JOURNAL OF UNDERGRADUATE RESEARCH

Caterpillar Quota Walk Sampling: A New Predictive Sampling Method for Spread Models in Networks

Caijun Qin
Computer Science

Faculty Mentor: Peter J. Dobbins, Department of CISE

Abstract

Network sampling identifies a subset of nodes and/or edges from a network, producing an induced graph. Exploration-based sampling algorithms grow the induced set from an initial node by iteratively traversing and accumulating unvisited neighboring nodes from the original network. Many exploration-based sampling algorithms suffer drawbacks that limit the sum of degrees of visited nodes and thus the number of high-degree nodes visited. Furthermore, a strategy may not adapt adequately to volatile degree frequencies throughout the original network architecture, which influences how deep into the original network an algorithm could sample. This paper proposes a novel, exploration-based network sampling algorithm called caterpillar quota walk sampling (CQWS) inspired by the caterpillar tree and is compared to four other algorithms. CQWS generates a spread model as its sample by visiting the highest-degree neighbors of previously visited nodes. For each previously visited node, the top proportion of highest-degree neighbors creates the next set of caterpillar trees. Tunable and trainable parameters allow CQWS to maximize the sum of the degrees of the induced graph from multiple trials when sampling dense networks. Test scenarios included both sparse and dense networks evaluated over all the algorithms. Results show CQWS performance was best over dense networks. Evaluating the degree sum results, CQWS consistently outperformed other algorithms when sampling both sparse and dense networks.

Keywords: spread model, network sampling, exploration-based sampling, caterpillar tree, lobster graph, caterpillar quota walk sampling

Introduction

Complex networks abundant within different community structures have far-reaching applications in modeling complex systems such as social media circles, food webs, international trade, and the World Wide Web (Ribeiro & Towsley, 2010). Theoretically, a network is synonymous with a graph from graph theory. This paper defines a network $G = (V, E)$, a nodes (vertex) set V, and edge set E. Each edge connects and establishes some relationship between two nodes in V. This definition specifically pertains to a simple undirected network, which is studied in this paper. For various applications, a node may present a social media account, a port in a shipping network, or cells in a biological network (Maiya & Berger-Wolf, 2010). An edge often stipulates a specific type of connection between two entities represented by nodes. For example, an edge may represent the strength of an activation potential across two neurons, the internet connection between two servers, or the status of social media connections (followers and friends). Depending on the application, both nodes and edges may carry qualitative and quantitative attributes to fulfill the network model's purpose. As opposed to regular graphs such as lattices and tessellation-based graphs, networks that model real-world phenomenon can have high variations in edge density, node density, clustering density, and other quantifiable attributes that delineate community structures throughout the entire network (Ribeiro & Towsley, 2010).

Network sampling establishes one way to analyze and mine a vast network for meaningful data. By excising a smaller subset of nodes and edges, executing algorithms and other computations on a representative subgraph eliminates computational and scalability constraints faced by the original network. Random sampling from statistical experiments can be conceptually transferred to network sampling as *random node (RN)* and *random edge (RE)* network sampling algorithms. For a budget of n nodes in a sample, RN selects n random nodes from V, while RE selects enough edges from E until the number of unique endpoints of edges (nodes) equals n (Ribeiro &Towsley, 2010).

However, large networks seldom have all nodes and edges initially accessible or at least feasibly reachable (Rozemberczki et al., 2020b). A common reason is saving extremely large networks in relatively slow-acess storage mediums.Furthermore, organizations use systems with limited memory and therefore can only load and view a small proportion of a stored network. This renders RN and RE unusable. Since a relatively small subset of V is feasibly accessible, sampling algorithms using graph traversal from already available nodes offer a more viable solution. Futrhermore, such exploration-based sampling algorithms can generate a sample proximal to specific nodes of interest. For example, if a new business seeks a location to set up shop, examining nodes (i.e. locations) close and reachable to nodes of interest (i.e. consumer communities) facilitates choosing the optimal location. The quality of sampled nodes can be evaluated by a metric that summarizes connectivity, clustering, degrees, or other characteristics of the sample. Many algorithms follow this general heuristic of starting at a single node s and iteratively adding unvisited nodes v in V adjacent to nodes already in the sample (Rozemberczki et al., 2020b). To compare different exploration-based sampling algorithms, establishing some time or resource bound B per sample creates an even field for comparisons. In this paper, a fixed sample size n is the bound, setting $B = n$.

The network sampling algorithms in this study are inspired by spread models from a network context. Conceptually, a spread model can simulate the spread of influence or flow of information from node to node (Kuikka, 2018). In a network, a spread model is simply an induced subgraph $S \subseteq G$ produced by traversing nodes from initial node s.Across many applications, node degree constitutes a value of interest. The degree of a node u, symbolically $\deg(u)$, is defined as the number of adjacent nodes to u. The set of adjacent nodes to $u$is the neighborhood of u, symbolically $N(u)$. Applying the concept of a spread model delivers insight on how to reach the highest-degree nodes proximal to s.

