Scalable Explanation of Inferences on Large Graphs

Chao Chen
Lehigh University
Bethlehem, PA, USA
chc517@lehigh.edu

Yifei Liu and Xi Zhang
Key Laboratory of Trustworthy Distributed Computing and Service
Ministry of Education, BUPT, Beijing, China
{*liuyifei, zhangx*}*@bupt.edu.cn*

Sihong Xie
Lehigh University
Bethlehem, PA, USA
six316@lehigh.edu

***Abstract*—Probabilistic inferences distill knowledge from graphs to aid human make important decisions. Due to the inherent uncertainty in the model and the complexity of the knowledge, it is desirable to help the end-users understand the inference outcomes. Different from deep or high-dimensional parametric models, the lack of interpretability in graphical models is due to the cyclic and long-range dependencies and the byzantine inference procedures. Prior works did not tackle cycles and make *the* inferences interpretable. We formulate the explaination of probabilistic inferences as a constrained cross-entropy minimization problem to find simple subgraphs that faithfully approximate the inferences. We prove that the optimization is NP-hard, while the objective is not monotonic and submodular to guarantee efficient greedy approximation. We propose a beam search algorithm to find trees to enhance the explanation interpretability and diversity. To allow efficient search on large and dense graphs without hurting faithfulness, we further propose parallelization and a pruning strategy. We demonstrate superior performance on four networks from distinct applications, comparing favorably to other explanation methods, including LIME.**

***Keywords*-interpretability, graphical model**

I. Introduction

Distilling knowledge from graphs is an important task found ubiquitously in applications, such as fraud detection in social networks and drug discovery in bioinformatics. The knowledge helps humans make high-stake decisions, such as whether to trust a business or an account on Yelp, or to conduct expensive experiments on a promising protein in drug discovery. The state-of-the-art approaches model the graphs as directed or undirected graphical models, such as Markov Random Fields (MRF). Different from predictive models on i.i.d. vectors, graphical models capture the dependencies among random variables. However, the inferences involve iterative and recursive computations, and are cognitively difficult to understand, verify, and ratify, locking away more applications of graphical models. We focus on belief propagation (BP) inferences that compute marginal distributions on MRFs, aiming to make the inference outcomes more interpretable and cognitively easier for humans. Figure 1 depicts the problem definition and the proposed solution.

Several challenges are due. First, simple but faithful explanations are desired [10] but have not been defined for inferences on MRFs. It is less known what's the best *graph* explanation complexity and faithfulness trade-off for

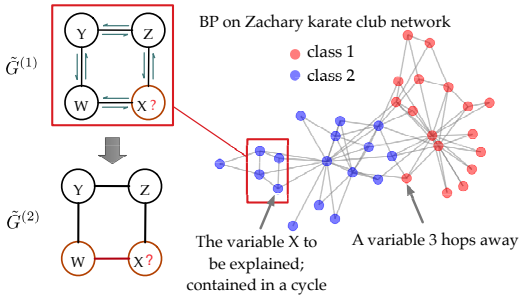

Figure 1. A cyclic graphical model G for the Zachary karate club network, with BP inference outcomes shown in two colors. We focus on explaining how BP calculates the belief on X, highlighted in the subgraph $\tilde{G}^{(1)}$. Due to the cycles and long-range dependencies on G, a complete explanation is recursive and long. GraphExp extracts a limited-size *tree* $\tilde{G}^{(2)}$, on which X has a marginal similar to that on G.

the end-users. Second, algorithmically, an MRF can be large, densely connected, and cyclic, while a simple, faithful and no-cyclic explanations need to be found efficiently.

We propose a new approach called "GraphExp" to address the above challenges. Given the full graphical model G and any target variable X on G, GraphExp finds an "explanans" graphical model $\tilde{G}$ consisting of a smaller number of random variables and dependencies. The goal of GraphExp is to minimize the loss in faithfulness measured by the symmetric KL-divergence between the marginals of X inferred on $\tilde{G}$ and G [19]. Starting from the graph $\tilde{G}^{(1)}$ consisting of X only, GraphExp greedily includes the next best variable into the previous subgraph. Theoretically, we prove that: **(1)** an exhaustive search for the optimal $\tilde{G}$ with highest faithfulness is NP-hard, and furthermore, the objective function is neither monotonic nor submodular, leading to the lack of a performance guarantee of any greedy approximation (Theorem 1); **(2)** GraphExp only generates acyclic graphs that are more explainable (Theorem 2).

There can exist multiple sensible explanations for the same inference outcome, and end-users can find the one that best fits their mental model. We equip GraphExp with beam search [1] to discover a set of distinct, simple, and faithful explanations for a target variable. Regarding scalability, when looking for $\tilde{G}$ on densely connected graphs, the branching factor in the search tree can be too large for fast search. While the search is parallelizable, we propose a safe

Table I
Comparing GraphExp with prior explanation methods: FS (Feature Selection), LIME, GSparse (graph sparsification), and GDiff (graph differentiation). (∗: "surely yes"; ○: "partially"; emptiness: "no").

