Distributed Gaussian Conditional Random Fields Based Regression for Large Evolving Graphs

Jelena Slivka[1], Mladen Nikolić[2], Kosta Ristovski[3], Vladan Radosavljević[3], Zoran Obradović[3]

Abstract

Dynamics of many real-world systems are naturally modeled by structured regression of representationally powerful Gaussian conditional random fields (GCRF) on evolving graphs. However, applications are limited to small and sparse evolving graphs due to high computational cost of the GCRF learning and inference. In this study, a new method is proposed to allow applying a GCRF model to large and extremely dense evolving graphs. Efficiency issues are addressed by graph partitioning and application of the GCRF model to each partition independently and in parallel. The hypothesis evaluated in this project is that the robustness of GCRF allows distributed learning of accurate regression models as long as most of the nodes after the graph partitioning are still linked to a subset of their heavily connected neighbors. To evaluate this hypothesis, GRCF-based distributed regression experiments were conducted on synthetic and two real-world evolving networks. The obtained results provide evidence that the proposed approach greatly speeds up time needed for modeling large graphs while obtaining accurate models. Finally, a simple graph characterization is proposed that, when satisfied allows application of distributed GCRF regression to extremely big graphs without construction of the entire graph.

Keywords: Evolving graphs, graph partitioning, Gaussian conditional random fields, citation networks.

1 Introduction

Evolving graphs are an indispensable tool for representing, understanding, and analyzing relationships in many diverse domains. For example, finding coevolution relationships of structure and function is an important task in structural genomics [1]. Evolving graphs are also studied in computer networks, where monitoring changes in dynamic topology can provide important insights about network configuration [2]. Similarly, analysis of the dynamics of large social networks can yield important findings in the domain of social science [3], [4].

Modeling data by evolving graphs in order to utilize a greater portion of available information leads to a huge increase in the problem complexity as well as in the amount of data to be analyzed [5], [6], [7]. Nonlinear data analysis techniques often do not scale to data of such dimensions. Also, another problematic aspect of analysis of large evolving graphs is that they can be heterogeneous with respect to the laws that govern the behavior of the phenomenon of interest. Thus, there is a need for efficient algorithms to identify more homogenous components in very large evolving graphs.

In this paper we address the problem of applying the Gaussian Conditional Random Fields model (*GCRF*) [9] to large evolving fully connected weighted graphs in which hidden link weights are derived based on node similarity, with the goal of predicting values of the nodes' attributes in the next time interval. The *GCRF* method is a computationally costly procedure that scales quadratically in the number of nodes in sparse graphs and cubically in dense graphs. We propose to address efficiency issues by graph partitioning and then applying the GCRF model to each partition in parallel.

Contributions of this paper are the following:

- Our approach greatly reduces the execution time needed for network modeling and significantly improves accuracy of the employed method. We confirm this on two real-world datasets (HEP-Th[4] [10] and US patents[5] [11]). For example, when applying GCRF to the graph of 4,598 nodes without graph partitioning, large computational costs require neglecting the majority of edges in the graph. Applying the GCRF to a greatly sparsified graph took more than 5 times longer than when using the partitioning approach applied to the full graph. In addition, an application to a sparse graph resulted in R^2 of 0.375, while our partitioning approach was much more accurate, achieving R^2 of 0.563.
- We provide evidence that the graph partitioning will not significantly hurt the prediction accuracy of GCRF as long as after partitioning most of the nodes are still connected to a small subset of their heavily connected neighbors. Two nodes will be called *heavily connected* if the link that connects them has a large weight as compared to link weights of other nodes. We find support for our hypothesis in both synthetic and real-world citation network data.
- We exploit the previous insight to allow distributed GCRF-based modeling of graphs that are too big to be constructed. This is achieved by random partitioning of big graphs prior to their construction followed by explicitly constructing only much smaller partitions.

[1] Faculty of Technical Sciences, University of Novi Sad, slivkaje@uns.ac.rs
[2] Faculty of Mathematics, University of Belgrade, nikolic@matf.bg.ac.rs
[3] Center for Data Anaytics and Biomedical Informatics, Temple University {tub09693, vladan, zoran.obradovic}@temple.edu

[4] https://kdl.cs.umass.edu/display/public/HEP-Th
[5] http://www.nber.org/patents/

The method we propose is based on the idea employed in a previous study (Peng *et al*. [8]) where graph partitioning is successfully used to improve prediction and runtime performance of predicting re-tweeting decisions of Twitter users by a Conditional Random Field (*CRF*) based classification method. In that study partitioning the network into sub-graphs not only improved the runtime, but also improved the prediction accuracy of the model by exploiting heterogeneity of users' decisions when re-tweeting. Our work is related to the work of Peng *et al*. [8]. However, significant differences include the following aspects:

- Peng *et al*. consider a sparse network, while we show that a partitioning can be beneficial for dense and even fully connected graphs.
- For partitioning, we use a local partitioning method, tailored to test our hypothesis, a purely random one, as well as METIS, a renowned global partitioning method in order to explore the robustness of GCRF.
- Peng *et al*. consider a classification problem of user behavior, while we consider a regression problem of node attribute prediction by deploying Gaussian Conditional Random Fields.
- Finally, Peng *et al*. explore only static graphs, while we model evolving graphs.