Two problems current exploration-based network sampling algorithms face are addressed in this paper. First, targeting high-degree nodes in S was not deeply addressed during the development of many sampling algorithms (Rozemberczki et al., 2020b). In many real-world applications, node degree constitutes a value of importance (Rozemberczki et al., 2020b). Actual meaning ranges from someone's social circle to the number of highways intersecting a city. Targeting high-degree nodes provides direct benefit when sampling networks that model these applications. Second, a network sampling algorithm may linger in proximity to the start node s rather than reach nodes farther away (Rozemberczki et al., 2020b). This renders S a weaker representation of G and forfeits opportunities to discover higher degree nodes farther away from s. Therefore, a larger spread of edge distances between sampled nodes and s indicates greater reachability by a tested algorithm.

This paper proposes a novel algorithm that attempts to fix these two issues. *Caterpillar Quota Walk Sampling (CQWS)* selects new nodes by degree. For each previously visited node v, unvisited neighbors in $N(v)$ that rank at or above a certain percentile by degree are visited. At each iteration, each newly visited node branches into new caterpillar trees rooted at v. Sample S thus becomes a *lobster graph* constructed from recursive branching of caterpillar trees off one another. Four other exploration-based sampling algorithms besides CQWS are analytically compared in performance regarding the two aforementioned problems,discussed in the Preliminaries section.

Preliminaries

Definitions and Notations

definition 1. $G = (V, E)$ is a *network* or *graph* with nodes (vertices) in vertex set V and edges in edge set E.

definition 2: S is a sample of G, which is an induced subgraph derived by sampling a subset of V and all edges with both endpoints in V.

definition 3: s is the chosen start node for a sampling algorithm based on traversing G and automatically in S initially. For the purpose of this paper, $s = 0$.

definition 4: $deg(v)$ is the degree of node v, the number of edges incident to v.

definition 5: $N(v)$ is the neighborhood of node v, the set of nodes adjacent to v.

definition 6: $RN(v)$ is the neighborhood of node v ranked by degree from highest to lowest.

definition 7: $dist(u, v)$ is the distance between nodes u and v, the number of edges in any of the shortest paths with endpoints u and v.

definition 8: A **caterpillar tree** is a tree that requires every vertex to be within 1 edge of a central path.

definition 9: A **lobster graph** is a graph such that can be transformed into a caterpillar tree after removing any set of leaf nodes.

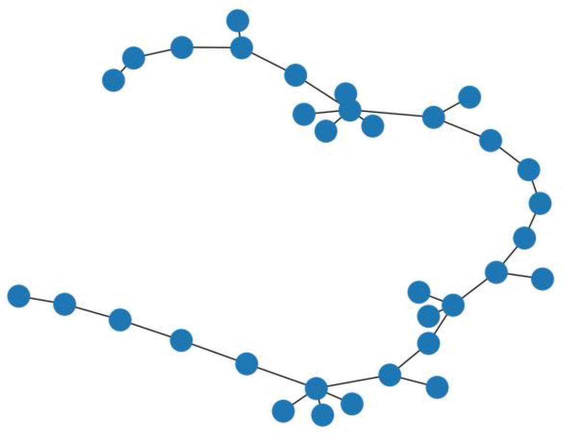


Figure 1. Caterpillar tree with 20-node backbone (central path) and p1=0.50 probability of edge offshooting off the backbone

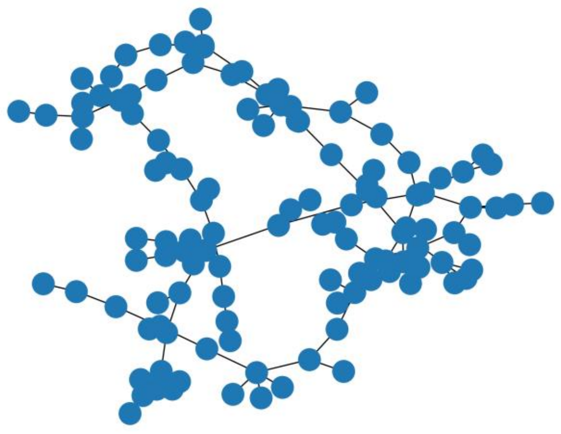


Figure 2. Lobster graph with 20-node backbone (central path) and p1=0.50 probability of edge offshooting off the backbone and p2=0.25 of offshoot edges >1 edge from backbone

Review of Other Network Sampling Algorithms

random walk sampling. *Simple random walk (SRW)* offers a computationally inexpensive method of network sampling. For each previously visited node v, choose any adjacent node $u \in N(v)$ with uniform probability (Li et al., 2019). SRW naturally predisposes bias towards higher degree nodes. However, the lack of any optimization criteria limits SRW on how to visit the next node.

