# A Fast Structured Regression for Large Networks

Fang Zhou, Mohamed Ghalwash, Zoran Obradovic
*Center for Data Analytics and Biomedical Informatics*
*Temple University, USA*
*Email: {fang.zhou, mohamed, zoran}@temple.edu*

***Abstract*—Structured regression has been successfully used in many applications where explanatory and response variables are inter-correlated, such as in weighted attributed networks. One of structured models, Gaussian Conditional Random Fields (GCRF), utilizing multiple unstructured models to learn the non-linear relationships between node attributes and the structured response variable, achieves high prediction accuracy. However, it does not scale well with large networks. We propose a novel model, called Scalable Approximate GCRF (SA-GCRF), which integrates weighted attributed network compression with GCRF, with the aim of making GCRF applicable to large networks. The model consists of three steps: first, it compresses a network into a smaller one by generalizing nodes into supernodes and edges into superedges; then, it applies GCRF to the reduced network; and finally, it unfolds the predicted response variables into the original nodes. Our hypothesis is that the reduced network maintains most information of the original network such that the loss in prediction accuracy obtained by GCRF on the reduced network is minor. The comprehensive experimental results indicate that SA-GCRF was 150-520 times faster than standard GCRF and 11-29 times faster than state-of-the-art UmGCRF on large networks, and provided regression results where GCRF and UmGCRF were not applicable. Furthermore, SA-GCRF achieved a *similar* regression accuracy, 0.76, to the one obtained from the original real-world weighted attributed citation network, even after compressing the network to 10% of its size.***

## I. INTRODUCTION

Predictive modeling on data that exhibits sequential, spatial, and/or temporal dependencies is a challenging task. Traditional predictive models assume independent and identically distributed random variables (iid) and thus often fail to provide high accuracy in real-world applications that naturally have structured dependence. Structured models like Conditional Random Fields (CRFs) [8], avoid iid assumption by simultaneously learning to predict all outputs given all inputs. These methods represent network structure of variables of interest ($\mathbf{y}$) and attribute values of the nodes ($\mathbf{x}$) in the form of graphical models in order to exploit correlations among output variables, which often results in accuracy improvements over traditional unstructured approaches [14].

F. Zhou and M. Ghalwash contributed equally to this work. Z. Obradovic is the corresponding author. M. Ghalwash is also with Ain Shams University, Egypt.

Table I
TIME AND ACCURACY ON *CAIDA* NETWORK [9] OF 26K NODES

| | GCRF [15] | UmGCRF [6] | Our model (SA-GCRF) |
|---|---|---|---|
| Accuracy | 0.99 | 0.99 | 0.90 |
| Time (mins) | 715 | 40.5 | 2.15 |

Recently, Gaussian CRF (GCRF), which is a discriminative model that defines feature functions in a Gaussian canonical form, is proposed ( [15], [18]). GCRF allows the utilization of unstructured predictors as feature functions, and modeling of non-linear relationships between inputs and outputs. These properties led to numerous extensions of GCRF ( [4], [13]). However, GCRF does not scale well on networks with tens of thousands of nodes. For example, GCRF takes 12 hours on *CAIDA* network [9] of 26K nodes (Table I). An approach [24] for sparse GCRF optimization exploits the network sparsity and provides specialized optimization techniques. However, the method is very computationally expensive on moderate networks (few hundreds of nodes). One approach for addressing this problem is the development of approximate methods for GCRF [16]. The method assumes fully connected networks in Euclidean feature space, which does not hold in many real-world applications. A very recent model [6], called Unimodal GCRF (UmGCRF), is 20 times faster than GCRF. However, the model still can not handle networks with more than $100K$ nodes (Table II).

We address the problem of applying GCRF to large attributed weighted networks. The idea is to compress a large network into a smaller one, and then map the predictions obtained from the reduced network to the original network, with the goal that the prediction accuracy on the reduced network is similar to the original one. There are two ways to decide the size of the reduced network. One is based on the user-specified size. A user could specify the size of the reduced network where GCRF is applicable according to his/her accepted running time. For large networks, regression accuracy is usually similar for different sizes of reduced networks, but running time varies a lot (Table II). The other approach is based on the user-specified error. A user could specify how much accepted error is in the prediction, and the network is compressed to the size where no more size could be reduced. However, the local compression may have global influence in the structured regression, thus the size of the reduced network is not easy to calculate. Therefore, in this work, we will focus on the first approach, where users could specify the resolution of the reduced network. The step of generalizing the original network into a compact representation is known as network summarization, which is different from graph partitioning/clustering [11] in that: the former one groups nodes that have similar link structure, whereas the latter one groups nodes that are more strongly connected to each other.

Network summarization (or graph compression, hereafter we use the term *network* and *graph* interchangeably) has been studied recently ( [3], [12], [20]) due to its successful applications [27]. The goal is to generalize a large network into a smaller one while maintaining approximately the same information. For example, a Minimum Description Length based algorithm is proposed to produce an unweighted graph summary, with a set of edge corrections to fix the errors introduced by merging nodes and edges [12]. An attributed graph with categorical node attributes and continuous edge attributes is considered in [19], where the goal was to find relatively homogeneous supernodes and superedges. This approach has been generalized to numerical node attributes [26], but the node attributes are categorized based on domain knowledge. Furthermore, a method that considers both edge weights and long-range indirect connections between nodes was developed in [20].

We propose a novel model, called Scalable Approximate GCRF (SA-GCRF), which integrates graph compression with GCRF. Our model first generates a network into a user-specified size, where GCRF could finish regression in an acceptable time. Then, it applies GCRF to perform prediction on the reduced network based on the generalized node attributes and edge weights. Finally, it maps the prediction from the generalized nodes to the original ones. The first step in the SA-GCRF model is weighted attributed graph compression, which is inspired by the model of Toivonen et al. [20], but we extended the model to consider not only edge weights, but also numerical node attributes during compression, which significantly improves the prediction accuracy of SA-GCRF. Our work differs from the lifted inference [7] in that the latter focuses on extracting compact representations from multi-rational domains.

The main conclusions of this paper are the following: (1) In SA-GCRF, the weighted attributed network compression approach extends the state-of-the-art model to consider both node attributes and link structure, which significantly improves the prediction accuracy. (2) The proposed SA-GCRF model greatly reduces the running time of GCRF on large networks, while the loss in prediction accuracy is minor (e.g., on a network of $26K$ nodes, SA-GCRF is **332 times faster** than standard GCRF, and **19 times faster** than state-of-the-art UmGCRF, whereas the loss in $R^2$ is less than 0.1). (3) SA-GCRF provides solutions in an acceptable time where GCRF and UmGCRF are not applicable (Table II). (4)

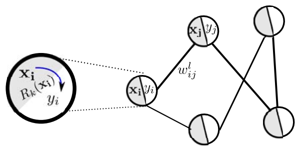


Figure 1. Unstructured predictor $R_k$ considers only $\mathbf{x}_i$ to regress $y_i$, while GCRF models the conditional distribution of all $\mathbf{y}$ given all $\mathbf{x}$ as a function of the structure $w^l$ and the unstructured predictor $R_k$.

Extensive experimental results on synthetic and real-world networks provide evidence that SA-GCRF allows *efficient* and *effective* learning and inference for structured regression on large networks.

