Sampling Large Graphs for Anticipatory Analytics

Lauren Edwards, Luke Johnson, Maja Milosavljevic, Vijay Gadepally, Benjamin A. Miller
Lincoln Laboratory
Massachusetts Institute of Technology
Lexington, MA 02420
{Lauren.Edwards, Luke.Johnson, Maja.Milosavljevic, vijayg, bamiller}@ll.mit.edu

***Abstract*—The characteristics of Big Data - often dubbed the 3V's for volume, velocity, and variety - will continue to outpace the ability of computational systems to process, store, and transmit meaningful results. Traditional techniques for dealing with large datasets often include the purchase of larger systems, greater human-in-the-loop involvement, or through complex algorithms. We are investigating the use of sampling to mitigate these challenges, specifically sampling large graphs. Often, large datasets can be represented as graphs where data entries may be edges, and vertices may be attributes of the data. In particular, we present the results of sampling for the task of link prediction. Link prediction is a process to estimate the probability of a new edge forming between two vertices of a graph, and it has numerous application areas in understanding social or biological networks. In this paper we propose a series of techniques for the sampling of large datasets. In order to quantify the effect of these techniques, we present the quality of link prediction tasks on sampled graphs, and the time saved in calculating link prediction statistics on these sampled graphs.**

I. Introduction

The volume of information passing through today's data systems often outpaces our ability to compute meaningful results. Storing all of this data will quickly stress the limits of storage systems. Memory issues also accompany this volume of data, as most useful predictive analytics require loading all relevant data into memory. Up-front data reduction, including sampling, would help to alleviate both of these issues, but sampling will have some effect on the results of any predictive analytic done on the data. The focus of our study is to explore the effects of a variety of sampling techniques on a specific predictive analytic and type of data: link prediction on large graphs.

Large graphs can often represent the data we collect, such as social networks, computer networks, epidemiological data, or any data that shows relationships between entities. One particularly interesting anticipatory analytic is link prediction [1], where we predict that two entities in a graph will form a relationship in the near future. Link prediction has been used among a diverse array of applications [2], including prediction in networks of web pages [3] and social networks [4]. The methods have also been applied to collaborative filtering for recommender systems [5]. Across these varied applications, accurately forecasting future activity is important, and link prediction is a prime example of an anticipatory analytic.

Sampling methods on large graphs have previously been investigated, but not in the context of predictive analytics. In [6], the authors describe a variety of sampling techniques and quantify the effect of sampling on the preservation of several graph features, including degree distributions, distributions of singular values, and hop-plots. The authors in [7] propose a random sampling technique in order to approach the graph partition optimization problem. In [8], the authors present a series of sampling techniques to perform frequent subgraph mining. Random walk and jump sampling methods are discussed in [9], also for subgraph mining applications. Vertex-based sampling methods, such as random node sampling [6], [8] or wedge sampling [10], have also been shown to preserve various graph features or clustering coefficient of a graph. Other techniques for sampling to preserve graph clusters are discussed in [11]. These studies highlight the growing interest in sampling for big data applications.

In this study, we sample several simulated graphs with different properties and perform link prediction. We first describe the link prediction methods we use in Section II, followed by a description of sampling methods in Section III. Section IV describes the methods we used to generate simulated graphs and the steps we take to sample and perform link prediction on these graphs. Finally, we show and discuss the results from these simulations in Section V.

II. Link Prediction Methods

There are two modes of operation for link prediction that are used in the experiments in this study. The first involves computing statistics of a graph based on data up to a certain point in time, and scoring the possible edges in the near future [12]. We implemented three link prediction methods in this mode of operation: common neighbors, Jaccard's coefficient, and low-rank approximation. All methods were chosen for both their computational efficiency and their ability to outperform chance. Both common neighbors and Jaccard's coefficient are based on vertex neighborhoods and work based on the intuition that two nodes are more likely to form a link in the future if they have many neighbors in common.