	FS [7]	LIME [17]	GSparse [6]	GDiff [2]	GraphExp
Cycles handling			∗	∗	∗
Completeness			○	∗	∗
Interpretability	○	∗	○	○	∗
Diversity	○	○			∗
Scalability		○		○	∗
Flexibility		○		○	∗

pruning strategy that retains the desirable candidates while cutting the search space down significantly (Figure 2).

II. Problem definition

Notation definitions are summarized in Table II. Given a set of n random variables $V = \{X_1, \ldots, X_n\}$, each taking values in $\{1, \ldots, c\}$ where c is the number of classes, BP inference computes messages $m_{i\to j}(x_j)$ from X_i to X_j:

$$\frac{1}{Z_j} \sum_{x_i} \left[\psi_{ij}(x_i, x_j)\phi_i(x_i) \prod_{k\in\mathcal{N}(X_i)\setminus\{j\}} m_{k\to i}(x_i) \right], \quad (1)$$

where Z_j is a normalization factor, ϕ_i is the prior distribution matrix of X_i without considering other variables. The compatibility $\psi_{ij}(x_i, x_j)$ encodes how likely the pair (X_i, X_j) will take the value (x_i, x_j) jointly. The messages on all edges $(i, j) \in E$ are updated until convergence. The belief b_X is

$$b_i(x_i) \propto \phi_i(x_i) \prod_{X_j\in\mathcal{N}(X_i)} m_{j\to i}(x_i), \quad (2)$$

where $\mathcal{N}(X_i)$ is the neighbors of X on G. We aim to explain how the marginal is inferred by BP. For any $X \in V$, b_X depends on messages over all edges reachable from X. To completely explain how b_X is computed, one has to trace down each message in Eq. (1) and Eq. (2). Such a complete explanation is hardly interpretable due to two factors: 1) on large graphs with long-range dependencies, messages and variables far away from X will contribute to b_X indirectly through many steps; 2) when there is a cycle, BP needs multiple iterations to converge and a message can be recursively defined by itself. However, the graph easily becomes too large for humans to interpret or analyze. Rather, b_X should be approximated using short-range dependencies without iterative and recursive computations. The question is, without completely following the original BP computations, how will the approximation affected? To answer this question, we formulate the following optimization problem:

Definition 1. *Given an MRF G, and a target node $X \in V$, extract another MRF $\tilde{G} \subset G$ such that $X \in \tilde{G}$ and $\tilde{G}$ containing no more than C variables and no cycle, so that BP computes similar marginals b_X and $\tilde{b}_X$ on G and $\tilde{G}$, respectively. Formally, solve the following*

$$\min_{\tilde{G}} d(b_X, \tilde{b}_X) = \mathrm{KL}(b_X||\tilde{b}_X) + \mathrm{KL}(\tilde{b}_X||b_X) \quad \text{s.t. } \tilde{G} \subset G, \quad |\tilde{G}| \leq C, \quad X \in \tilde{G}, \quad \tilde{G} \text{ acyclic.} \quad (3)$$

Table II
Notation Definitions

Notation	Definition
$G = (V, E)$	Undirected graphical model (MRF)
V, E	Random varaibles and their connections
X_i, X (x_i, x)	Random variables (and their values)
$\phi_X(x)$ (or $\phi_i(x_i)$)	Prior probability distribution of X (or X_i)
$\psi_{ij}(x_i, y_j)$	Compatibility matrix between x_i and x_j
$m_{j\to i}(x_i)$	Message passed from x_j to x_i
$b_X(x) \triangleq P(X = x)$	Marginal distribution (belief) of X
$\mathrm{KL}(p\|\|q)$	KL Divergence between p and q
$\partial G'$	Variables in $G \setminus G'$ connected to G'
$\mathcal{N}(X_i)$	Neighbors of X_i on G

The objective captures the faithfulness of $\tilde{G}$, measured by the symmetric KL-divergence d between marginal distributions of X on G and $\tilde{G}$, where $\mathrm{KL}(P||Q) = \sum_{x=1}^{c} P(x) \log[P(x)/Q(x)]$. The choice of d as a faithfulness measured can be justified: $\mathrm{KL}(b_X||\tilde{b}_X)$ measures the loss when the "true" distribution is b_X while the explaining distribution is $\tilde{b}_X$ [19]. Symmetrically, a user can regard $\tilde{b}_X$ as the "true" marginal, which can be explained by the b_X. The simplicity of $\tilde{G}$ can be measured by the number of variables on $\tilde{G}$, and for $\tilde{G}$ to be interpretable, we control the size of $\tilde{G}$ to be no more than C. Since a graphical model encodes a joint distribution of a set of variables, searching $\tilde{G}$ is equivalent to searching a joint distribution of a smaller number of variables with fewer dependencies, so that the two joint distributions lead to similar marginals of X.