## II. Methodology

The structured (network) regression problem can be defined as modeling a function that predicts the response variables from the nodes' explanatory variables, where there is a structure among the response variables [14]. This problem was formulated as Gaussian Conditional Random Fields (GCRF) [15]. In Section II-A, we introduce the main building blocks of GCRF and then briefly introduce UmGCRF [6], which is much faster than GCRF. The graph compression is then discussed in Section II-B and the proposed algorithm SA-GCRF is explained in Section II-C.

### A. GCRF and UmGCRF

In regression on networks, a vector of attributes $\mathbf{x}$ and a real-valued response variable $\mathbf{y}$ are observed, while the objective is to predict $\mathbf{y}$ at all nodes given new values for $\mathbf{x}$. GCRF is a discriminative model for regression on an attributed weighted network that models the conditional distribution $P(\mathbf{y}|\mathbf{x})$ over $N$ nodes for $\mathbf{y}$ given $\mathbf{x}$ as:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x},\alpha,\beta)} \exp\Big(\sum_{i=1}^{N} A(\alpha, y_i, \mathbf{x}) + \sum_{j\sim i} I(\beta, y_i, y_j, \mathbf{x})\Big), \tag{1}$$

where $\alpha$ and $\beta$ are parameters of the association $A$ and the interaction $I$ potentials, respectively, and the normalization term $Z(\mathbf{x},\alpha,\beta)$ is an integral over $\mathbf{y}$ of the term in the exponent. The association and interaction potential functions are defined as [15], respectively,

$$A(\alpha, y_i, \mathbf{x}) = -\sum_{k=1}^{K} \alpha_k (y_i - R_k(\mathbf{x}))^2,$$

$$I(\beta, y_i, y_j, \mathbf{x}) = -\sum_{l=1}^{L} \beta_l w^l_{ij} (y_i - y_j)^2,$$

where $R_k(\mathbf{x})$ represents any function that maps $\mathbf{x} \to y_i$ for each node in the network (Figure 1). We refer to this function as unstructured predictor that gives independent predictions. The influence of each unstructured predictor $R_k$ on the final predicted value is modeled by optimizing parameters $\alpha_k$, where $K$ is the number of unstructured predictors. The similarity between two nodes $i$ and $j$ is defined as $w^l_{ij}$. The influence of network structure $w^l$ is modeled through the interaction potential and is weighted by the parameter $\beta_l$, where $L$ is the number of similarity functions.

Modeling association and interaction potentials as quadratic functions of $\mathbf{y}$ enables GCRF to represent (1) as a multivariate Gaussian distribution [15]:

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\mathbf{y}-\mu)^T \mathbf{Q}(\mathbf{y}-\mu)\Big), \quad (2)$$

where $\Sigma^{-1}(:= 2\mathbf{Q})$ is the inverse covariance matrix:

$$\mathbf{Q} = \begin{cases} \sum_k \alpha_k + \sum_h \sum_l \beta_l w^l_{ih} & \text{if } i = j \\ -\sum_l \beta_l w^l_{ij} & \text{if } i \neq j \end{cases} \quad (3)$$

**Inference.** The inference task $argmax_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$ is straightforward. Since GCRF is represented as a multivariate Gaussian distribution, the maximum posterior estimate of $\mathbf{y}$ is obtained by computing the expected value $\mu = \mathbf{Q}^{-1}\mathbf{b}$, where $b_i = \sum_k \alpha_k R_k(\mathbf{x})$.

**Learning.** The learning objective is to optimize the parameters $\alpha$ and $\beta$ by maximizing the conditional log–likelihood

$$ll = \underset{\alpha,\beta}{argmax} \sum_{\mathbf{y}} \log P(\mathbf{y}|\mathbf{x}).$$

The feasibility of the model is ensured when the $\mathbf{Q}$ matrix is positive definite and hence all parameters $\alpha$ and $\beta$ are greater than 0 (constrained optimization). The exponential transformation of parameters is used to make the optimization unconstrained [14], which can be solved by any standard optimization algorithm, such as gradient descent. The resulting negative log-likelihood is convex, which leads to a globally optimal solution of $\alpha$ and $\beta$.

GCRF suffers from the computational explosion issue in large networks. In each iteration, in the gradient descent algorithm computing the inverse of the precision matrix takes $O(N^3)$ time for dense networks and $O(N^2)$ for sparse networks.[1] So, the total running time for learning GCRF is $O(IN^3)$, where $I$ is the number of iterations (for details, please review [15]). In order to reduce the running time of the optimization, a *simple but efficient* trick to eigen-decompose the precision matrix once before the iterations was recently proposed [6], which is called UmGCRF[2]. Once the matrix is decomposed, the eigenvalues are used to perform all other operations inside the iterations, resulting in a model that is much faster than the standard GCRF. The time complexity of UmGCRF is $O(N^3 + IN)$.

[1]Since $Q$ is a Laplacian matrix there exists a fast solver [22] for pseudo-inverse in $O(Mlog(N))$, where $M$ is the number of edges, resulting in $O(I(Mlog(N)+N^2))$ learning time for GCRF, which is still slower than the proposed SA-GCRF model $O(Id^3)$, where $d$ is the node degree $\ll N$ (Sec. II-B4).

[2]The fastest exact state-of-the-art GCRF optimization method.

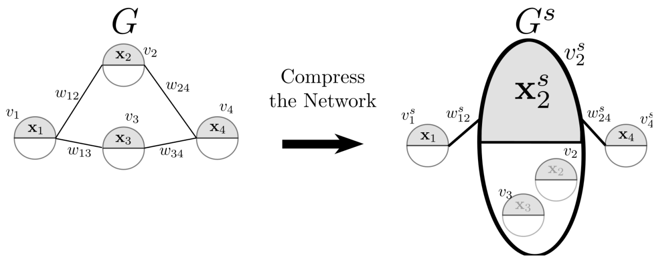


Figure 2. Illustration of Compression. $G$ has $N = 4$ nodes and 4 edges, while $G^s$ has $N^s = 3$ supernodes $\{v_1^s, v_2^s, v_4^s\}$ and two superedges $e_{12}^s$ with weight $w_{12}^s$ and $e_{24}^s$ with weight $w_{24}^s$.

## *B. Graph compression*

The goal of graph compression is to discover hidden structure in a given network and use it to generalize the network into a user-specified size such that the generalized one maintains original information as much as possible. The graph compression algorithms consider only structure and node attributes $\mathbf{x}$.

*1) Preliminaries:* **Definition 1 (Weighted Attributed Network).** A weighted attributed undirected network is defined as a tuple $G = (V, E, \mathcal{W}, \mathcal{X})$, where $V$ is a set of $N$ nodes, $E \subset V \times V$ is a set of edges, $\mathcal{W} : E \to \mathbb{R}^+$ assigns a (non-negative) weight to each edge $e \in E$, and $\mathcal{X}$ is a set of $m$ attributes associated with vertices in $V$, such that the vector $\mathbf{x}_i = \{x_{i1}, \cdots, x_{im}\}$ is a set of numerical attribute values to describe the properties of the node $v_i, \forall i = \{1, 2, \ldots, N\}$.

**Definition 2 (Compressed Network).** A weighted attributed network defined as $G^s = (V^s, E^s, \mathcal{W}^s, \mathcal{X}^s)$ is a compressed representation of $G = (V, E, \mathcal{W}, \mathcal{X})$ if $V^s = \{v_1^s, v_2^s, \cdots, v_{N^s}^s\}$, $v_i^s \subset V$, and $V = \cup_{i=1}^{N^s} v_i^s$, and $v_i^s \cap v_j^s = \emptyset$ for any $i \neq j$. The nodes $v_i^s \in V^s$ are called *supernodes*, and the edges $e_{ij}^s \in E^s$ are called *superedges* between $v_i^s$ and $v_j^s$.