Distribution A: Public Release

This work is sponsored by the Intelligence Advanced Research Projects Activity (IARPA) under Air Force Contract FA8721-05-C-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The common neighbors statistic is simply the number of neighbors two vertices have in common and is found by calculating the inner product of the adjacency matrix of the graph in question. Jaccard's coefficient starts with the common neighbors statistic and then takes into account the degrees of each vertex:

$$J(i,j) = \frac{|\Gamma(i) \cap \Gamma(j)|}{|\Gamma(i) \cup \Gamma(j)|} \quad (1)$$

where $\Gamma(i)$ and $\Gamma(j)$ are the set of neighbors of vertices i and j, respectively. Here we are taking the number of neighbors i and j have in common, and dividing by the total number of neighbors of i and j [12].

Our final link prediction method for this mode of operation is the low-rank approximation. In this method, using small values of k, we compute the rank-k matrix, M_k, that best approximates the full adjacency matrix, M, by performing a singular value decomposition on the adjacency matrix. We then score the potential edges by calculating the inner product on M_k, which is essentially the common neighbors calculation on a low-rank matrix, M_k, instead of the full matrix M. The low-rank approximation also is also particularly good for this study because it applies significant noise reduction to the original large graph, while preserving most of the structure in the graph [12].

We can make use of more fine-grained temporal information through the use of another mode of operation. Instead of predicting links in the near future based on one graph representing the relevant past, we predict links at specific times in the future given a sequence graphs at different time points in the past. Here, performing a tensor decomposition of the data will allow us to predict future links by extrapolating the temporal factors [13]. This method provides a higher degree of temporal resolution than performing edge-scoring or aggregate statistics on the same graph.

This method uses a three-dimensional tensor, where each cell $T(i,j,t)$ represents a connection between two vertices i,j at a point in time t. Because this information is represented using a tensor, it is possible to perform CP decomposition to factor out the rank-one matrices that represent the temporal patterns of the graph. The Holt-Winters additive forecasting method, allows us to create a new rank-one matrix, where each value in the matrix represents a predicted time step. This rank-one matrix is used to construct a new tensor containing predicted edges for future time steps. In theory, we could predict edges infinitely into the future, but the performance will diminish the further out we predict [13].

III. Sampling Methods

In the paper, we propose a variety of sampling methods based on a search of techniques in the literature. For the purpose of the discussion of sampling methods, we describe techniques applied to static graphs (i.e, graphs that are not changing through the sampling process). Consider a graph $G = (V, E)$ where V and E correspond to the vertices and edges respectively. The number of edges in G is given by $|E|$ and the number of vertices given by $|V|$. The degree of a vertex $v_i \in V$ is the sum of the edges incident to v_i and is represented as $d(v_i)$. Sampling a graph, within the context of this paper, is to generate a new graph $G' = (V, E')$ where $E' \subseteq E$ and $|E'| \leq |E|$. The ratio of $|E|/|E'|$ is referred to as the desired sampling factor k. For example, a sampling factor of $k = 4$ implies that the sampled graph G' will have 25% of the total edges in G. Given a sampling factor and graph, there are a variety of methods in which the sampling may take place.

A. Random Edge Sampling

Random edge sampling is the simplest form of sampling a graph and requires the least amount of computation. Given a desired sampling factor k, the sampled graph $G' = (V, E')$ is determined by sampling an edge with probability $\frac{1}{k}$. The expected number of output edges, $|E'|$ will be approximately $k|E|$. Intuitively, this sampling method is suitable for tasks in which the relative degree of nodes is important, such as ranking popular users in a social media graph, as it tends to bias high-degree vertices [6].

B. Popularity Based Sampling

A slightly more involved sampling method makes use of the degrees of the vertices between which an edge is present. The idea is that edges between popular vertices (vertices with high degree) should be sampled at a lower rate than edges between unpopular vertices. This corrects for the bias for high-degree vertices introduced by random sampling. In order to implement this sampling method, each edge e_{ij} with connecting vertices v_i and v_j is kept with a probability proportional to the degrees of v_i and v_j, $d(v_i)$ and $d(v_j)$ respectively. Specifically, for an edge e_{ij} we compute the probability of sampling e_{ij} as $\frac{1}{\sqrt{d(v_i)d(v_j)}}$. Thus, if the edge is between two popular nodes, the probability that we keep the edge is relatively low.