2 Methodology

The methodology for scalable training of GCRF we propose in this paper consists of the following steps:

1. Aggregate given evolving graph G to obtain a static graph G_s such that the adjacency matrix of G_s is the average of adjacency matrices of graphs G for all time steps.
2. Partition G_s by a graph partitioning method to obtain sub-graphs $g_1,...,g_l$ such that each node from G_s belongs to one and only one sub-graph g_i for i=1,...,l.
3. Create a set $V=\{V_1,...,V_l\}$ where V_i is a set of nodes of g_i for i=1,...,l.
4. Let G_t be a snapshot of G in time step t. Let g_t^i be a sub-graph of G_t induced by nodes V_i. The set of evolving partitions of G is $\{\{g_1^1,...,g_T^1\},...,\{g_1^l,...,g_T^l\}\}$.
5. Train a separate GCRF on each evolving partition of G in parallel.
6. For each node v from G, use the predictive model corresponding to the partition v belongs to.

The aggregation in the first step is performed for two reasons. Firstly, in the current implementation, the GCRF model assumes graphs to be stable in terms of nodes they are comprised of, i.e. all nodes need to be present in graph in all considered time steps. In other words, the current implementation is not aimed at modeling newly emerging nodes or nodes that are sometimes unobserved. Secondly, there is much larger variety of algorithms for partitioning static rather than evolving graphs, as well as algorithms for modeling static communities as opposed to algorithms for modeling evolving communities and they are more easily obtainable. Thus, in this article we focus on modeling stable communities and leave modeling of partially observed evolving communities as a task for future research.

The maximal size of the individual subnetworks should be chosen so that it allows the feasible application of considered models.

By modeling the extracted subnetworks independently, we are neglecting the influence that might exist among the nodes that belong to different partitions. It is clear that with such a partitioning introduced to speed up computation there is a risk of disregarding important connections between the nodes that were placed in different partitions. However, evidence will be provided in the results section that in practice this risk is not compromising prediction accuracy of GCRF-based regression by much.

The parallelization in our implementation was achieved by using MPI (*Message Passing Interface*). We have employed a master-slave model of communication. In our implementation, the master process is responsible for graph partitioning. The master sends one partition to each of the available slave processes that work in parallel. After receiving a partition, the slave process is considered busy until it finishes processing the assigned partition. Each slave process first trains the GCRF model on the partition it was assigned and then uses the obtained model to predict future values for the nodes that belong to that partition. The slave process then reports back to the master with the results, after which it is considered available again and the master process can assign it with another partition, if needed. The master process keeps assigning partitions to slaves until all partitions have been modeled.

In section 2.1 the GCRF model is described and the graph partitioning methods are described in section 2.2.

2.1 *GCRF model*

2.1.1 Continuous Conditional Random Fields. Conditional Random Fields (CRF) is a type of discriminative probabilistic graphical model designed to predict structured output. In was originally designed for classification of sequential data [12].

In regression problems, the output y_i is associated with input vectors $x=(x_1,\ldots,x_n)$, by a real-valued function called association potential $A(\alpha,y_i,x)$, where α is a K-dimensional set of parameters. Interactions among two outputs can be modeled through a real-valued function, referred to as interaction potential $I(\beta,y_i,y_j,x)$, where β is an L-dimensional set of parameters. In general case, I can also depend on an input x. For the defined

association and interaction potentials, CRF models a conditional distribution $P(y|x)$, y= ($y_{1,\dots}y_n$):

$$(2.1) \qquad P(y|x) = \frac{1}{Z(x,\alpha,\beta)} \exp\left(\begin{matrix}\sum_{i=1}^{N} A(\alpha, y_i, x) + \\ \sum_{j\sim i} I(\beta, y_i, y_j, x)\end{matrix}\right)$$

Where $j \sim i$ denotes that the outputs y_i and y_j are connected, and where $Z(x,\alpha,\beta)$ is a normalization function defined as

$$(2.2) \qquad Z(x,\alpha,\beta) = \int_y exp\left(\begin{matrix}\sum_{i=1}^{N} A(\alpha, y_i, x) + \\ \sum_{j\sim i} I(\beta, y_i, y_j, x)\end{matrix}\right) dy$$

Association and Interaction potential are usually defined as linear combinations of a set of feature functions in terms of α and β [13].

$$(2.3) \qquad A(\alpha, y_i, x) = \sum_{k=1}^{K} \alpha_k f_k(y_i, x),$$

$$(2.4) \qquad I(\beta, y_i, y_{j,} x) = \sum_{l=1}^{L} \beta_l g_l(y_i, y_{j,} x).$$

In this way, any potentially relevant feature can be included in the model and parameter estimation automatically determines actual relevance by feature weighting. The learning task is to choose values of parameters α and β to maximize the conditional log-likelihood of the set of training examples (we assume that interactions among outputs are defined over the whole training set):

$$(2.5) \qquad L(\alpha,\beta) = logP(y|x)$$

$$(2.6) \qquad (\hat{\alpha}, \hat{\beta}) = \text{argmax}_{\alpha,\beta}(L(\alpha,\beta))$$

The inference task is to find the outputs y for a given set of observations x and estimated parameters α and β, such that the conditional probability $P(y|x)$ is maximized

$$(2.7) \qquad \hat{y} = \text{argmax}_y(P(y|x))$$

CRFs were originally designed for classification problems where the normalizing function Z is a sum over a finite set of possibilities. For regression, Z must be an integrable function, which can be very difficult and computationally expensive due to the complexity of interaction and association potentials. This issue can be addressed by representing $P(y|x)$ as a multivariate Gaussian distribution, resulting in Gaussian Conditional Random Fields (*GCRF*) method [9].