The idea is that a supernode represents all original nodes within it, and that a single superedge represents all edges between its subnodes and their neighbors. For example, in Figure 2, the supernode $v_2^s$ represents the original nodes $v_2$ and $v_3$. The superedge between $v_1^s$ and $v_2^s$ represents two edges: one is between the original nodes $v_1$ and $v_2$, and the other is between the original nodes $v_1$ and $v_3$.

Since the time complexity of GCRF is dominant by node number $N$, the *compression ratio* is defined as follows:

$$cr = \frac{N^s}{N}, \quad (4)$$

which measures how much smaller the compressed network $G^s$ is with respect to the original network $G$. We then measure the compression error, denoted by $dist(G, G^s)$, by calculating (1) the change of node attributes related to each individual node, denoted as $dist(\mathcal{X}, \mathcal{X}^s)$, and (2) the change of connections between any pair of nodes with respect to the original network, denoted as $dist(\mathcal{W}, \mathcal{W}^s)$.

$$\begin{aligned}
dist(\mathcal{X}, \mathcal{X}^s) &= \frac{1}{m}\sum_{j=1}^{m}\sum_{v_i \in V, v_i \in v_k^s}(x_{ij} - x_{kj}^s)^2 \\
dist(\mathcal{W}, \mathcal{W}^s) &= \sum_{\{v_i, v_l\} \in V \times V}(w_{il} - w_{il}^s)^2 \\
dist(G, G^s) &= \Big(dist(\mathcal{X}, \mathcal{X}^s) + dist(\mathcal{W}, \mathcal{W}^s)\Big)^{\frac{1}{2}}, \quad (5)
\end{aligned}$$

where $x_{kj}^s$ represents the jth attribute of the supernode $\mathbf{x}_k^s$ to which the original node $v_i$ belongs, so $dist(\mathcal{X}, \mathcal{X}^s)$ is the total difference between the original attributes of nodes and attributes of their corresponding supernodes. $dist(\mathcal{W}, \mathcal{W}^s)$ is the total change of the original edge weights between any pairs of nodes and the weights of the connections between their corresponding supernodes. For example, in Figure 2, the edge weight between $v_1$ and $v_3$ is changed from $w_{13}$ to $w_{12}^s$. If $v_i$ and $v_l$ are in the same supernode, then $w_{il}^s$ is 0.

**Definition 3 (Graph Compression).** Given a weighted attributed network $G$ and a compression ratio $cr$ ($0 < cr < 1$), the *weighted attributed graph compression problem* is to produce a compressed representation $G^s$ of $G$ with $cr(G^s) \leq cr$ such that $dist(G, G^s)$ is minimized.

The goal of compression is to generalize a network into a user-specified size while maintaining most original information. Another way of using a compression algorithm is to compress the network until a user-defined value of loss in $R^2$ is reached. This approach is out of our scope of this paper and we leave it as future work.

*2) Merge operation:* A network is compressed by merging a pair of nodes iteratively (Section II-B3). A merge operation groups a pair of (super)nodes into a new supernode, and links the new supernode with the neighbors of the merged nodes, and then assigns weights to the new superedges. Thereby, it generalizes the attributes of two merged nodes and the weights of the original edges between the new generalized supernode and the neighbors of the merged node. An illustrated example is shown in Figure 2. Nodes $v_2$ and $v_3$ are chosen to be merged to a new supernode $v_2^s$. The new supernode $v_2^s$ has links with the merged nodes' neighbors, which are $v_1^s$ and $v_4^s$ (here $v_1^s$ is the same as $v_1$, but when the process of compression begins, all nodes are considered as supernodes). The attributes of the two merged nodes $\mathbf{x}_2$ and $\mathbf{x}_3$ are generalized to $\mathbf{x}_2^s$. The edge weights between $v_1$ and $v_2$, and $v_1$ and $v_3$ are generalized to $w_{12}^s$. Similarly, the edge weights between $v_4$ and $v_2$, and $v_4$ and $v_3$ are generalized to $w_{24}^s$. We next discuss the error induced by such merge.

In each iteration, two (super)nodes are merged into a new supernode. Assume that in the tth iteration, (super)nodes $v_m^{s_{t-1}}$ and $v_n^{s_{t-1}}$ are merged into supernode $v_z^{s_t}$. The value of the jth attribute of the supernode $v_z^{s_t}$ is the average of the corresponding values of the two merged nodes, $v_m^{s_{t-1}}$ and $v_n^{s_{t-1}}$, weighted by the corresponding number of nodes within it; that is,

$$x_{zj}^{s_t} = \frac{|v_m^{s_{t-1}}| * x_{mj}^{s_{t-1}} + |v_n^{s_{t-1}}| * x_{nj}^{s_{t-1}}}{|v_m^{s_{t-1}}| + |v_n^{s_{t-1}}|}, \quad (6)$$

where $|v_m^{s_{t-1}}|$ represents the number of subnodes that are within the supernode $v_m^{s_{t-1}}$.

Let $v_l^{s_{t-1}}$ be one of the $k$ neighbors of $v_m^{s_{t-1}}$ and $v_n^{s_{t-1}}$. Similarly, the weight of the superedge between $v_z^{s_t}$ and $v_l^{s_{t-1}}$ is the average of edge weights between two merged nodes, $v_m^{s_{t-1}}$ and $v_n^{s_{t-1}}$, and $v_l^{s_{t-1}}$ weighted by the number of subnodes within $v_m^{s_{t-1}}$ and $v_n^{s_{t-1}}$; that is,

$$w_{zl}^{s_t} = \frac{|v_m^{s_{t-1}}| * w_{ml}^{s_{t-1}} + |v_n^{s_{t-1}}| * w_{nl}^{s_{t-1}}}{|v_m^{s_{t-1}}| + |v_n^{s_{t-1}}|}. \quad (7)$$

Merging a node pair, $v_m^{s_{t-1}}$ and $v_n^{s_{t-1}}$, compared to the previous stage of the network $G^{s_{t-1}}$, only changes the attributes of two merged nodes and the connection weights between the merged nodes and their neighbors. Therefore, we calculate the error caused by a merge locally (Equation 8). In addition, we could have a parameter $\gamma$ to assign a weight to the change of attributes in order to put a different emphasis to the attributes vs. edge weights with respect to the application of the problem. The default value of $\gamma$ could be the average node degree for balancing the amount of attribute change and edge weight change.

$$\begin{aligned}
&err\_merge(v_m^{s_{t-1}}, v_n^{s_{t-1}}) \\
&= \Big(\gamma \cdot \frac{1}{m}\sum_{j=1}^{m}\big((x_{mj}^{s_{t-1}} - x_{zj}^{s_t})^2 + (x_{nj}^{s_{t-1}} - x_{zj}^{s_t})^2\big) \\
&\quad + \sum_{k}\big((w_{ml}^{s_{t-1}} - w_{zl}^{s_t})^2 + (w_{nl}^{s_{t-1}} - w_{zl}^{s_t})^2\big)\Big)^{\frac{1}{2}}. \quad (8)
\end{aligned}$$

*3) Compression algorithms:* We present three algorithms for the weighted attributed graph compression. Similar to Toivonen et al. [20][3], in all algorithms pairs of (super)nodes, which are within 2 hops apart from each other, and also their edges are merged until the specified compression ratio is achieved.