common neighbor aware random walk sampling. A variant of random walk samplers, the *common neighbor aware random walk (CNARW)* focuses on reaching nodes less likely to be visited by SRW. For each previously visited node v, a random number $q \mid 0 < q < 1$ is selected. The value $q_{uv} = 1 - \frac{C_{uv}}{\min\{\deg(u), \deg(v)\}}$ is a threshold, where C_{uv} is the number of common neighbors between *u* and *v* (Li et al., 2019). A common neighbor w to u and v is any node adjacent to both u and v. The developed strategy helps the algorithm traversefrom *s* faster than SRW. This makes the distance distribution of S less likely to be constrained to a small value.

snowball sampling. *Snowball sampling* expands outward with a new layer of visited nodes per iteration. For $k \in \mathbb{N}$, each node v from the previous iteration selects *k* adjacent, unvisited nodes (Goodman, 1961).a random subset of $N(v)$ with size k is selected following a uniform probability distribution (Heckathon, 1997). Visiting a fixed number of neighbors per node per iteration presents a major drawback. In a dense network, a small k decreases the number of opportunities to sample large-degree nodes within G.However, a large k visits large-degree nodes more often while increasing computational demand. For each node v in the previous iteration, all neighbors are exhaustively visited when $k > \deg(v)$. The specific value of k becomes pointless when all neighbors of v are visited.

community structure expansion sampling. Using an optimization strategy, *community structure expansion sampling (CSES)* explicitly targets higher degree nodes during each iteration to expand the sample outwards from the current set of visited nodes. The algorithm uses an *expansion factor* $X(S) = \frac{|N(v)|}{|S|}$ (Maiya & Berger-Wolf, 2010). The algorithm expands the current sample S at a maximal rate while targeting high-degree nodes per iteration.Each iteration chooses the next node to visit that will append the most unvisited neighbors to S. This combination makes CSES perform well when evaluated by the degree sum and distance variance metrics.

frontier sampling. A multidimensional variant of random walk, *frontier sampling (FS)* operates as a group of m dependent SRW traversal operations (Ribeiro & Towsley, 2010). Suppose that queue L contains m nodes in G, symbolically $L(v_1, v_2, \dots, v_m)$. Select one node $u \in L$ with probability $\frac{\deg(u)}{\sum_{\forall v \in L} \deg(v)}$ where. Replace u with any neighbor $w \in N(u)$ and repeat the process until the sample reaches the required size (Ribeiro & Towsley, 2010). As a probabilistic algorithm, the node in L with maximum degree is most likely, but not guaranteed, to be visited over another node with lesser degree. Since FS essentially acts as RWS distributed across separate sampling processes taking turns, this algorithm is omitted from the experimental procedure for redundancy.

The time complexity of each algorithm from the experiment is given in Table 1. The time complexity is focused on each visited node during each iterative step rather completion of a sampling.

Table 1. Time complexity of each algorithm per visited node per step.

Network Sampling Algorithm	Time Complexity Per Visited Node Per
Random Walk Sampling	O(1)
Common Neighbor Aware Random Walk Sampling	O(V)
Snowball Sampling	O(1)
Community Structure Expansion Sampling	O(V^2)
Caterpillar Quota Walk Sampling	O(VlogV)

Proposed Method: Caterpillar Quota Walk Sampling

CQWS expands the graph, incorporating additional nodes by selecting from the upper end of $RN(v)$, where *v* represents each node visited in the previous iteration. The selection criteria depends on two thresholds q_1 and q_2 where $0 < q_1 < q_2 < 1$. During each iteration, CQWS partitions $RN(v)$ into two halves and selects the largest possible subset Q_1 from $RN(v)$ such that

$\Sigma_{u \in Q_1}^{n} \deg(u) \leq \left(\Sigma_{v \in N(v)} \deg(v)\right) * q_1$. Repeat this division of $RN(v)$ into two halves, but a larger subset of highest nodes in $RN(v)$ named Q_2 are selected such that $\Sigma_{u \in Q_2}^{n} \deg(u) \leq \left(\Sigma_{v \in N(v)} \deg(v)\right) * q_2$. Remove any nodes from Q_2 already in Q_1.More concisely, $Q_2 \coloneqq Q_2 \setminus Q_1$ is executed after initially filling the q_2 quota. In synopsis, Q_1 and Q_2 fulfill quotas for two consecutive groups of nodes in $RN(v)$ such that $\frac{\Sigma_{u \in Q_1} \deg(u)}{\Sigma_{v \in S} \deg(v)} < q_1$ and $\frac{\Sigma_{u \in (Q_1 \cup Q_2)}}{\Sigma_{v \in S}} < q_2$. Each iteration,each node $u \in (Q_1 \cup Q_2)$ joins sample S. However, nodes in Q_2 become inactive for future iterations and their neighbors do not undergo the algorithm. While each node in Q_1that each branches into new caterpillar trees, each node in Q_2 becomes an endpoint of a permanent 1-edge offshoot from its parent node v . Figure 3 compacts the proposed algorithm into pseudocode.

```
$S \coloneqq \{s\}$
$Q_1 \coloneqq \{s\}$
$Q_2 \coloneqq \{s\}$
FOR $v$ IN $L$:
    Create subset $Q_1$ from $RN(v)$.
    FOR $u$ IN $Q_1$:
        $S \coloneqq S \cup u$
        $L \coloneqq L \cup u$
    Create subset $Q_2$ from $RN(v)$.
    FOR $u$ IN $Q_2$:
        $S \coloneqq S \cup u$
REPEAT UNTIL $|S| = B$
```