C. Random Area Sampling

Random area sampling [8] is a "snowball" sampling method in which a set of random seed vertices are selected and areas (edges and vertices) connected to these seed vertices are kept, while the others are discarded. For a starting set of vertices V', all the vertices adjacent to V' are added to V' and all the edges are kept until the desired size of the sampled output is reached. Determining the number of input seed vertices can be done by estimating the maximum size of the connected areas and selecting the number of vertices that would result in the desired output graph size. Random area sampling works well in instances where we want to maintain common neighbors or sections of the original graph in their entirety.

D. Random Jump Sampling

Random walks are a popular method for sampling large graphs. In a random walk, certain seed vertices are selected and neighbors are chosen randomly. The edges connecting the selected neighbors form the set of edges in the sampled graph, and the neighbors become the new seed vertices. However,

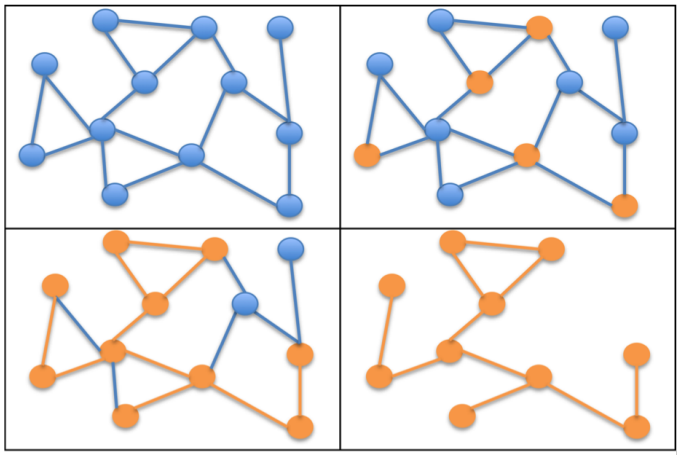

Fig. 1. Example of wedge sampling being performed on a 12-node graph. A set of vertices are chosen at random and wedges are formed by randomly selecting 2 edges incident to the selected vertices to generate the final sampled graph.

this technique may get stuck at vertices with no out-edges or at vertices that are a part of isolated components. In order to avoid this problem, we use the random jump sampling. Random jump sampling is similar to random walk sampling except that along a given walk, the walk may terminate and "jump" to a random vertex in G. Specifically, one picks a set of seed vertices, V', with degree greater than one, and either continue the walk by selecting neighbors of the vertices in V' or jump to a random vertex in V with probability p. The value of p is often set to 0.15 [6].

E. Wedge Sampling

A variant of the wedge sampling algorithm proposed in [10] has also been implemented. In this algorithm, a number of random vertices from V are selected and two incident edges to each of these vertices are selected at random to create wedges centered at the vertices from V. This method is then repeated until the desired output graph G' is generated, where each of the wedge edges form the edges in E'. An example of wedge sampling is depicted in Fig. 1. Wedge sampling has been shown to preserve the clustering coefficient and similar properties of a graph [10].

IV. Experimental Setup

In this paper, we show the results of link prediction in several simulated dynamic graphs, each with about 10,000 vertices. Each graph is based on the Block Two-Level Erdős–Rényi (BTER) model [14], which was developed to address shortcomings of current graph generators: specifically, the lack of clustering behavior. This model begins by splitting a vertex set into many subsets of various sizes, and making each of these subsets densely connected. The subsets are then randomly connected to one another, with external connections being more likely for larger clusters. The generative model is parameterized by a degree distribution and the average clustering coefficient of a vertex for each possible degree. We consider an undirected version of the graph with no self-loops, where a graph is made undirected via the "clip-and-flip" procedure suggested in [15] (i.e., the upper-triangular part of the adjacency matrix is maintained when it is made symmetric).