• *Brute-force algorithm.* It iteratively evaluates the merge error of all possible pairwise merges and selects the best merge, and then repeats this until the requested compression ratio is achieved.

• *Random algorithm.* It merges pairs of nodes randomly without any aim to produce a good compression. It provides a baseline for the quality of other methods that make informed decisions about mergers.

• *Semi-greedy algorithm.* This method (outlined in Figure 3) is a hybrid between brute-froce and random algorithms. In each iteration, it first picks a node $v$ at random, and then chooses a node $u$ from the 2-hop neighbors, so that the merge of $v$ and $u$ is optimal with respect to the error caused by the merge.

[3]We did not consider the thresholded algorithm in [20], as the threshold is difficult to choose for different types of networks.

```
Input: Network $G$, compression ratio $cr$
Output: Compressed representation $G^s$
1: $G^s \leftarrow G$ {i.e., $(V^s, E^s, w^s) \leftarrow (V, E, w)$}
2: while $N^s > cr\ N$ do
3:     randomly choose $v^s \in V^s$
4:     for all nodes $u^s$ that are within 2-hops of $v^s$ do
5:         $d_{u^s} \leftarrow err\_merge(v^s, u^s)$
6:     end for
7:     $G^s \leftarrow$ merge $v^s$ with $\arg\min_{u^s} d_{u^s}$
8: end while
```

Figure 3. Semi-greedy Algorithm

*4) Computational complexity for compression:* Let $d$ be the maximum node degree in the original network, and $I = (1 - cr)N$ be the number of iterations. The time complexity of the brute-force algorithm is is $O(Id^3N)$. For the random algorithm, the total complexity is $O(Id)$. The time complexity of semi-greedy algorithm is $O(Id^3)$. The efficiency of the semi-greedy algorithm is highly affected by the node degree $d$ (Section III-B7). In real-world networks, the node degree $d$ is much smaller than $N$, which makes the compression algorithm much faster than GCRF and UmGCRF as evident in our experiments.

### C. Scalable Approximate GCRF (SA-GCRF)

The proposed SA-GCRF model integrates graph compression with GCRF. Our model consists of three steps illustrated in Figure 4. **STEP 1:** it compresses a network into a smaller size by using the weighted attributed graph compression algorithms introduced in Section II-B; **STEP 2:** it applies GCRF to the reduced network to infer the response of supernodes; and **STEP 3:** it unfolds the prediction from supernodes to its subnodes. The prediction is evenly distributed among all its subnodes.

In step 2, UmGCRF is applied in most cases, as it is a fast version of GCRF. For simplicity, we use SA-GCRF without explicitly mentioning that UmGCRF is applied. The only case where GCRF is applied in step 2 is when measuring the speed up (see Sec. III-A). We use $SpeedUP_{Um}$ to specify that UmGCRF is applied, and $SpeedUP_{GR}$ to specify that GCRF is applied.

## III. Experimental evaluation

We present the experimental results using SA-GCRF on both synthetic and real-world datasets. With these experiments we aim to address the following questions: (1) How effective and efficient are weighted attributed graph compression algorithms? (2) How do node attributes affect SA-GCRF regression accuracy? (3) What is the trade-off between the speedup in the running time and the loss in the regression accuracy of SA-GCRF? (4) What is the performance of SA-GCRF on different types of networks with different size and density? and (5) How scalable is SA-GCRF?

### A. Datasets and experimental setup

We briefly describe the networks used in our experiments and their node attributes and edge weights.

**Random networks**. The network structure was generated using an Erdős-Rényi random graph model. We varied the number of nodes $N = \{1K, 5K, 10K, 20K, 30K\}$ generated by the model with average node degree 14. In addition, we fixed the number of nodes $N = 10K$ and varied average node degree $d = \{5, 10, 20, 40, 80\}$, to get another set of random networks.

**Scale-free networks.** The networks were generated using the RTG method [1]. We varied the parameters of the RTG method to obtain networks with sizes $N = \{1K, 3K, 5K, 7K, 9K\}$ with average node degree around 10.

In both random and scale-free networks, for each network setting, we generated 5 networks and computed the statistics as average among all networks of the same size.

**Networks with real-world structure.** To test the scalability of our model on more realistic situations, we also used the network structure extracted from seven real-world datasets ( [2], [5], [9], [25]). The node degree distribution of those seven networks (Table II) follows the power-law distribution. Since all datasets are unlabeled and unweighted networks, we randomly generated link weights and the unstructured predictions from $[0, 1]$. In order to generate the response variable $\mathbf{y}$, we used GCRF as a generative model and set $\alpha = 3$ and $\beta = 1$. (We set $\alpha > \beta$ because in most real-world applications studied elsewhere we found that the unstructured predictions are more influential than the network structure on the response variable.) Finally, we added Gaussian noise $\mathcal{N} \sim (0, 0.5\sigma)$ to the unstructured prediction, where $\sigma$ is the variance of unstructured predictor. Since the compression algorithm uses the unstructured prediction as input, we *added the noise to the unstructured prediction* instead of the response variable $\mathbf{y}$ to test the robustness of SA-GCRF to the noise.

**Real-world weighted attributed networks.** Finally, we evaluated our model on a real-world weighted attributed network. The network used is the High Energy Physics Theory citation network (HEP-TH), which is a bibliographic network extracted from arXiv for the 2003 KDD cup competitions [10]. Node attributes, link weights and response variables are extracted from data (Section III-D).

We evaluated the performance of SA-GCRF using:

• $R^2$ (the regression accuracy of the model): $R^2 = 1 - \frac{\sum_i (y_i - \hat{y_i})^2}{\sum_i (y_i - \bar{y})^2}$, where $y_i$ ($\hat{y_i}$) is the true response value (predicted value) for node $i$ and $\bar{y}$ is the average of $\mathbf{y}$. A score of 0 indicates a poor matching, while a score of 1 indicates a perfect match.

• *Speed up* which measures the speedup of SA-GCRF with respect to UmGCRF [6]. The speed up is computed as $SpeedUP_{Um} = \frac{T_{\text{UmGCRF}}}{T_{\text{Compression}} + T_{\text{UmGCRFComp}}}$, where $T_{\text{UmGCRF}}$ ($T_{\text{UmGCRFComp}}$) is the learning and inference running time

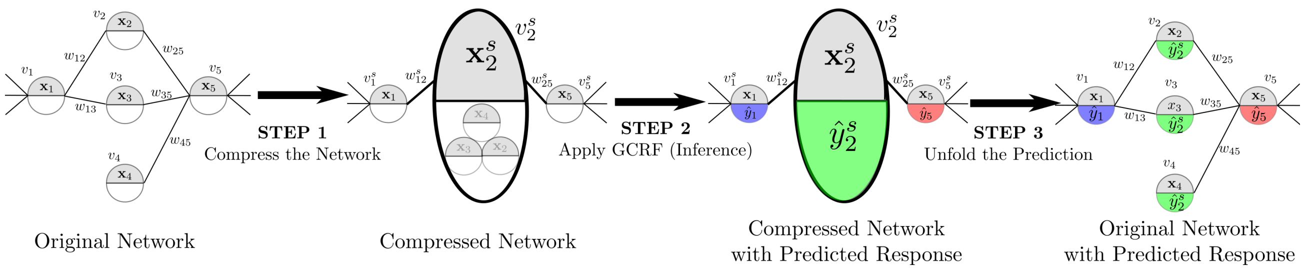


Figure 4. SA-GCRF process. Step 1: Compress a network by generalizing nodes and edges; Step 2: Apply GCRF to the compressed graph to predict the response of each supernode; Step 3: Unfold the predicted response of supernodes to original nodes.

of UmGCRF on the original (compressed) network, respectively, and $T_{\text{Compression}}$ is the running time of graph compression. We also compared our model with the standard GCRF [15]. This speedup is computed as $SpeedUP_{GR} = \frac{T_{\text{GCRF}}}{T_{\text{Compression}}+T_{\text{GCRFComp}}}$, where $T_{\text{GCRF}}$ ($T_{\text{GCRFComp}}$) is the learning and inference running time of GCRF on the original (compressed) network, respectively. All experiments were conducted on a PC with Intel Core i7-3770 3.40 GHz and 32 GB memory.