2.1.2 Gaussian Conditional Random Fields. Functions f and g can be defined as quadratic functions in terms of y, for example:

$$(2.8) \qquad f_k(y_i, x) = -(y_i - R_k(x))^2 ,$$

where $R_k(x)$ is the *unstructured (or baseline) predictor* that predicts y_i relying on any subset of *x,* but disregarding structure among predicted variables and

$$(2.9) \qquad g_l(y_i, y_j, x) = -e_{ij}^{(l)} S_{ij}^{(l)}(x)(y_i - y_j)^2,$$

where $S_{ij}^{(l)}$ represents the similarity between outputs y_i and y_j, and $e_{ij}^{(l)} = 1$ if an edge exists between y_i and y_j under the particular interaction potential l, and $e_{ij}^{(l)} = 0$ otherwise. In this case it can be shown that P(y|x) can be written as

$$(2.10) \; P(y|x) = \frac{1}{(2\pi)^{\frac{N}{2}} |\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}(y-\mu)^T \Sigma^{-1}(y-\mu))$$

for some covariance matrix Σ and mean vector μ that can be effectively determined.Note that due to the matrix inversion (Σ^{-1}), the complexity of GCRF method is $O(N^3)$, where N stands for number of nodes. However, if Σ^{-1} is sparse, the training time decreased from $O(N^3)$ to $O(N^2)$.

In the case of evolving graph of N nodes spanning over T time steps, covariance matrix Σ^{-1} is obtained by arranging covariance matrices for each time step along the diagonal of $N{\cdot}T \times N{\cdot}T$ matrix.

For the purposes of experiments in this paper, we have developed an implementation of GCRF in C++ code, by using PETSc (Portable, Extensible Toolkit for Scientific Computation) package.

2.2 *Graph partitioning.*

We hypothesize that for successful training of GCRF it suffices that each node is connected to only a small subset of its heavily connected neighbors and that we will not significantly hurt the predictive performance of GCRF by disregarding the rest of the node's connections (both week and strong ones). Given that the graph has large enough underlying clusters (by cluster we informally mean a subset of tightly connected nodes, i.e. the group of nodes that highly influence each other) and that we choose large enough partition size, chances are that, regardless of the partitioning method, most of the nodes will be in the same partition as some of their heavily connected neighbors. In such cases, GCRF performance should be fairly robust to the choice of partitioning method, maybe even if random partitioning is performed. However, if these requirements are not met, we expect bad performance of random partitioning, as it may disregard important influences between heavily connected nodes by accidentally placing them into separate partitions. In this case, we expect that using more intelligent partitioning techniques would yield better performance of GCRF method.

In order to evaluate our hypothesis, we will employ several partitioning methods. The simplest partitioning approach we considered is random partitioning, which is used to test the robustness of GCRF training in the extreme case of partitioning which is blind to graph structure. Then, we design a simple Greedy Graph Partitioning (GGP) algorithm which relies only on the principle of keeping the most tightly connected nodes in

the same partition. Its purpose is not to outperform existing partitioning algorithms in any way, but only to clearly (thanks to its simplicity) demonstrate that the main principle expressed by our hypothesis suffices to provide good results in GCRF training. We expect it to outperform random partitioning. We also used Metis [14], an intelligent and renowned global partitioning method. Metis tends to group most heavily connected nodes into the same partition, which should, according to our hypothesis, yield good results in GCRF training. However, Metis is not a simple algorithm and distinguishing which of its properties contributes the most to the results is nontrivial. Therefore, we use it only as an additional tool in evaluating our hypothesis. If our hypothesis is true, Metis should not significantly outperform GGP, since GGP already implements the principle expressed by the hypothesis.

2.2.1 Random graph partitioning. By random partitioning of a graph we mean a particularly simple approach of assigning nodes randomly to partitions, regardless of the way the nodes are connected.

2.2.2 Greedy graph partitioning. Greedy graph partitioning (GGP) algorithm is a local partitioning algorithm we have developed aiming specifically to test our hypothesis that for successful GCRF training, such that each node only needs to be connected to a subset of its most influential neighbors. Each partition is created by selecting a random seed node and expanding from it by adding one node at a time until size limit is reached or a whole connected component of the graph is extracted. At each step, we add the node which is most heavily connected to the current partition and is not already assigned to some partition. The pseudo code of the algorithm is given in Figure 1.

```
graphPartitioning(Graph G, int size) {
    partitions = {}
    n = pickFreshNode(G)
    while(n <> null) {
         P = findLocalPartition(n,size)
         partitions = partitions ∪ {P}
         n = pickFreshNode(G)
    }
    return partitions;
}
findLocalPartition(Node n, int size) {
    part = {n}
    n.fresh = false
    while(expandable(part) & |part|<size) {
         m = bestFreshNeighbor(partition)
         m.fresh = false
         part = part ∪ {m}
    }
    return part
}
```

Figure 1: Pseudo-code of Greedy graph partitioning algorithm

Node property `fresh` indicates if a node is assigned to partition. If this property is true for a node, we call it a fresh node and consider unassigned. We assume that function `pickFreshNode(G)` returns the first fresh node or `null` if such node does not exist. Function `expandable(P)` returns false if all the fresh neighbors of nodes in `P` are also in `P` and returns true otherwise. Function `bestFreshNeighbor(P)` returns a fresh node from graph which is not in `P`, but is connected to some node in `P` by an edge of the greatest weight among such nodes.