To simulate growth of the network, the graph at time $t+1$ is created based on the graph at time t. Specifically, to determine whether v_i and v_j in V will share an edge in $G(t+1)$, the number of common neighbors of v_i and v_j in $G(t)$ are computed, and normalized by $\sqrt{d(v_i)d(v_j)}$ (this is the cosine similarity between the rows of the adjacency matrix associated with v_i and v_j). New edges are then added with probability proportional to these values. This will account for about 70% of the new edges at time $t+1$, and is meant to simulate connections made between a large proportion of mutual acquaintances. The other 30% are created by generating a new BTER graph with the same parameters, and randomly selecting a subset of the edges. This accounts for the possibility of "chance" establishment of connections, without necessarily having many present connections in common.

For each graph, we generate 10 time steps and predict new edges in the 11th. We consider two different parameterizations of the BTER model, where one has a heavier tail in its degree distribution and a higher average degree than the other. One initial graph is generated with an average degree of approximately 17 and a powerlaw exponent of about -1.4 (i.e., the number of nodes with degree i is approximately proportional to $i^{-1.4}$), while the other initial graph's expected degree is roughly 34, with a powerlaw exponent of about -1.8. For each parameterization, we considered 10 random instantiations.

For each of the 5 sampling methods, we sample the graphs at rates where the number of edges is reduced by a power of two, from $1/2$ to $1/64$. We set the number of seed vertices in the random jump to be approximately $0.15|E'|$, or the probability of a jump times the number of desired edges in the sampled graph. Sampling is performed on the integrated graph over the entire time period (i.e., all edges that are present over the 10 training time steps), and the temporal activity records of the sampled edges are preserved. In both the low-rank approximation and tensor link prediction statistics, we used rank-10 matrices and tensors, respectively. Predictive performance is evaluated based on the receiver operating characteristic (ROC) curve over all possible edges where, over the training period, (1) the associated vertices were active, and (2) there was not already an edge present. In order to analyze the tradeoffs the different sampling techniques offer, we also recorded the amount of time required to compute the prediction statistic. All sampling and link prediction was done using the MIT SuperCloud [16] architecture at the MIT Lincoln Laboratory. This system consists of approximately 300 nodes with dual-socket 16-core AMD processors and 128 GB of RAM per node.