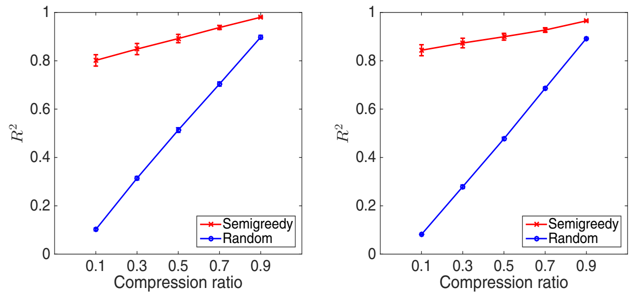
### *B. Performance on synthetic networks*

Experiments were conducted on weighted attributed synthetic networks. Node attributes and link weights were generated according to the procedure described in Section III-A. Each boxplot in Figures 5 – 12 shows results of 5 networks (in some cases, the variance is small and not noticeable).

*1) Effectiveness of SA-GCRF:* The objective of SA-GCRF is to compress a large network to a smaller size on which GCRF is applicable more effectively. Our hypothesis is that the error caused by the compression algorithm will not significantly affect the loss in $R^2$ obtained by GCRF. Figure 5 shows the prediction accuracy $R^2$ of SA-GCRF as a function of compression ratio on two types of synthetic networks: scale-free networks with $9K$ nodes (left panel) and random networks with $30K$ nodes (right panel). The performance on both types of networks are similar. It is clear that SA-GCRF using the semi-greedy compression algorithm produced much more accurate regression results than the one using random compression. The performance of using the brute-force compression algorithm will be shown in Section III-B2.

Compression to half of the original node size using the semi-greedy algorithm, SA-GCRF maintains average $R^2 = 0.89$ on scale-free networks and $R^2 = 0.9$ on random networks. Compression to even a smaller size, such as 10% of original node size, still yields good results. These results indicate that the superiority of SA-GCRF using the semi-greedy algorithm is more obvious when the compression ratio decreases (that is, compressing a network to much smaller size) which will be very useful in large networks.

*2) Comparison of three compression algorithms:* We proceed to compare the performance of three compression algorithms: brute-force, semi-greedy, and random algorithms. Since the brute-force compression algorithm is very computationally expensive and slower than semi-greedy (Section II-B4), we applied brute-force algorithm to scale-free networks with $1K$ nodes using ratio from 0.5 to 0.9. As expected, brute-force algorithm gives better results than semi-greedy compression (Figure 6(a)), but the gap (around 0.04) is not big, and their accuracy is more similar with increased compression ratio and network size.



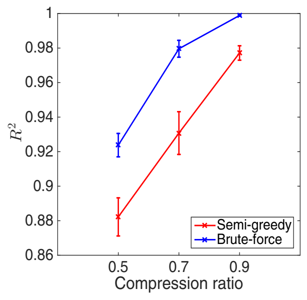
(a) Scale-free networks ($9K$) (b) Random networks ($30K$)

Figure 5. Accuracy comparison between SA-GCRF using the semi-greedy (red) merge and SA-GCRF using random (blue) merge. $R^2$ on the original network is 0.99.

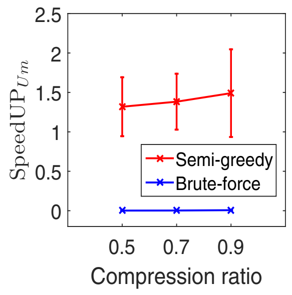
Figure 6(b) shows the speedup by semi-greedy and brute-force algorithms as compared to using UmGCRF. Semi-greedy is 1.5 times faster than brute-force on the networks with $1K$ nodes, which indicates that brute-force compression is not applicable to large size networks. Since the semi-greedy algorithm has comparable accuracy performance to the brute-force algorithm and it is much faster, in the subsequent experiments we will report only the results of SA-GCRF using the semi-greedy algorithm.

*3) Effect of node attributes on regression:* We evaluated the benefit of using the node attributes in improving network regression. Including the node attributes in the compression process would group nodes that have similar structure and similar node attributes. The hypothesis is that including the node attributes in the compression process would make the compressed nodes more homogeneous rather than relying solely on the node structure; hence, the regression on the compressed nodes would be very similar.

We applied SA-GCRF using the semi-greedy algorithm in which node attributes are taken into account, versus the

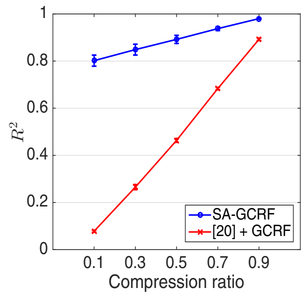


(a) $R^2$

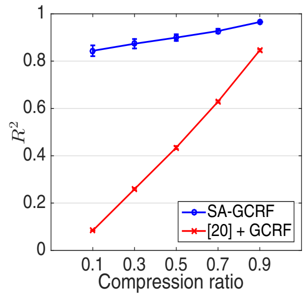


(b) *SpeedUP*$_{Um}$

Figure 6. $R^2$ (left) and *SpeedUP*$_{Um}$ (right) comparison between semi-greedy (red) and brute force (blue) on scale-free networks with size 1K. $R^2$ on the original network is 0.99.



(a) Scale-free networks $(9K)$



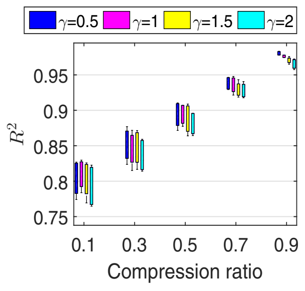
(b) Random networks $(30K)$

Figure 7. Effect of node attributes on the quality of regression. $R^2$ on the original network is 0.99.

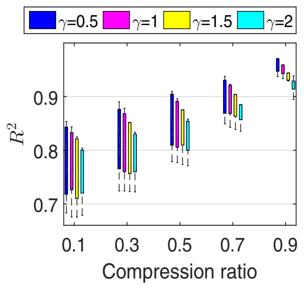
model [20] that does not consider node attributes. The results on both scale-free networks with $9K$ nodes and random networks with $30K$ nodes are shown in Figure 7. It is clear that including the node attributes into the compression algorithm significantly improves the regression results on the compressed network, which validates our hypothesis.