Figure 3. Caterpillar quota walk sampling algorithm (CQWS)

Methodology

The hardware used for this work consists of 10th generation Intel i7 core and 16 GB of RAM, and computations were executed on a WSL Ubuntu 20.04 terminal. All code was written in Python version 3.8.5. To evaluate each algorithm, two synthetically generated randomnetworks, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, were sampled. Each network is a *gnm random graph*, a variant of Erdős–Rényi graph. Each network is a random selection out of all possible networks with n nodes and m edges. The parameters for G_1 are $n = 1000, m = 5000$ and G_2 are $n = 1000, m = 500000$, respectively. With equal node count, the disparity in the number of edges by a factor of $\frac{|V_2|}{|V_1|} = 100$ contrasts the performance of algorithms between sparse and dense networks. Initially, *SnowballSampler* and *CaterpillarQuotaWalkSampler* instances have undergone parameter tuning (Rozemberczki et al, 2020b; Qin, 2020). Each algorithm exceptCQWS was executed on G_1 and G_2 via instances of classes from the Python library *littleballoffur* (Rozemberczki et al, 2020a). Each sampling algorithm per network had been evaluated with two scoring metrics, degree sum and distance variance, yielding a total of four sampling executions per algorithm per metric per network. Scores are reported as a 4×2 tables, namely Table 1 and Table 2 under Results. The tables correspond to sampling on G_1 and G_2, respectively.

Scoring Metrics

degree sum. This is the frequency, or hit rate, of high-degree nodes. Since the exact degree dividing high-degree and low-degree nodes is subjective and application-specific, the sum of degrees for all n sampled nodes, $\Sigma_{u \in S}^{n} \deg(u)$, is employed as a holistic, application-agnostic measurement to indicate how strongly an algorithm targets higher degrees.

distance variance. This evaluates the spread of nodes in S by computing the distance variance $VAR\big(dist(u,s)\big)$, where $dist(u,s)$ is the edge countin any shortest path between u and s. The term “distance” here thus generally refers to the minimum edge count between any sampled node and s. This gauges how deep an algorithm can reach into G from s. Other statistics such as the mean, median, and mode may also convey the reachability of an algorithm. However, the variance is a more stable and more representative option than the three aforementioned point estimates. The mean is susceptible to outliers, as a sampled node extremely far from s gives an inaccurate value of the true center of distances. The median, although more robust against outliers than the mean, is determined by the ordinal position of the middle element in the distance distribution andfails to measure the spread of distances. . Similarly as a point estimate, the mode also does not incorporate the distances of all sampled nodes and does not capture the spread of distances. Compared to the three aforementioned metrics, computing the variance accounts for all values in the distance distribution and retains consistency across multiple trials.

Parameter Tuning

Parameters for snowball sampling and CQWS have each been tuned by an *NetworkSamplerTuner* instance from *NetworkSampling.py* (Qin, 2020). For snowball sampling, the only tunable parameter is k. The considered test range for k was all positive integers in $[2,20]$. With CQWS, the tunable parameters are q_1 and q_2. The test range for q_1 consisted of the Cartesian product $\{0.00, 0.01, 0.02, 0.03, \dots, 0.25\} \times \{0.00, 0.01, 0.02, 0.03, \dots, 0.25\}$, and the test range for q_2 consisted of the Cartesian product $\{0.25, 0.26, 0.27, 0.28, \dots, 0.50\} \times \{0.25, 0.26, 0.27, 0.28, \dots, 0.50\}$. Tuning was done for each algorithm-metric pair before sampling. Both algorithms utilized GridSearchCV from the Scikit-Learn library.