3 Experimental evaluation

In this section, we report the results of experimental evaluation of our approach and our hypothesis. First, we use a synthetic dataset to check the hypothesis in an idealized setting. Then, we use a real-world citation network from the field of high-energy physics to show the usefulness of our approach. Lastly, we use random partitioning to facilitate processing of the large dataset of US patents citations, the graph of which is too big to be easily constructed in a reasonable time on a computer used in our experiments.

All experiments are conducted on a Linux-based machine with 20 nodes, where each node has 12 Intel Xeon 2.8GHz cores and 12Gb memory. Note that on this platform the maximal number of communities that can be processed in parallel is 239 (20x12 – 1 master process).

As the measure of the accuracy of the employed model, we use the predicted R^2 measure:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_{true}^i - y_{pred}^i)^2}{\sum_{i=1}^{n}(y_{true}^i - \overline{y_{true}})^2} \quad (2.11)$$

where n is the number of nodes in the graph, y_{true}^i is the true value for the node i, y_{pred}^i is the predicted value for the node i, and $\overline{y_{true}}$ is the mean of the true values for all nodes.

3.1 *Synthetic dataset.*

As a first step, we perform an experiment on synthetic data, which enables us to control the parameters of the network we consider important, namely the size of node clusters in the network. Also, we design it to suit the GCRF model, so that, at this first experiment, we need not worry about peculiarities of real-world data, but focus on the phenomenon of interest in its simplest form.

3.1.1. Synthetic data generation. The experiments rely on data sampled from GCRF treated as a generative model. The data are constructed to have a naturally clustered structure with the cluster size used as a parameter. In order to produce such data from a generative model, we specified the graph size, number of time steps, parameters α and β, values of the unstructured predictor, and a similarity/weight matrix. In all experiments we have chosen the graph size to be 1000 nodes, and number of time steps to be 16. The objective was for GCRF to display large improvement

over the baseline predictor, and thus we have empirically selected the parameters α and β so that the unstructured predictor achieves R^2 around 0.3 and GCRF applied to the full graph improves R^2 to around 0.6. For a node, the prediction values r_t of the unstructured predictor were randomly generated. The matrix of similarities/weights between the nodes was constructed as a block diagonal matrix where each block represents a node cluster, which is more heavily connected within itself than to the other clusters. The values within the blocks are randomly generated from the interval [0,1], while the values outside of the blocks were randomly generated from the interval [0,1/M], where M is the number of blocks in the matrix. The strength of the links between clusters was by construction dependent on the cluster size, because in networks with smaller clusters there are more links between clusters than in networks with larger clusters, and the summed influence of those weights could overpower the influence within the cluster. The similarity matrix is kept constant for all time steps. Finally, we sample target values y_t according to the GCRF generative model.

3.1.2 Experiments on generated data. We observe the behavior of three partitioning approaches. We expect to see significant deterioration of predictive performance with the decrease of cluster size in the network when random partitioning is employed. Greedy graph partitioning should exhibit much more robustness if our hypothesis were true. We also expect METIS to behave well, being an intelligently designed and renowned partitioning algorithm.

We sample 10 evolving graphs for each cluster/partition size: 5, 10, 25, 50, 100, 200 such that each graph contains equal-sized clusters. For each graph, we perform training on 15 steps and evaluate the model on the last step. Maximal partition size used for each partitioning approach corresponds to the cluster size in the generated graph. We have tested both accuracy (in terms of achieved R^2 measure) and execution time for GCRF applied on the whole graph, which we will refer to as *No partitioning*, as well as for our partitioning approaches, which we will refer to as Random, and GGP, depending on the method used to extract partitions. We refer to the performance of the unstructured predictor as *Baseline*. The reported execution time and R^2 for each setting were averaged over five different runs of the same experiment. The R^2 measures for these settings are shown in Figure 2, and their execution time is shown in Figure 3.

As we can observe from Figure 2, on the networks generated to have large cluster size (≥100 nodes), all partitioning settings, including Random, achieve approximately the same R^2 as GCRF applied on the whole graph without partitioning.

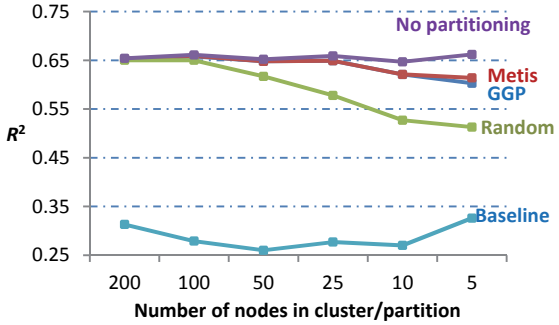

Figure 2: Synthetic data: R^2 accuracy of regression for graph partitioned to clusters of 5 to 200 nodes is shown. The GCFR regression is applied to the entire graph (No partitioning) and distributed GCFR regression is applied to partitions obtained by 3 methods (Metis, GGP and Random). Baseline results are obtained by the unstructured regression model applied to the corresponding partitions.