*4) Effectiveness with respect to noise:* In order to evaluate the robustness of SA-GCRF against the noise in the node attributes, we varied the noise level in the unstructured predictions. In particular, we added Gaussian noise $\mathcal{N} \sim (0, \gamma\sigma)$ to unstructured prediction, where $\gamma = \{.5, 1, 1.5, 2\}$. $R^2$ on the original network with these four different noises are 0.99, 0.98, 0.958 and 0.93, respectively. Since the $R^2$ of UmGCRF on the original network decreases with more noisy data, we expect similar accuracy performance of SA-GCRF by adding more noise to unstructured prediction.

We applied SA-GCRF using the semi-greedy algorithm to both scale-free networks $(9K, d = 10)$ and random networks $(10K, d = 10)$ with different noise levels. In both types of networks, when adding more noise to unstructured prediction, the accuracy performance of SA-GCRF decreases (Figure 8), but not significantly. For example, with ratio 0.5 on random networks, $R^2$ decreases from 0.86 to 0.826 when noise level drops from $\gamma = 2$ to $\gamma = 0.5$. The prediction performance with different levels of noise is quite similar especially when compression ratio is small. The compression could smooth out the prediction so that our model is robust to noise in the node attributes.

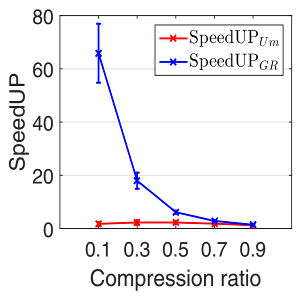


(a) Scale-free networks $(9K)$

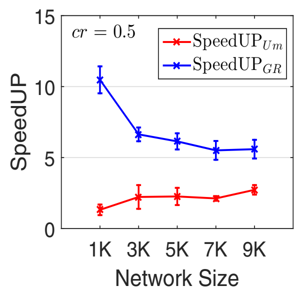


(b) Random networks $(10K)$

Figure 8. Prediction accuracy of SA-GCRF using the semi-greedy algorithm with respect to different noises.



(a) Scale-free networks $(5K)$



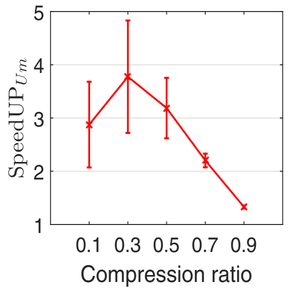
(b) Scale-free networks

Figure 9. Comparison between *SpeedUP*$_{GR}$ (blue) and *SpeedUP*$_{Um}$ (red).

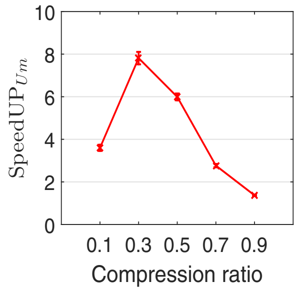
*5) Efficiency of SA-GCRF:* Figure 9(a) shows the speedup of our model with respect to GCRF [15] and UmGCRF [6] using the semi-greedy algorithm on scale-free networks $(5K$ nodes) as a function of compression ratio. SA-GCRF with compression ratio 0.1 is 65 times faster than GCRF on average, whereas it is 1.74 times faster than UmGCRF. The difference between the two speedups is huge, especially when compression ratio is small, but the gap is becoming smaller with the increased compression ratio. The speedup with respect to UmGCRF is not obvious in Figure 9(a) because of the small size of networks (a clear pattern of speedup with respect to UmGCRF can be seen in Figure 10).

Figure 9(b) shows SA-GCRF using the semi-greedy algorithm with a fixed compression ratio (0.5) as a function of network size. *SpeedUP*$_{GR}$ is decreasing with the increased network size, whereas *SpeedUP*$_{Um}$ is increasing, the reason is that GCRF is costly when the size of compressed networks is large. Therefore, we used UmGCRF in our model on large networks in the following experiments.

The efficiency of SA-GCRF with respect to UmGCRF on scale-free networks $(9K, d = 10)$ and random networks $(10K, d = 10)$ are shown in Figure 10. It is clear that SA-GCRF is 2-4 times faster than UmGCRF on scale-free networks, and 2-8 times faster on random networks. The reason that SA-GCRF has high speedup on random networks is that (1) the efficiency of semi-greedy compression is highly affected by node degree, and (2) more compression could cause the compressed network to be denser, as the

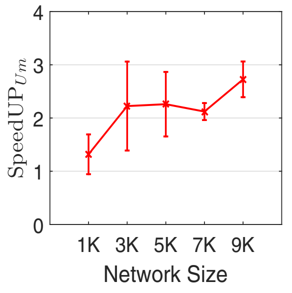


(a) Scale-free networks ($9K$)

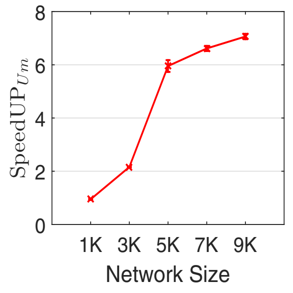


(b) Random networks ($10K$)

Figure 10. Efficiency of SA-GCRF using the semi-greedy algorithm as a function of compression ratio.

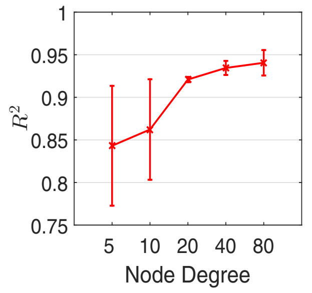


(a) Scale-free networks

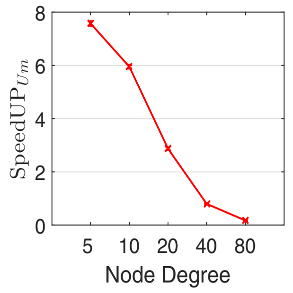


(b) Random networks

Figure 11. Efficiency of SA-GCRF using the semi-greedy algorithm as a function of network size.



(a) $R^2$



(b) *SpeedUP*$_{Um}$

Figure 12. $R^2$ and speedup of SA-GCRF using the semi-greedy algorithm with 0.5 compression as a function of node degree on random networks of sizes 10K. $R^2$ on the original network is 0.98.

new supernode links to all the neighbors of the merged nodes. For example, when compression ratio is small (0.3), in the scale-free networks, a few nodes have quite large node degree which makes compressed network denser and causes compression take more time.

Note that the value of speedup is small when the compression ratio is small (0.1), and it increases until it reaches the peak (when $cr = 0.3$), after which it decreases with increased compression ratio (0.9). The rationale for such behavior is that the running time for UmGCRF is the dominant term in computing the speedup. In particular, when the compression ratio is large (0.9) the compression algorithm is fast but UmGCRF still takes much time to optimize because the compressed network is not small enough to make UmGCRF fast, and vice versa. Therefore, a trade-off between running time of the compression algorithm and UmGCRF is advised. Nevertheless, since the time consumed by UmGCRF is the dominant term, SA-GCRF achieves higher speedup on lower compression ratios (0.3). Together with Figure 5, we see the utility of SA-GCRF on large networks, where it achieves good accuracy while being fast.

*6) Efficiency as a function of network size:* The speedup of SA-GCRF as a function of network size using both types of networks with a fixed compression ratio (0.5) is shown in Figure 11. When the network is small, e.g. $1K$ nodes, the time spent by SA-GCFR is similar to UmGCRF on the original network. However, when the network size increases to $30K$ nodes, SA-GCRF is 7 times faster than UmGCRF on the original network. It is quite obvious that with the increased size of networks, the speedup of SA-GCRF is increasing. It clearly indicates the benefit of applying SA-GCRF on large networks where UmGCRF is not applicable.