Plots

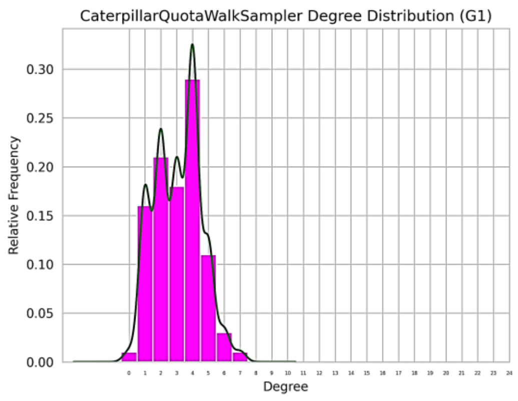


Figure 4. Degree distribution of sample from sparse graph (G1) using CQWS

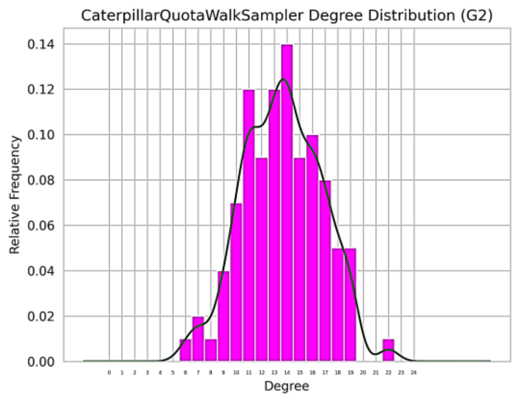


Figure 5. Degree distribution of sample from dense graph (G2) using CQWS

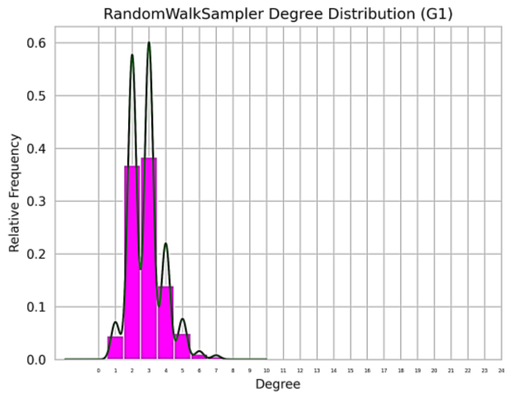


Figure 6. Degree distribution of sample from sparse graph (G1) using RWS

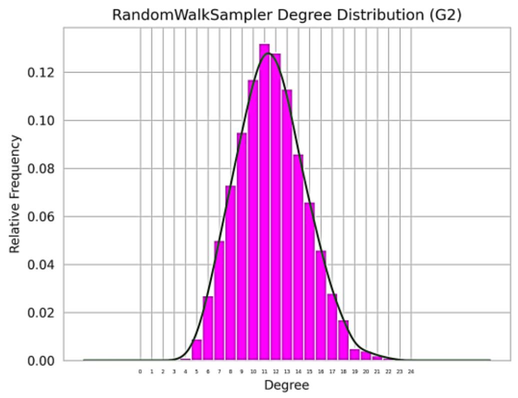


Figure 7. Degree distribution of sample from dense graph (G2) using RWS

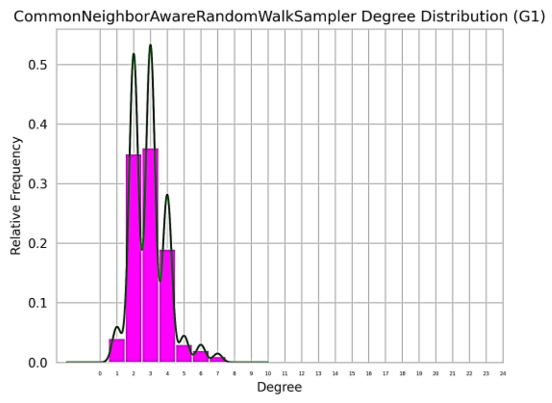


Figure 8. Degree distribution of sample from sparse graph (G1) using CNARW

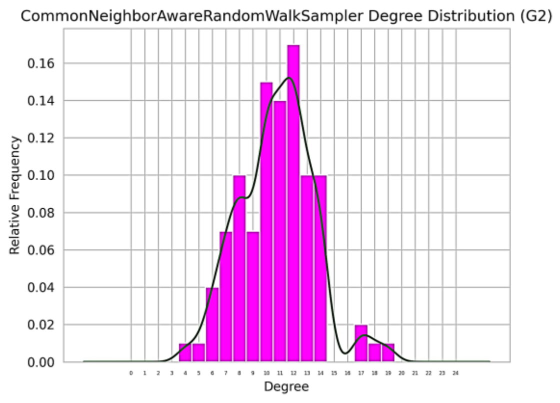


Figure 9. Degree distribution of sample from dense graph (G2) using CNARW

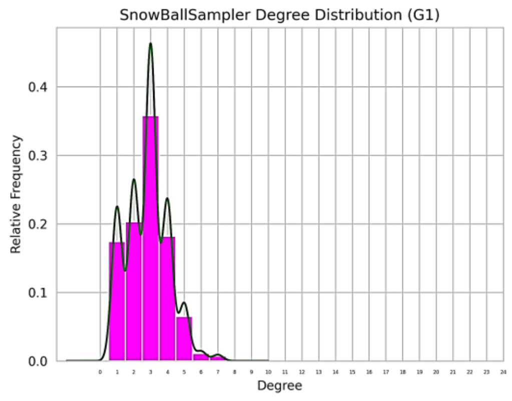


Figure 10. Degree distribution of sample from sparse graph (G1) using SnowballSampler