However, as the cluster size decreases, the number of links between the nodes from different clusters increases, and although the weights of those links are very small, their combined influence becomes stronger. On such graphs GCRF applied to partitions resulting from Metis and GDP partitioning achieves smaller R^2 than *No partitioning* setting; however this loss in accuracy is small. However, there is a significant drop of accuracy for Random setting as the cluster size of the generated graphs decreases. We conclude that the more intelligent partitioning techniques like Metis and GGP were able to identify tightly connected clusters and thus did not hurt the performance of GCRF.

On the other hand, there is a significant drop of accuracy for Random setting as the cluster size of the generated graphs decreases, since in a small randomly chosen partition it is more likely for a node to be isolated from all the nodes from its cluster. These results support our hypothesis – if the underlying clusters are large enough and we have used large enough partitions, regardless of the partitioning method used, GCRF performance is not hurt. As random partitioning is the least complex method, it is best suited for huge networks with large underlying clusters. In case that the clusters in the network are very small, more intelligent partitioning techniques than random partitioning should be used.

It should be noted that even with random partitioning, GCRF still succeeded in significantly improving the performance of the baseline predictor. Thus, even in case the clusters in the network are very small, random partitioning can still be useful if other approaches are not applicable for some reason.

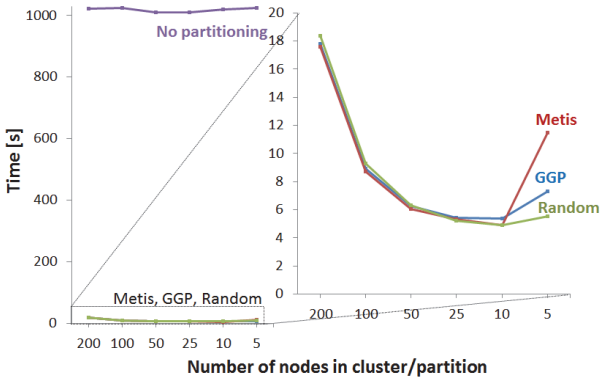

Figure 3: Synthetic data: execution time of distributed GCFF regression (Metis, GGP and Random graph partitioning to clusters of 5 to 200 nodes) versus time for centralized GCRF regression (No partitioning)

As we can observe from Figure 3, the execution times for all partitioning settings are comparable due to the small size of the graph. We can see the significant drop in execution time as the partition size drops. However, when the partition size drops to 5, we can actually see that the computational time increases. This is due to the fact that GCRF training time is already very low for partitions of size 10 and for lower partition sizes other factors such as communication and synchronization overhead take over due to the increased number of processes.

Execution time for *No partitioning* setting was 1023 seconds, which is more than 50 times longer than execution time for any of the partitioning-based alternatives. Thus, by partitioning the graph we are able to greatly reduce the computational requirements of GCRF without significantly compromising accuracy of GCRF-based regression.

3.2 *High-energy physics citation dataset.*

High-energy physics theory (HEP-Th) is a bibliographic network which was extracted from arXiv for the 2003 KDD Cup competition [10]. The network consists of 29,955 nodes (papers) and 352,807 links (citations) spanning over 11 years.

We have preprocessed the datasets with the goal of selecting the fixed group of nodes (i.e. the group of nodes whose presence in the network does not change over the observed time period) as in [15]. We have kept all the papers published before January 2000. We have observed the citation numbers for these papers received after January 2000 on a monthly basis, thus obtaining data for 39 time points. We have eliminated all papers that have less than 3 citations over the whole observed time period. The intuition behind this is that papers which have very low citation in a prolonged period of time are very unlikely to be cited in the future. Therefore, it would be better to reduce the problem dimensionality for a computationally expensive method by removing such nodes from the prediction and predicting their citation count by some cheaper method (e.g., unstructured predictor of GCRF or simply 0). After preprocessing, we were left with 4,598 papers.

In the experiments performed in this study, our goal was to predict the number of citations each scientific paper will receive in the next unobserved time step. To do this, we have constructed the evolving graph in the following way. Scientific papers were treated as graph nodes, and number of citations the scientific paper received in time step t was used as the attribute of the node representing that paper in time step t. The links among the nodes (in this case scientific papers) and their weights are determined based on the similarity of sequences of citation numbers of two papers, which we refer to as citation histories. We define the distance between nodes a and b in time step t as

$$d_{a,b,t}{}^2 = \sum_{i=1}^{n}(a_{t-i} - b_{t-i})^2, \tag{2.11}$$

where a_l and b_l denote the number of citations of papers a and b in time step l, respectively. We want the weights to approximate the actual similarity of the nodes a and b in the time t, which we define to be by $e^{-(a_t-b_t)^2}$. We cannot calculate this similarity directly because we do not know the citation numbers a_t and b_t for time step t that we want to predict. Therefore, we estimate it by $e^{-k\cdot d_{a,b,t}{}^2}$ for a suitable k and use it as a weight between a and b. We learn the coefficient k by minimizing

$$\sum_{\substack{a,b\in S,\\ t\le T}}(e^{-kd_{a,b,t}{}^2} - e^{-(a_t-b_t)^2})^2 \tag{2.12}$$

with respect to k, where S is the random sample of graph nodes and T is the last time step used for training. It should also be noted that calculating this similarity measure has time complexity of $O(T\cdot N^2)$, because we have to compare every pair of nodes for each time step we are considering. This poses no problem for graphs of a few thousand nodes, but may be the problem for huge graphs. In our experiments we have used the citation history of n=19 time steps.