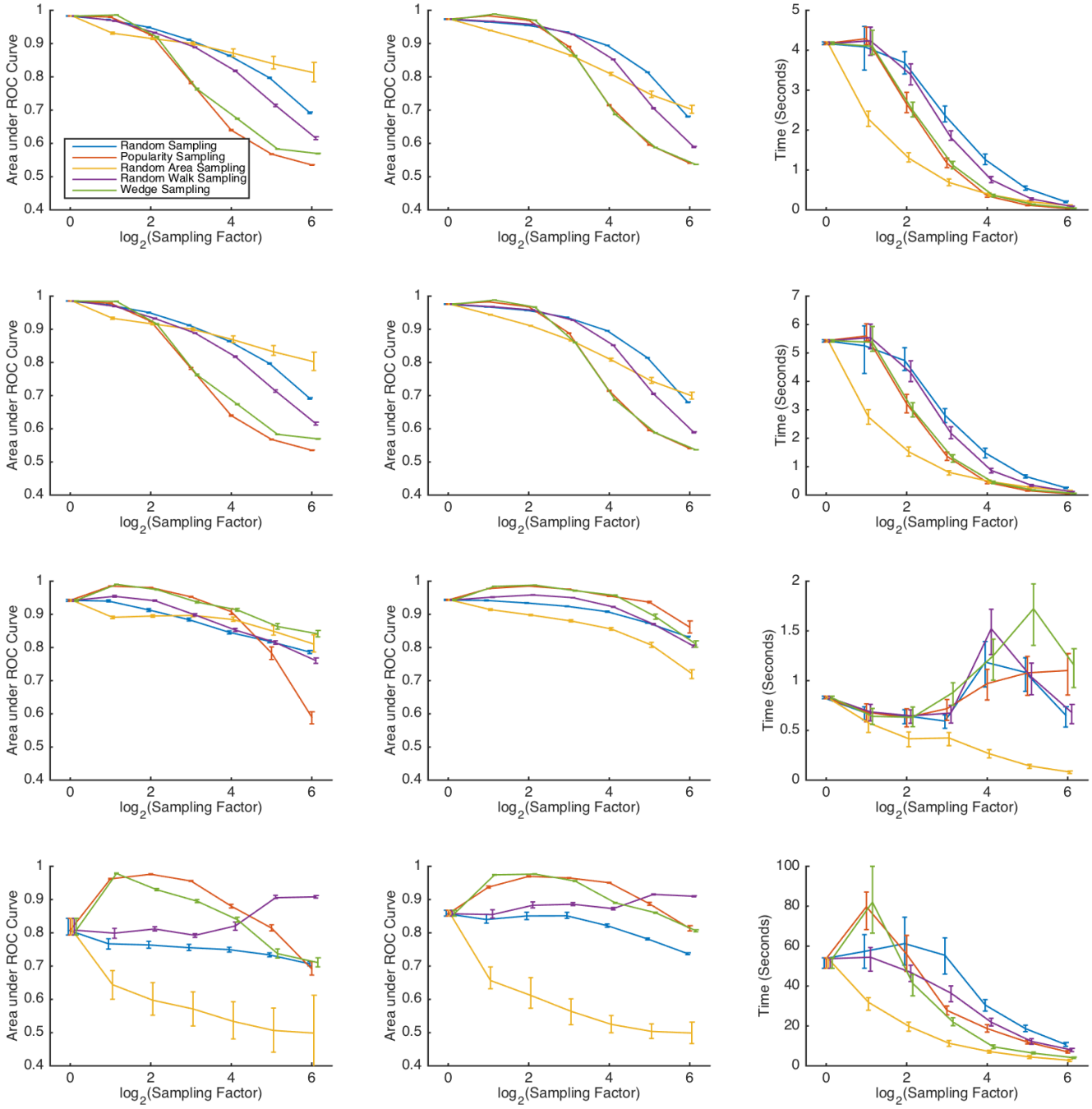

Fig. 2. Performance and timing for running various link prediction methods over two types of graphs. Each row shows results for the common neighbors, Jaccard's coefficient, low-rank approximation, and tensor link prediction methods, respectively. The first two columns contain plots of the area under the ROC curve across all trials of light-tail (first column) and heavy-tail (second column) distribution graphs. The third column shows the time to compute the link prediction statistic on all trials of the heavy-tail distribution graphs, which had timing that was representative for all graph types.

V. Results and Discussion

Detection performance using all link prediction methods and all sampling techniques is illustrated in Fig. 2. We run the link prediction algorithm for each sample and computed the ROC curve. The area under the ROC curve (AUC) is calculated to summarize detection performance, where an AUC of 1 indicates perfect detection and an AUC of 0.5 indicated that the statistic is powerless for discrimination. For each of the 10 instantiations of the two graphs, performance is averaged over all samples taken. In addition, the error bars in the plots show performance at the 25th and 75th percentiles.

While prediction performance primarily decreases as the

graphs are more aggressively sampled, there are some interesting phenomena that occur with some of the methods. First, with a few exceptions, performance with popularity-based sampling and wedge sampling track each other fairly closely, and in fact performance in both *increase* slightly at low sampling levels. We suspected this may be due to more vertices becoming isolated after sampling, since prediction is only scored for vertices that are active during the training period. Upon closer inspection, however, the number of isolated vertices using these methods did not substantially increase until sampling to 1/32–1/64. This phenomenon may be due to a greater number of false positives being associated with high-degree vertices in link prediction on the unsampled graph. After the slight increase in performance, the common neighbors and Jaccard's coefficient had a steep descent in performance in the popularity-based and wedge sampling, which slows at the 1/64 sampling rate. This may be due to the high number of isolated vertices left at this rate. The percentage of isolated vertices in the unsampled light- and heavy-tailed graphs were approximately 25% and 14%, respectively. Popularity sampling at the 1/64 level brought this to 45% and 19% on average, and wedge sampling brought the light-tailed graph to about 30% at the 1/64 level, but did not increase the number of isolated vertices significantly in the heavy-tailed graph.