*7) Performance of SA-GCRF as a function of network density:* We applied SA-GCRF to random networks ($10K$) with varied node degree. The performance of SA-GCRF with a fixed compression ratio (0.5) is shown in Figure 12. Figure 12(a) shows a very nice pattern: with the increased node degree, the regression accuracy ($R^2$) increases as well. It indicates that for denser networks the loss in prediction accuracy diminishes. However, the time complexity $O(Id^2)$ of SA-GCRF increases with denser networks (Section II-B4). Figure 12(b) verifies this point: when the network is becoming denser, the running time taken by SA-GCRF is getting close to UmGCRF on the original network. Therefore, the utility of SA-GCRF is more evident in sparse networks.

## C. *Scalability of SA-GCRF*

In this section, we discuss the performance of SA-GCRF on large networks, where UmGCRF is not applicable. The network structures were extracted from real-world datasets. The node attributes and link weights were generated following the procedure described in Section III-A. To have an intuition about $R^2$ on the original network, we set zero noise to the unstructured prediction when we generated the response variable, to assume that $R^2$ obtained by UmGCRF on the original network is 1.

Table II shows the effectiveness and efficiency of SA-GCRF with varied compression ratios on large networks. For example, on *CAIDA network* [9] with $26K$ nodes, GCRF takes 12 hours and UmGCRF takes 40.5 minutes to perform regression on the original network, whereas SA-GCRF obtains regression results in 5.92 minutes with 0.93 regression accuracy, which is comparable to the accuracy on the original network. On *Facebook* [23], a relatively dense network, SA-GCRF takes 32 mins to finish regression with 0.7 accuracy due to its high density. However, UmGCRF takes 6 hours, and GCRF takes more than 7 days.

For the networks containing more than one hundred thousand nodes to which UmGCRF is completely inapplicable,

Table II
EFFECTIVENESS AND EFFICIENCY OF SA-GCRF USING THE SEMI-GREEDY ALGORITHM ON LARGE NETWORKS

| Network name | $N$ | $\|E\|$ | Running Time | | SA-GCRF | | SA-GCRF Running time (mins) | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | GCRF [15] | UmGCRF [6] | $cr$ | $R^2$ | $T_{\text{Compression}}$ | $T_{\text{UmGCRFComp}}$ | Total |
| CAIDA [9] | 26,475 | 53,381 | **12 hours**[1] | **40.5 mins** | 0.5 | **0.93** | 0.84 | 5.08 | **5.92** |
| | | | | | 0.3 | 0.90 | 1.02 | 1.12 | 2.15 |
| | | | | | 0.1 | 0.86 | 1.33 | 0.05 | 1.38 |
| Facebook friend-ships [23] | 63,392 | 816,831 | $>$ 7 **days**[1] | **6 hours** | 0.5 | 0.77 | 11.87 | 43.78 | 55.65 |
| | | | | | 0.3 | **0.70** | 22.83 | 9.29 | **32.11** |
| | | | | | 0.1 | 0.65 | 68.19 | 0.39 | 68.58 |
| WordNet [5] | 145,145 | 656,230 | NA | NA | 0.2 | 0.69 | 2.83 | 52.23 | 55.06 |
| | | | | | 0.15 | 0.68 | 3.08 | 22.31 | 25.39 |
| | | | | | 0.1 | 0.65 | 3.54 | 6.64 | 10.18 |
| Gowalla [2] | 196,591 | 950,327 | NA | NA | 0.15 | 0.75 | 95.77 | 53.88 | 149.65 |
| | | | | | 0.1 | 0.73 | 158.72 | 16.09 | 174.81 |
| | | | | | 0.05 | 0.71 | 214.06 | 2.18 | 216.24 |
| Amazon [25] | 334,863 | 925,872 | NA | NA | 0.1 | 0.73 | 0.95 | 86.14 | 87.09 |
| | | | | | 0.05 | 0.70 | 1.92 | 10.36 | 12.28 |
| | | | | | 0.01 | 0.65 | 9.99 | 0.08 | 10.07 |
| DBLP [25] | 317,080 | 1,049,866 | NA | NA | 0.1 | 0.71 | 2.30 | 68.92 | 71.22 |
| | | | | | 0.05 | 0.67 | 4.23 | 8.72 | 12.95 |
| | | | | | 0.01 | 0.66 | 19.23 | 0.08 | 19.31 |
| Youtube [25] | 1,134,890 | 2,987,624 | **NA** | **NA** | 0.03 | 0.76 | 3492.9 | 86.4 | **3579.3** |

[1] The experiment was conducted on a sever with 128 GB of memory and Intel(R) Xeon(R) E5-2630 v2 2.60GHz processors.

we chose a compression ratio with which UmGCRF is applicable to the compressed network. For example, on *WordNet network* [5] that contains $145K$ nodes, SA-GCRF could finish regression in 10 mins with 0.1 compression ratio. Compared to the result with 0.2 compression ratio, there is a large decrease in running time, from 1 hour to 10 mins, with little decrease in regression accuracy.

*Gowalla* [2] has $196K$ nodes, which is smaller than *Amazon network* [25] and *DBLP collaboration network* [25], however, it is much denser. Hence, SA-GCRF takes more time, 174.81 mins, to finish regression with 0.73 accuracy.

On *Amazon* and *DBLP* networks, which contain similar number of nodes ($\sim 300K$) and edges ($\sim 1M$), SA-GCRF takes around 12 mins to compress the network to 5% of original size and finishes the regression with 0.7 accuracy.

On *Youtube social network* [25] that consists of $1M$ nodes and $2M$ edges, SA-GCRF takes 3579 mins (=59.65 hours) to reach a 0.79 regression accuracy with a compression ratio 0.03. SA-GCRF training was costly, but much more efficient than UmGCRF and GCRF, which could not provide results in an acceptable time on such big networks.

From Table II, we conclude that SA-GCRF allows *effective* and *efficient* learning and inference for structured regression on networks with *hundreds of thousands* of nodes.

### D. Performance on citation networks

The effectiveness of SA-GCRF is further characterized on real-world weighted attributed networks with the objective of predicting the number of citations in a high-energy physics citations network [10]. The dataset consists of 29,955 papers (nodes) and 352,807 citations spanning over 11 years. Following [17], [21], we filtered out papers that received less than 3 citations from 2000 to 2002, selected papers that were published before January 2000, and collected their citation numbers received after January 2000 on a half-yearly basis. Therefore, we constructed six weighted attributed networks for the period 2000-2002. Each node in the network represents a paper and the attribute of the node is the number of citations the paper received in the last 6 months (Figure 13). Two papers $A$ and $B$ are connected based on the coCiter measure [21], which is computed as:

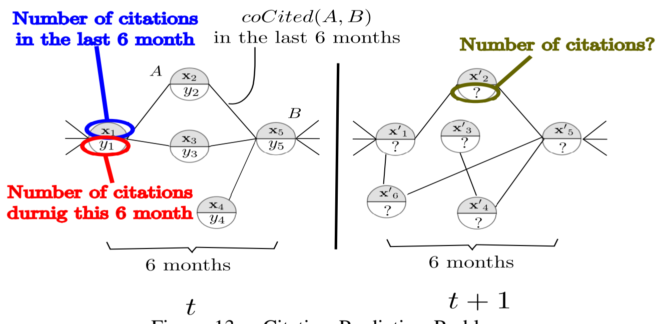

Number of citations in the last 6 month
$coCited(A,B)$ in the last 6 months
Number of citations?
$A$
$B$
Number of citations durnig this 6 month
6 months
6 months
$t$
$t+1$

Figure 13. Citation Prediction Problem.