The complexity of GCRF depends on the number of nodes. So, in our experiments, we try different sizes for the partitions which we model. Also, the complexity of *GCRF* is dependent on the graph density[6]. By following the described graph constructing procedure we create a fully connected graph.

As the unstructured predictor for GCRF, we used the citation count from the previous time step. For training, we used the first 29 time steps, and for testing we used the remaining 10 steps. We compare the performance of the same partitioning settings used in 3.1.

First, we present our partitioning approach applied to the fully connected graph. Application of *No*

[6] The complexity of the *GCRF* model on a fully connected graph is $O(N^3)$, but it can be reduced to $O(N^2)$ if the graph is sparse.

partitioning setting on this graph is infeasible due to the memory limitations. The R^2 measure for these settings is shown in Figure 4, and their execution time is shown in Figure 5.

As we can observe on Figure 4, Metis and GGP have very similar performance. Random has somewhat worse performance than other settings, but this difference is small. However, it should be noted that all partitioning settings, including Random, achieve a significant improvement over the unstructured predictor. It should also be noted that the partition size did not significantly influence the prediction accuracy.

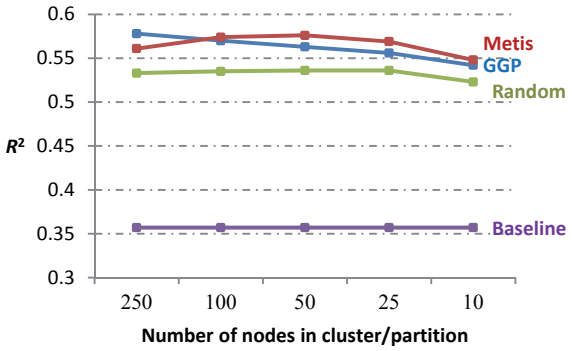

Figure 4: HEP-Th: R^2 accuracy of regression for graph partitioned to clusters of 10 to 250 nodes is shown. The GCFR regression is applied to the entire graph (No partitioning) and distributed GCFR regression is applied to partitions obtained by 3 methods (Metis, GGP and Random). Baseline results are obtained by the unstructured regression model applied to the corresponding partitions.

We can see the significant drop in execution time as the partition size drops (Figure 5). As in synthetic data, we experience the increase in execution time when the partition size drops to 10. This is due to the limit of 240 cores, such that we cannot process all 459 communities in parallel. Again, graph partitioning time is negligible compared to the overall execution time.

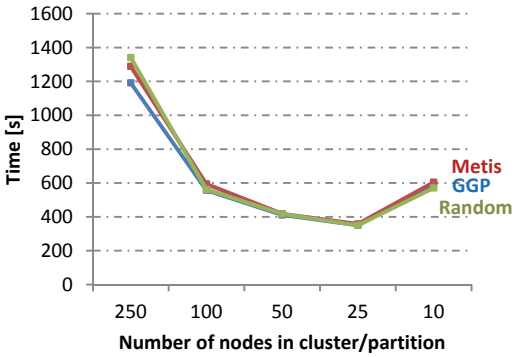

Figure 5: HEP-Th: Execution time for distributed GCFR regression for clusters of 10 to 250 nodes by Metis, GGP and Random graph partitioning.

Due to the large number of nodes and time steps, direct application of GCRF to the whole network of this fully connected graph is computationally infeasible. In order to apply GCRF to the network, we must cut some edges to make the graph sufficiently sparse. We have successively removed half of the edges of the lowest weight until GCRF application on all nodes was feasible. In this way, we ended up with a graph that had $5 \cdot 10^{-4}$ % of edges of the full graph. Applying GCRF to this sparse graph took 2,123 seconds, which is more than for any of the partitioning settings applied to the fully connected graph. Furthermore, the R^2 measure achieved by GCRF in this way was 0.357. This means that, in order to apply GCRF to all nodes, we were forced to neglect so much information that GCRF was unable to improve upon the unstructured predictor. We can conclude that, in the case that it is infeasible to apply GCRF to the whole graph, partitioning can not only reduce the execution time, but also improve the prediction accuracy by enabling us to include more information in the model.

In order to get a better feeling of how the partitioning affects the prediction accuracy and execution time in the case where it is feasible to apply GCRF without discarding edges, we have applied all settings on a smaller subset of 400 top cited nodes of HEP-Th dataset. The R^2 measure for different settings is shown in Figure 6, and the execution time is shown in Figure 7.

As we can observe from Figure 6, Metis and GGP outperformed *No partitioning* setting in terms of accuracy, for all partition sizes, except for partition size 10, where they have approximately the same accuracy. As expected, Random achieved lower accuracy than GGP and Metis. However, for partition sizes of 50 and more, its accuracy is roughly equal to accuracy of *No partitioning*.