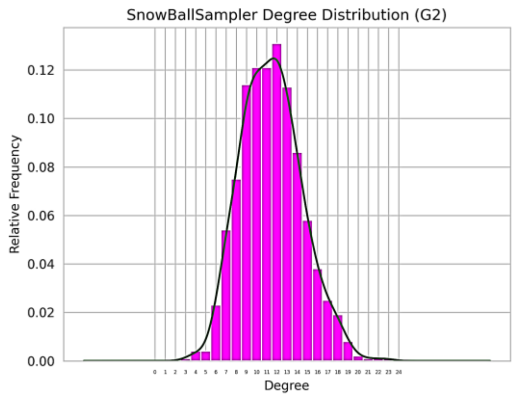


Figure 11. Degree distribution of sample from dense graph (G2) using SnowballSampler

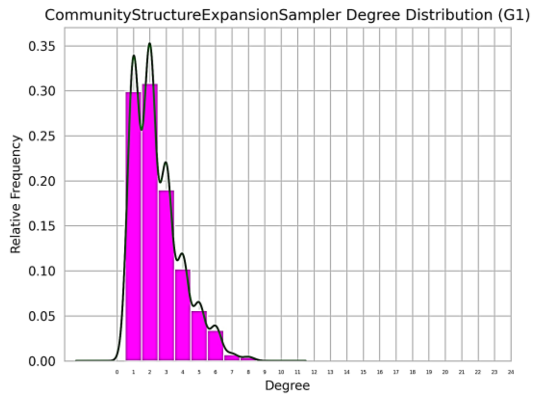


Figure 12. Degree distribution of sample from sparse graph (G1) using CSES

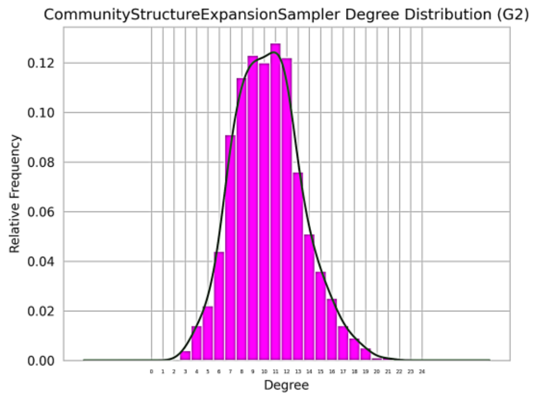


Figure 13. Degree distribution of sample from dense graph (G2) using CSES

For each algorithm and each network from $\{G1, G2\}$, a degree distribution histogram is plotted. These histograms are given in Figures 4-13. Each histogram visually approximates the center and spread of degrees of sampled nodes. Across five algorithms and networks, there are $5 \times 2 = 10$ histograms. Note that each histogram uses data based on nodes accumulated over 10 trials on the same network.

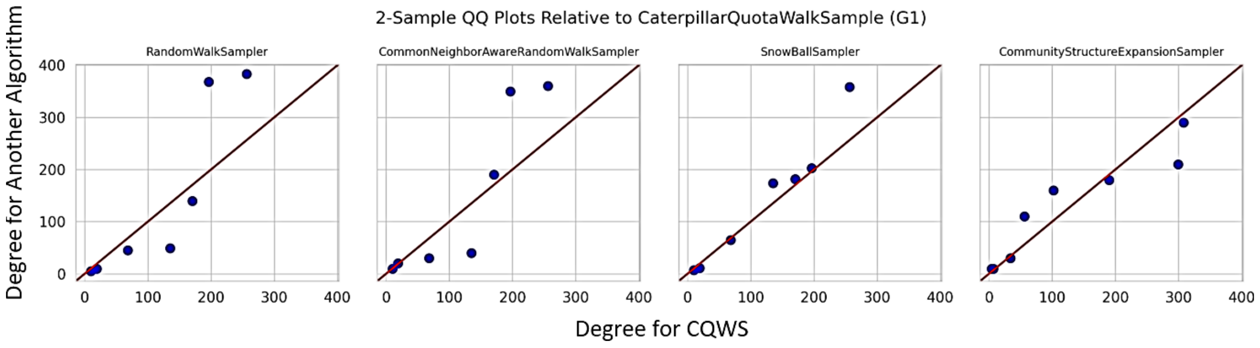


Figure 14. 2-sample QQ plots of degree distributions of CQWS (x-axis) and other sampling algorithms (y-axis) sampled on sparse graph (G1)