The similarity in performance between these two methods is likely due to the fact that both sample fewer edges from higher-degree vertices than with random edges sampling. The largest deviation in performance between the two methods is in the most aggressive sampling levels for low-rank inner product on the light-tailed graph. This may be due to wedge sampling finding two-hop paths, which, if they are not already closed, have a higher-than-average chance of creating a new edge between the external vertices.

Random edge sampling and random jump sampling also have similar performance characteristics to one another. One interesting result is that random edge sampling outperforms random jump sampling for link prediction based on common neighbors and Jaccard's coefficient, but the opposite is typically true for the low-rank matrix statistics (low-rank approximation and tensor decomposition). This could be due to the relationship between random walks on the graph and the space of the eigenvectors of the adjacency matrix [17].

The remaining sampling method, random area sampling, has a different performance characteristic. For tensor decompositions, performance degrades extremely quickly, and has significant variation over samples. For the low-rank approximation, performance is somewhat similar to the other methods. For common neighbors and Jaccard's coefficient, however, we see a faster initial reduction in performance. This reduction remains steady as fewer edges are retained and eventually outperforms the other methods. This may be due to the significant number of isolated vertices produced by random area sampling. At the 1/8 sampling level, where most other sampling methods begin to decrease in performance, random area sampling leaves more than 50% of the vertices in each graph isolated.

Between the lighter- and heavier-tailed graphs, results are similar with a few exceptions. Random area sampling performs slightly better in graph with the lighter-tailed degree distribution for all of the methods based on local statistics from [12]. This could be because, with a lighter-tailed degree distribution, more of the vertices will have low degree, and therefore the sample will remain more localized when using this method. The unsampled light-tailed graph has more isolated vertices than the heavy-tailed graph, which is consistent with this interpretation.

There are also several interesting aspects of the running time results. (Results are shown for the heavier-tailed graph, but the trends are similar for those with lighter tails.) For both common neighbors and Jaccard's coefficient, all methods except random area sampling reduce their running time rather slowly until far fewer edges are retained. After this point, there is a steady decrease in running time, as there is with random areas at all sampling levels. The steeper descent in timing for random area sampling most likely is due to the larger number of isolated vertices that it leaves behind. The low-rank approximation has a strange timing characteristic, where the timing for samples with fewer edges have seemingly erratic trends. This may be due to changes in the distribution of the eigenvalues, which can alter the convergence rate of the eigensolver when computing the 10-dimensional subspace. Random area sampling does not follow this trend, where the number of isolated vertices may reduce the dimensionality of the adjacency matrix enough to reduce the time to compute the 10-dimensional subspace. Finally, the tensor decomposition method, which uses the Matlab Tensor Toolbox [18], sees a greater stall in the timing decrease in the methods other than random area sampling.

VI. Conclusion

In our experiments, we sampled two types of graphs using five sampling techniques and performed four link prediction methods. We found the performance of both popularity based and wedge sampling were very close. They were characterized by a small initial increase at low sampling rates (1/2–1/8), followed by a sharp decrease at higher sampling rates. Performance for random edge and random jump sampled similarly seemed to pair, but we saw no initial increase in performance, and a slower decrease in performance at higher sampling rates. The performance of random area sampling saw the smallest decrease in all link prediction methods except the tensor method, however this may be due to the large number of isolated vertices that remained after sampling, which were not taken into account for link prediction performance. As expected, sampling the graph decreased the amount of time to compute the link prediction statistic for most methods. The exception is the low-rank approximation, where we suspect a change in the distribution of eigenvalues due to sampling slows down the calculation of low-rank subspace.

For the task of link prediction, we found that both random edge and random jump sampling seemed to have the most

consistent results, even at high sampling rates. We would propose that, if a high sampling rate were required for a link prediction task, either random edge or random jump sampling would be the best choice of these five. We would also like to note that both popularity-based and wedge sampling delivers a higher performance in link prediction at low sampling rates than no sampling at all, so this may be a useful technique to improve performance even when sampling is not required.