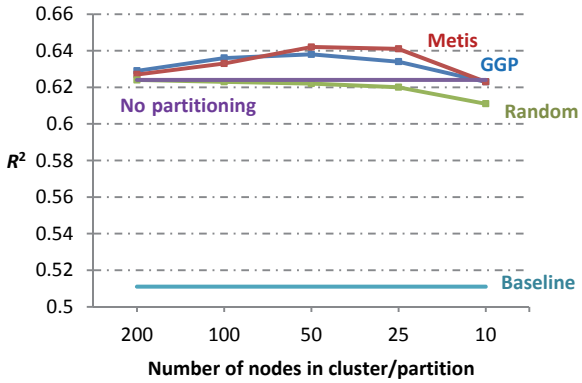

Figure 6: Top 400 HEP-Th: R^2 accuracy of regression for graph partitioned to clusters of 10 to 200 nodes is shown. The GCFR regression is applied to the entire graph (No partitioning) and distributed GCFR regression is applied to partitions obtained by 3 methods (Metis, GGP and Random). Baseline results are obtained by the unstructured regression model applied to the corresponding partitions.

In accordance with our hypothesis, accuracy of Random drops as the size of the partition drops. Finally, it should be noted that all approaches provide significant improvement in accuracy over the unstructured predictor.

Partitioning the network also results in significant decrease in execution time (Figure 7). For example, our partitioning approach for partition size of 50 nodes, where both Metis and GGP achieve better performance than *No partitioning*, is more than 100 times faster than GCRF applied to the whole graph. From Figure 7, we can also observe that the execution time drops with the decrease of partition size to around 50 nodes. Choosing partition size of less than 50 nodes did not bring much improvement in terms of reducing the execution time, as the increased number of processes increases execution time due to factors such as communication and synchronization overhead.

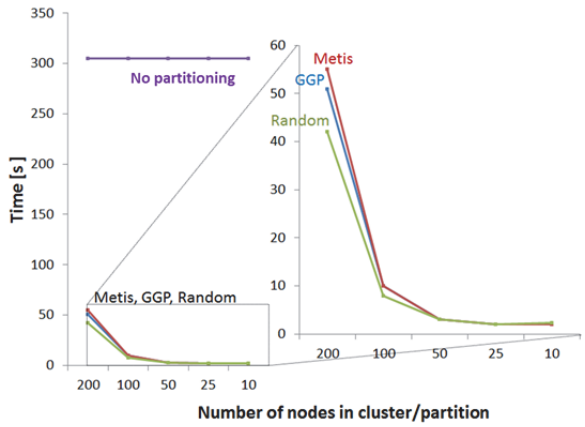

Figure 7: Top 400 HEP-Th: Execution time for distributed GCFR regression for clusters of 10 to 200 nodes by Metis, GGP and Random graph partitioning is compared to centralized GCRF regression (No partitioning).

We may conclude that partitioning can not only significantly reduce the execution time of GCRF, but also help improve prediction accuracy. This is in accordance with results presented in [8].

Interestingly, even a light-weight technique such as Random was able to significantly improve the prediction accuracy, and, for the large enough partition size, its accuracy was equal to GCRF applied to the whole graph.

3.3 *US patents citations dataset*

US patents citations dataset[7] [11] comprises detailed information on almost 3 million U.S. patents granted between January 1963 and December 1999 and over 16 million citations made to these patents between 1975 and 1999.

We have preprocessed the US patent dataset in the same way as the HEP-Th dataset. We have observed the citation numbers that patents published before January 1980 received after January 1980 on a 6-month basis, thus obtaining data for 40 time points. We have eliminated papers that have less than 5 citations over the observed time period. After preprocessing, we were left with 748,090 papers. Similarly to our experiments on HEP-Th dataset, our goal was to predict the number of citations each patent will receive in the next unobserved time step. The patents represent the graph nodes, and number of citations the patents received in time step t was used as the attribute of the node representing that patent in time step t. We define the links among the nodes (patents) and their weights based on their citation histories of n=19 time steps.

As we have mentioned earlier in section 3.1, creating the graph by calculating link weights has complexity $O(T \cdot N^2)$, because we have to compare each pair of nodes in each time step. In the case of the US patents dataset, constructing such a large graph would be very time consuming and memory demanding.

Metis and GGP would require all link weights to be known in advance in order to start partitioning, which may result in very long graph construction time. Moreover, most of the calculated similarities will be discarded by partitioning and will not influence the final performance of the model. Finally, in order to apply partitioning, we would need to load the whole graph in the memory, which may be infeasible for very large graphs. However, Random can be applied without prior graph construction, and still give us a reasonable accuracy, given that the assumptions about sufficiently large underlying clusters and partitions are true. Thus, on US patents dataset, we randomly partition the graph by assigning the nodes randomly to different partitions and, since the links between the nodes that belong to different partitions are discarded, only construct the small sub-graphs defined by random partitions.

As the unstructured predictor for GCRF we used the citation count from the previous time step. For training, we used the first 35 time steps, and for testing we used the remaining 5 steps. The R^2 measure for all tested setting is shown in Figure 8, and their execution time is shown in Figure 9.

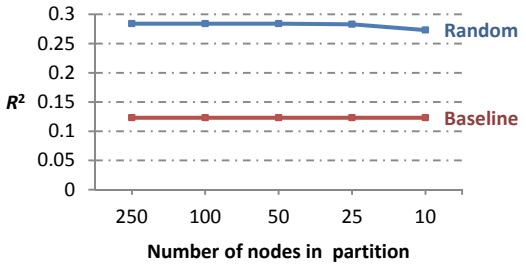

Figure 8: R^2 accuracy of distributed vs centralized GCFR regression on US patents dataset for partitions to clusters of 10 to 250 nodes.