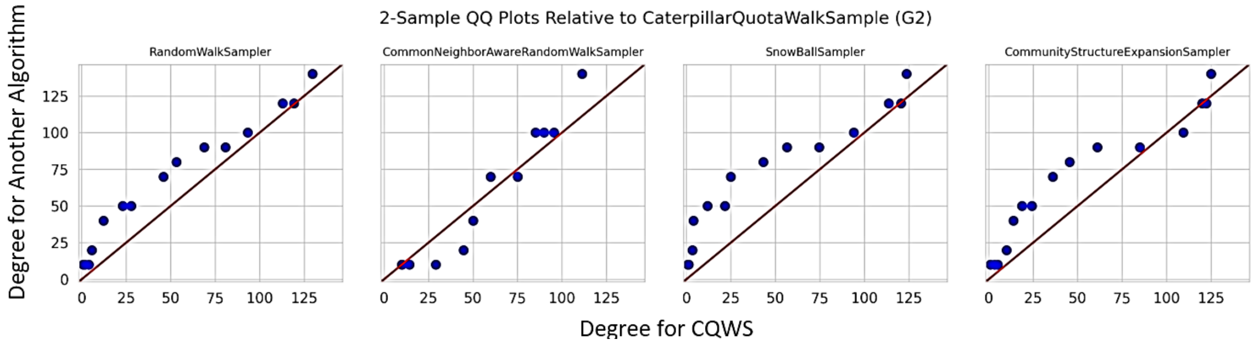
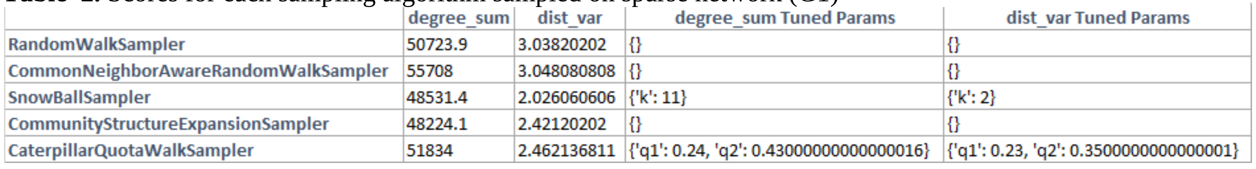


Figure 15. 2-sample QQ plots of degree distributions of CQWS (x-axis) and other sampling algorithms (y-axis) sampled on dense graph (G2)

In Figures 14-15, 2-sample quantile-quantile (QQ) plots are shown for each pair of network sampling algorithms that contain CQWS. These plots compare the skeweness of CQWS with every other algorithm and highlights whether CQWS has an advantage or disadvantage in sampling nodes of higher degree relative to another algorithm. Note that both x-axis and y-axis use degree rather than quantile of each degree distribution as units.

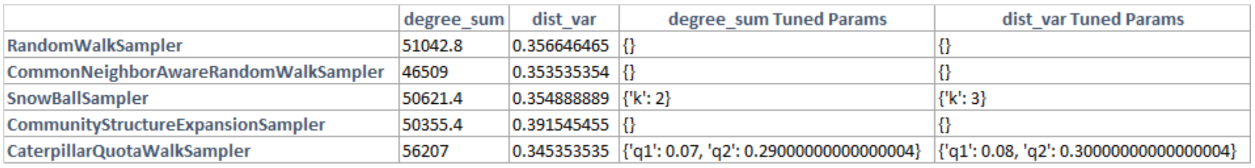
Results

Table 2. Scores for each sampling algorithm sampled on sparse network (G1)



	degree_sum	dist_var	degree_sum Tuned Params	dist_var Tuned Params
RandomWalkSampler	50723.9	3.03820202	{}	{}
CommonNeighborAwareRandomWalkSampler	55708	3.048080808	{}	{}
SnowBallSampler	48531.4	2.026060606	{'k': 11}	{'k': 2}
CommunityStructureExpansionSampler	48224.1	2.42120202	{}	{}
CaterpillarQuotaWalkSampler	51834	2.462136811	{'q1': 0.24, 'q2': 0.43000000000000016}	{'q1': 0.23, 'q2': 0.3500000000000001}

Table 3. Scores for each sampling algorithm sampled on dense network (G2)



	degree_sum	dist_var	degree_sum Tuned Params	dist_var Tuned Params
RandomWalkSampler	51042.8	0.356646465	{}	{}
CommonNeighborAwareRandomWalkSampler	46509	0.353535354	{}	{}
SnowBallSampler	50621.4	0.354888889	{'k': 2}	{'k': 3}
CommunityStructureExpansionSampler	50355.4	0.391545455	{}	{}
CaterpillarQuotaWalkSampler	56207	0.345353535	{'q1': 0.07, 'q2': 0.29000000000000004}	{'q1': 0.08, 'q2': 0.30000000000000004}

Discussion

CQWS achieved the highest degree sum in 1000-node sample averaged over $n = 10$ trials compared to all other sampling algorithms for G_2 (Table 3). Due to its trainable parameters relevant to node selection, a narrow window $(q1 = 0.07, q2 = 0.29)$ of highest degree neighbors that CQWS visits each iteration is ideal for sampling networks that have dense connections (high degree density). However CQWS performed more poorly against other algorithms when sampling the sparse G_1 network (Table 2) relative to the dense G_2 network (Table 3). For G_1, CQWS performed worse than CNARW by a larger margin than the amount of improvement in degree sum over any of the other three algorithms in Table 2. While for G_2, CQWS outperformed all algorithms in degree sum and held the largest margin of improvement against its runner-up, Random Walk, compared to any other algorithm and its corresponding runner-up in Table 3. This performance disparity is further evidenced by comparing Figures 14 and 15, based on a repeat of the same sampling trials for G_1 and G_2 shown in Tables 2 and 3. As Figure 14 shows, CQWS only outperformed one algorithm (Snowball Sampling) on G_1. When evaluated on the G_2 network in Figure 15, CQWS clearly outperformed all other algorithms except CNARW. A QQ plot displays a right-skewed distribution if points start above the 45° line, close in towards the line, and veer back up along the positive x-axis. Right skewness conveys that the algorithm on the y-axis obtained a lower mean degree (and other measures of center of a distribution) across the 10 sampling trials than CQWS. Specifically, this phenomenon occurred when comparing CQWS against CSES in Figure 14 and against each algorithm besides CNARW in Figure 15.