There are several avenues of investigation to pursue from this point. Next steps will include determining the impact of sampling on performance as other parameters of the graph are changed, such as the size, the temporal growth pattern, or the generative model. In addition, the same sampling methods will be studied in the context of other anticipatory analytics, and the detection of emerging communities in particular [19] and spectral fingerprinting techniques such as those proposed in [20]. Ultimately, we intend to develop a framework for choosing sampling methods based on performance needs and computational constraints, within the context of a big data analytics architecture, such as [21].

ACKNOWLEDGMENT

The authors thank Dr. Jeremy Kepner and Dr. Albert Reuther for their assistance in developing the sampling and link prediction concepts. The authors also thank the LLGrid team at MIT Lincoln Laboratory for their support in performing the experiments.

REFERENCES

[1] L. Lu and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[2] B. Taskar, M. fai Wong, P. Abbeel, and D. Koller, "Link prediction in relational data," in *Advances in Neural Information Processing Systems 16*, ser. 659–666, S. Thrun, L. Saul, and B. Schölkopf, Eds. MIT Press, 2004.

[3] R. R. Sarukkai, "Link prediction and path analysis using markov chains," *Computer Networks*, vol. 33, pp. 377–386, 2000.

[4] D. Wang, D. Pedreschi, C. Song, F. D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi, "Human mobility, social ties, and link prediction," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2011, pp. 1100–1108.

[5] Z. Huang, X. Li, and H. Chen, "Link prediction approach to collaborative filtering," in *Proc. ACM/IEEE-CS Joint Conf. Digital Libraries*, 2005, pp. 141–142.

[6] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 631–636.

[7] D. R. Karger, "Random sampling in cut, flow, and network design problems," *Mathematics of Operations Research*, vol. 24, no. 2, pp. 383–413, 1999.

[8] R. Zou and L. B. Holder, "Frequent subgraph mining on a single large graph using sampling techniques," in *Proceedings of the eighth workshop on mining and learning with graphs*. ACM, 2010, pp. 171–178.

[9] M. Al Hasan and M. J. Zaki, "Output space sampling for graph patterns," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 730–741, 2009.

[10] C. Seshadhri, A. Pinar, and T. G. Kolda, "Triadic measures on graphs: The power of wedge sampling," in *Proceedings of the SIAM Conference on Data Mining (SDM)*. SIAM, 2013.

[11] V. Satuluri, S. Parthasarathy, and Y. Ruan, "Local graph sparsification for scalable clustering," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 2011, pp. 721–732.

[12] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *J. Amer. Soc. Inform. Sci. and Tech.*, vol. 58, no. 7, 2007.

[13] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Trans. on Knowledge Discovery from Data*, vol. 5, no. 2, 2011.

[14] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of Erdős-Rényi graphs," *Physical Review E*, vol. 85, no. 5, p. 056109, 2012.

[15] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proc. of the 4th SIAM Conference on Data Mining*, 2004, pp. 442–446.

[16] A. Reuther, J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, M. Hubbell, P. Michaleas, J. Mullen, A. Prout *et al.*, "Llsupercloud: Sharing hpc systems for diverse rapid prototypingsupercloud: Sharing hpc systems for diverse rapid prototyping," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*. IEEE, 2013, pp. 1–6.

[17] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 849–856.

[18] B. W. Bader, T. G. Kolda *et al.*, "Matlab tensor toolbox version 2.5," Available online, January 2012.

[19] B. A. Miller, N. Arcolano, and N. T. Bliss, "Efficient anomaly detection in dynamic, attributed graphs," in *Proc. IEEE Int. Conf. Intelligence and Security Informatics*, 2013, pp. 179–184.

[20] V. Gadepally and J. Kepner, "Big data dimensional analysis," in *Proc. IEEE High Performance Extreme Computing*, 2014.

[21] V. Gadepally, J. Bolewski, D. Hook, D. Hutchison, B. Miller, and J. Kepner, "Graphulo: Linear algebra graph kernels for NoSQL databases," in *Proc. IEEE Parallel and Distributed Process. Symp. Workshops*, 2015.