Table III
AVERAGE (STANDARD DEVIATION) $R^2$ OF SA-GCRF ON PREDICTING CITATION FOR THE FOLLOWING 6 MONTHS IN A NETWORK OVER 5 TIME POINTS. AVERAGE $R^2$ ON ORIGINAL NETWORK OF UMGCRF IS 0.823($\pm$0.042).

| $cr$ | $R^2$ |
|---|---|
| 0.9 | 0.825($\pm$0.043) |
| 0.5 | 0.814($\pm$0.043) |
| **0.1** | 0.766($\pm$0.055) |

$$sim_{coCiter}(A,B) = \frac{2 \times \#\text{ cocitations of A and B}}{\#\text{ citations of A} + \#\text{ citations of B}}.$$

The regression problem is, based on the number of citations obtained from the current 6 months, to predict the number of citations in the following 6 months. We train the model on the network at time $t$ and test it on the network at time $t+1$, where $t = \{1, 2, \ldots, 5\}$. The average $R^2$ of SA-GCRF over 5 time points $t$ are shown in Table III.

We conclude from Table III: (1) SA-GCRF has very comparable results with respect to UmGCRF on the citation network even when using compression ratio 0.1, which is consistent with results on synthetic networks; and (2) The prediction obtained by SA-GCRF when $cr = 0.9$ is *surprisingly* even better than the one obtained by UmGCRF, which indicates that the compression helped the model to overcome the noise and smooth out the prediction such that it became robust to outliers.

## IV. Conclusion

We proposed a novel model (SA-GCRF) that integrates graph compression with GCRF for structured regression. The goal is to make GCRF applicable to large networks. The model is based on the hypothesis that the compressed network maintains most information of the original network, such that the loss in regression accuracy obtained by GCRF is minor. In particular, the weighted attributed graph compression extends the existing approach to take node attributes into account during compression.

We conducted comprehensive experiments to test our hypothesis on synthetic and real-world networks, and conclude that: (1) SA-GCRF allows efficient and effective learning and inference on large networks; (2) SA-GCRF is robust to the noise in the node attributes; (3) Node weights can guide the compression process to better group nodes with similar attributes, such that the regression accuracy is improved.

Our work in progress is how to utilize the domain knowledge to further guide the compression and unfolding to achieve better predictions. Besides, we will integrate graph compression with other structured models to evaluate the performance of regression in large networks. Finally, we will study the theoretical guarantee for our model.

## Acknowledgment

This research was supported in part by DARPA grant FA9550-12-1-0406 negotiated by AFOSR, NSF BIGDATA grant 14476570 and ONR grant N00014-15-1-2729.

## References

[1] L. Akoglu and C. Faloutsos. RTG: A Recursive Realistic Graph Generator Using Random Typing. *DATA MIN KNOWL DISC*, 19(2):194–209, 2009.

[2] E. Cho, S. A. Myers, and J. Leskovec. Friendship and Mobility: User Movement in Location-based Social Networks. In *SIGKDD*, pages 1082–1090. ACM, 2011.

[3] R. R. Coifman and M. Maggioni. Diffusion wavelets. *APPL COMPUT HARMON A*, 21(1):53–94, 2006.

[4] N. Djuric, V. Radosavljevic, Z. Obradovic, and S. Vucetic. Gaussian Conditional Random Fields for Aggregation of Operational Aerosol Retrievals. *IEEE Geosci. Remote Sens Lett*, 12(4):761–765, 2015.

[5] C. Fellbaum, editor. *WordNet: an Electronic Lexical Database*. MIT Press, 1998.

[6] J. Glass, M. Ghalwash, M. Vukicevic, and Z. Obradovic. Extending the Modelling Capacity of Gaussian Conditional Random Fields while Learning Faster. In *AAAI*, 2016.

[7] K. Kersting. Lifted probabilistic inference. In *ECAI*, pages 33–38, 2012.

[8] J. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, pages 282–289, 2001.

[9] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph Evolution: Densification and Shrinking Diameters. *ACM Trans. Knowledge Discovery from Data*, 1(1):1–40, 2007.

[10] J. N. Manjunatha, K. R. Sivaramakrishnan, R. K. Pandey, and M. N. Murthy. Citation Prediction Using Time Series Approach KDD Cup 2003 (Task 1). *SIGKDD Explor. Newsl.*, 5(2):152–153, 2003.

[11] N. Mueller, K. Haegler, J. Shao, C. Plant, and C. Böhm. Weighted Graph Compression for Parameter-free Clustering With PaCCo. In *SDM*, pages 932–943, 2011.

[12] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph Summarization with Bounded Error. In *SIGMOD*, pages 419–432, 2008.

[13] A. Polychronopoulou and Z. Obradovic. Hospital Pricing Estimation by Gaussian Conditional Random Fields Based Regression on Graphs. In *BIBM*, pages 564–567, 2014.

[14] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li. Global Ranking Using Continuous Conditional Random Fields. In *NIPS*, pages 1281–1288, 2009.

[15] V. Radosavljevic, S. Vucetic, and Z. Obradovic. Conditional Random Fields for Regression in Remote Sensing. In *ECAI*, pages 809–814, 2010.

[16] K. Ristovski, V. Radosavljevic, S. Vucetic, and Z. Obradovic. Continuous Conditional Random Fields for Efficient Regression in Large Fully Connected Graphs. In *AAAI*, pages 840–846, 2013.

[17] J. Slivka, M. Nikolic, K. Ristovski, V. Radosavljevic, and Z. Obradovic. Distributed Gaussian Conditional Random Fields Based Regression for Large Evolving Graphs. In *Proc. 14th SDM Workshop on Mining Networks and Graphs*, 2014.

[18] M. F. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman. Learning Gaussian Conditional Random Fields for Low-Level Vision. In *CVPR*, 2007.

[19] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient Aggregation for Graph Summarization. In *SIGMOD*, pages 567–580, 2008.

[20] H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka. Compression of Weighted Graphs. In *SIGKDD*, pages 965–973, 2011.

[21] A. Uversky, D. Ramljak, V. Radosavljević, K. Ristovski, and Z. Obradović. Which Links Should I Use? A Variogram-based Selection of Relationship Measures for Prediction of Node Attributes in Temporal Multigraphs. In *ASONAM*, volume 4(1), pages 211–224, 2013.

[22] N. K. Vishnoi. Laplacian solvers and their algorithmic applications. *THEOR COMPUT SCI*, 8(1-2):1–141, 2012.

[23] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the Evolution of User Interaction in Facebook. In *Proc. Workshop on Online Social Networks*, pages 37–42, 2009.

[24] M. Wytock and Z. Kolter. Sparse Gaussian Conditional Random Fields: Algorithms, Theory, and Application to Energy Forecasting. In *ICML*, pages 1265–1273, 2013.

[25] J. Yang and J. Leskovec. Defining and Evaluating Network Communities Based on Ground-truth. In *MDS*, pages 1–8, 2012.

[26] N. Zhang, Y. Tian, and J. Patel. Discovery-Driven Graph Summarization. In *ICDE*, pages 880–891, 2010.

[27] F. Zhou, H. Toivonen, and R. D. King. The Use of Weighted Graphs for Large-Scale Genome Analysis. *PLoS ONE*, 9(3):e89618, 2014.