Comparing CQWS against itself across both G_1 and G_2 also affirmed CQWS's suitability for dense networks relative to sparser networks. The degree histogram of CQWS for G_2 in Figure 5 showcases a center of mass leaning more to the right (i.e. towards a higher degree) than the degree histogram of CQWS for G_1 in Figure 4.

For distance variance, CQWS achieved better performance than half of the other algorithms when sampling G_1, evidenced in Table 1. However, CQWS performed the worst when sampling G_2 as shown in Table 2. In a dense network, there can be high-degree neighbors of visited nodes that are closer to the start node and have not been discovered earlier. By targeting high-degree nodes, there would be some backtracking during sampling because high-degree nodes become more closely packed. This indicates a trade off between degree sum and distance variance.

Conclusion

Comparing the degree sum of several exploration-based network sampling algorithms against the proposed CQWS yielded deep insight into how both criteria and the connectivity of a network affects the sampling behavior in a network. This paper not only compared the performance of four existing algorithms against a newly proposed one, but the results also suggest how algorithms applying strategic criteria such as CQWS and CNARW outperformed algorithms using purely probabilistic rules to sample nodes. Although snowball sampling employs a tunable parameter, the algorithm still performed poorly because it is not deterministic (i.e., sampling the same network twice yields different results). Despite also being nondeterministic, CNARW employs an intelligently designed rule for choosing nodes to visit and thus performed better than the random walk and snowball sampling. Finally, CQWS has the advantage of incorporating the most tunable parameters and being a deterministic algorithm.

For future work, the most viable improvements on the algorithm would be using counting sort algorithm to order neighboring nodes by degree. This will improve the worst-case time complexity from $O(VlogV)$ to $O(V)$. Even this improved time complexity for CQWS does not exceed the worst-case time complexity of some other algorithms (random walk and snowball sampling). However, the effective performance of CQWS on targeting high-degree nodes can be exploited as a second step in a combination of sampling algorithms. In the first step, a computationally cheaper algorithm can quickly generate a preliminary sample on which CQWS uses for parameter tuning. The tuned CQWS model can then performantly generate a larger second sample to identify much more high-degree nodes in the locality of a node of interest. With its focus on collecting high-degree neighbors, CQWS is suited for tasks such as recommending social media connections, identifying local communities of resources online, and prioritizing logistical tasks drawn from a dependency graph.

Acknowledgements

This research was made possible by the generous funding of University Scholar's Program under the University of Florida. The author also wishes gratitude towards Dr. Peter J. Dobbins for countless hours of mentorship and advice over the past academic year of research.

References

Goodman, L. A. (1961). Snowball Sampling. *The Annals of Mathematical Statistics*, *32*(1), 148–170. https://doi.org/10.1214/aoms/1177705148

Heckathorn, D. D. (1997). Respondent-driven sampling: A new approach to the study of hidden populations. *Social Problems*, *44*(2), 174–199. https://doi.org/10.2307/3096941

Kuikka, V. (2018). Influence spreading model used to analyse social networks and detect sub-communities. *Computational Social Networks*, *5*(1), 12. https://doi.org/10.1186/s40649-018-0060-z

Li, Y., Wu, Z., Lin, S., Xie, H., Lv, M., Xu, Y., & Lui, J. C. S. (2019). Walking with perception: Efficient random walk sampling via common neighbor awareness. *Proceedings - International Conference on Data Engineering*, *2019-April*, 962–973. https://doi.org/10.1109/ICDE.2019.0009

Maiya, A. S., & Berger-Wolf, T. Y. (2010). Sampling community structure. *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, 701–710. https://doi.org/10.1145/1772690.1772762

Qin, C. (2020). Caterpillar-Quota-Walk-Sampling [Computer Software] Retrieved from https://github.com/Fennec2000GH/Caterpillar-Quota-Walk-Sampling

Ribeiro, B., & Towsley, D. (2010). Estimating and sampling graphs with multidimensional random walks. *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, 390–403. https://doi.org/

Rozemberczki, B., Aspert, N., Ricaud, B., & Kiss, O. (2020a). Littleballoffur (Version 2.1.7) [Computer software]. Retrieved from https://github.com/benedekrozemberczki/littleballoffur

Rozemberczki, B., Kiss, O., & Sarkar, R. (2020b). Little Ball of Fur: A Python Library for Graph Sampling. *International Conference on Information and Knowledge Management, Proceedings*, 3133–3140. https://doi.org/10.1145/3340531.3412758