[7] http://www.nber.org/patents/

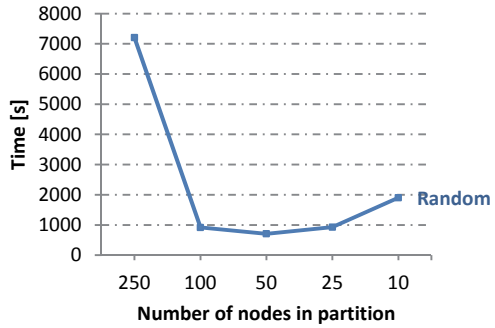

Figure 9: Execution time of distributed GCRF regression for partitions to clusters of 10 to 250 nodes on US patents dataset

As we can observe from Figure 8, Random partitioning achieved a significant improvement over the Baseline. The performance of Random was robust to the choice of the partition. Applying GCRF to the full graph of 748,090 nodes was infeasible due to both memory and execution time limitations. However, by using Random setting with partition size of 50 nodes, we were able to give predictions in 927 seconds. We may conclude that, with the help of random partitioning, we were able to achieve a significant improvement over the performance of the unstructured predictor, without the need of calculating and loading into memory all weights of the large time evolving graph.

4 Conclusion

In this paper, we explored a way to apply GCRF, a structured regression method, to large fully connected graphs. The proposed approach is based on partitioning the graph to smaller sub-graphs and training independent GCRFs on separate partitions. We evaluated our approach on the problem of citation prediction in the domain of high-energy physics. We achieved huge speedup and significant accuracy improvement compared to training on all nodes at once, which can be done only on a sparse graph, leading to loss of information. Moreover, we confirmed the hypothesis of robustness of GCRF with respect to the partitioning technique employed. It turns out that a particularly simple approach – random partitioning – behaves like other partitioning methods if the clusters of nodes in the graph are sufficiently large. This enables us to train GCRF even on graphs which are too big to be constructed in whole. We do that by constructing only the partitions of the graphs (and not the whole graph), which are obtained by randomly assigning the nodes to those partitions. We demonstrate good performance of our approach on the problem of citation prediction on the large US patents dataset.

In the future, we intend to apply this partitioning approach with other powerful, but computationally complex, models for graph analysis. With that in mind, we are currently developing an easily extendible library based on this approach which would allow users to easily apply various prediction models to very large graphs. Also, we intend to work on data other than citations and on prediction of different kinds of attributes.

Acknowledgments

This research was supported by DARPA Grant FA9550-12-1-0406 negotiated by AFOSR, National Science Foundation through major research instrumentation grant number CNS-09-58854. It was partially supported by Serbian Ministry of Science grant 174021 and Serbian Ministry of Science grant “Infrastructure for Technology Enhanced Learning in Serbia" (III47003).

References

[1] Shakhnovich, B. E., Harvey, J. M.: Quantifying structurefunction uncertainty: A graph theoretical exploration into the origins and limitations of protein annotation. Journal of Molecular Biology, 4(337), 2004.

[2] Bunke, H., Dickinson, P. J., Kraetzl, M., Wallis, W. D.: A Graph-Theoretic Approach to Enterprise Network Dynamics. Birkhauser, 2006.

[3] Backstrom, L., Huttenlocher, D., Kleinberg, J., Lan, X.: Group formation in large social networks: membership, growth, and evolutio. In: KDD, 2006.

[4] Cha, M., Mislove, A., Gummadi, K.P.: A Measurement-driven Analysis of Information Propagation in the Flickr Social Network. In: WWW, 2009.

[5] Leskovec J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: KDD, 2005.

[6] Chakrabarti, D., Kumar, R., Tompkins, A.: Evolutionary clustering. In: KDD, 2006.

[7] Habiba, H., Yu, Y., Berger-Wolf, T. Saia, J.: Finding spread blockers in dynamic networks. In: ASONAM, 2008.

[8] Peng, H.-K., Zhu, J., Piao, D., Yan, R., Zhang, Y.: Retweet Modeling Using Conditional Random Fields. ICDM Workshops, pp. 336-343, 2011.

[9] Radosavljevic, V., Obradovic, Z., Vucetic, S.: Continuous Conditional Random Fields for Regression in Remote Sensing. In: Proc. 19th ECAI, August, Lisbon, Portugal, 2010.

[10] Gehrke, J., Ginsparg, P., Kleinberg, J. M.: Overview of the 2003 KDD Cup. SIGKDD Explorations 5(2): 149-151, 2003.

[11] Hall, B. H., Jaffe, A. B., Trajtenberg, M.: The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools. NBER Working Paper 8498, 2001.

[12] Lafferty, J., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. of ICML, pp. 282–289, 2001.

[13] McCallum, A., Lafferty, J., Pereira, F.: Conditional random fields:Probabilistic models for segmenting and labeling sequence data. In: Proceedings of ICML, pp. 282–289, 2001.

[14] Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. In: SIAM J. Sci. Comput., 20:359–392, December 1998.

[15] Uversky, A., Ramljak, D., Radosavljević, V., Ristovski, K., Obradović, Z.: Which Links Should I Use? A Variogram-Based Selection of Relationship Measures for Prediction of Node Attributes in Temporal Multigraphs. In: Proc. ASONAM 2013, Aug. 2013.