of Chebyshev's inequality.

(Theorem 4.2.1) The case of K: From above, $\hat{F}^{(K_{\text{dyn}},R)}(\mathbf{W}, f)$ is an unbiased estimate of $F(\mathbf{W}, f)$. The remaining of the proof is straightforward. The tours are independent. Thus $\mathbb{E}[\hat{F}^{(K,R)}(\mathbf{W})] = \mathbb{E}[\hat{F}^{(K,1)}(\mathbf{W})]$. Note that

$$E[\xi^{(\cdot)}] - \sum_{k=1}^{K-1} kP[\xi^{(\cdot)} = k] = \sum_{k=K}^{\infty} kP[\xi^{(\cdot)} = k],$$

and as B upper bounds $\|f(\cdot)\|_1$, then the bias $\mathbb{E}[\hat{F}^{(K_{\text{dyn}},1)}(\mathbf{W}) - \hat{F}^{(K,R)}(\mathbf{W})]$ can be at most $(E[\xi^{(\cdot)}] - \sum_{k=1}^{K-1} kP[\xi^{(\cdot)} = k]) \cdot B$.

□

Theorem 4.2.2

Proof of Theorem 4.2.2. Note that the condition

$$\inf_{\mathbf{x} \in \Omega \setminus \mathcal{S}} \sum_{\mathbf{y} \in \mathcal{S}} p_{\Phi}(\mathbf{x}, \mathbf{y}) \geq \epsilon$$

ensures that the MC $\Phi'(\mathbf{W})$ satisfies Doeblin's condition, and therefore $\Phi'(\mathbf{W})$ is geometrically ergodic with convergence rate $(1 - \epsilon)$ [120, pp. 30]. Finally, by Kendall's theorem [119, Theorem 15.1.1], a geometric ergodicity and a geometric decay in the tail of the return time distribution are equivalent conditions. □

Proof of Corollary 3

Proof. An unbiased estimate of $\nabla_{\mathbf{W}} \mathcal{L}_Z$ for one tour is obtained from $F^{(K_{\text{dyn}},R)}(\mathbf{W}, f)$ in eq. (4.6) of Theorem 4.2.3 with $f(\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial E(\mathbf{x}_n;\mathbf{W})}{\partial \mathbf{W}} - \frac{\partial E(\mathbf{y};\mathbf{W})}{\partial \mathbf{W}}$. Averaging the gradient of each tour over $R \geq 1$ tours gives the desired result. □

Source Code

My source code and detailed results are hosted at `https://github.com/PurdueMINDS/MCLV-RBM`.