If the negation of the above objective function is submodular and monotonically increasing, then a greedy algorithm can generate a solution whose value is within $(1 - 1/e)$ of the optimum [14]. The greedy algorithm iteratively builds $\tilde{G}$ by adding one variable at a time to increase the negation of objective function (namely, to decrease $d(b_X, \tilde{b}_X)$).

Definition 1 (Submodularity)**.** *Let $\mathcal{V}$ be a set and $2^{\mathcal{V}}$ be the power set of $\mathcal{V}$. A set function $f : 2^{\mathcal{V}} \to \mathbb{R}$ is submodular if $\forall \mathcal{A} \subset \mathcal{A}' \subset \mathcal{V}$ and any $Y \notin \mathcal{A}'$, $f(\mathcal{A} \cup \{Y\}) - f(\mathcal{A}) \geq f(\mathcal{A}' \cup \{Y\}) - f(\mathcal{A}')$.*

Definition 2. *A set function $f : 2^{\mathcal{V}} \to \mathbb{R}$ is monotonically increasing if $\forall \mathcal{A} \subset \mathcal{A}' \subset \mathcal{V}$, implies $f(\mathcal{A}) \leq f(\mathcal{A}')$.*

Theorem 1. *The objective function in Eq. (3) is not submodular nor monotonically increasing.*

Proof: See [4] for the detailed proof. ■

III. METHODOLOGIES

Due to the iteration of BP, the explanation subgraph should be acyclic to avoid the target node being explained by itself. The optimization problem Eq. (3) can be solved by exhaustive search in the space of all possible trees under the specified constraints and it is thus a combinatorial subset maximization problem and NP-hard. Greedy algorithms are a common solution to approximately solve NP-hard problems. Since finding multiple alternative sensible explanations is one of our goals, we adopt beam search in the greedy search [1], maintaining in a beam several top candidates ranked by faithfulness and succinctness through the search.

Algorithm 1 GraphExp (the general search framework)

Input:a graphical model $G = (V, E)$; a target random variable $X \in V$ to be explained;
prior ϕ_i and belief b_i, $\forall X_i \in V$; messages $m_{i\to j}$ for $\forall (X_i, X_j) \in E$;
maximum subgraph complexity C; beam size k.
Output: Multiple explaining subgraphs $\tilde{G}$ for X, along with approximated computations of b_X.
Init: $\tilde{G}^{(1)} = (\tilde{V}, \tilde{E})$, $\tilde{V} = \{X\}, \tilde{E} = \emptyset$. `Beam[1]`=$\{\tilde{G}^{(1)}\}$
for $t = 2 \to C$ **do**
  `Beam[t]`$= \emptyset$.
  for each subgraph $\tilde{G}^{(t-1)}$ in `Beam[t-1]` **do**
    Find and add the top k extensions of $\tilde{G}^{(t-1)}$ to `Beam[t]`.
  end for
  Retain the top k candidates in `Beam[t]`.
end for
Run BP on each candidate graph in `Beam[C]` and obtain converged messages and beliefs as an approximation of computation of b_X on G.

Theorem 2. *The output $\tilde{G}$ from Algorithm 1 is a tree.*

A general greedy beam search framework is presented in Algorithm 1. The algorithm finds multiple explaining trees $\tilde{G}^{(C)}$ of size C in $C-1$ iterations, where C is the maximum subgraph size. Here, we explicitly illustrate how a single $\tilde{G}^{(C)}$ is constructed. Starting from the initial $\tilde{G}^{(1)} = \{X\}$, at each step t, the graph $\tilde{G}^{(t-1)}$ is extended to $\tilde{G}^{(t)}$ by adding one more explaining node and edge to optimize a certain objective function without forming a loop. After all $\tilde{G}^{(C)}$ are found and right before the algorithm exits, BP will be run again on each $\tilde{G}^{(C)}$ in `Beam[C]` to compute $\tilde{b}_X$ so that we can use the converged messages on $\tilde{G}^{(C)}$ to explain to an end-user how b_X is approximated on $\tilde{G}^{(C)}$. Since $\tilde{G^{(C)}}$ is small and contains no cycle, the explanation is significantly simpler than the original computations on G. We substantiate the general framework with two alternative ways to rank candidate extensions of the trees during the beam search.

A. GraphExp-Global (*GE-G*): search explanations via evaluating entire subgraphs

We propose **GE-G**, an instantiation of Algorithm 1. At iteration t $(t = 2, \ldots, C)$, the algorithm evaluates the candidate extensions of $\tilde{G}^{(t-1)}$ using Eq. (3). Define $\partial\tilde{G}^{(t-1)}$ be the set of nodes in $G \setminus \tilde{G}^{(t-1)}$ that are connected to $\tilde{G}^{(t-1)}$. A candidate $\tilde{G}^{(t)}$ is generated by adding a variable $Y \in \partial\tilde{G}^{(t-1)}$ through the edge (Y, W) to $\tilde{G}^{(t-1)}$, where W is a random variable in $\tilde{G}^{(t-1)}$. A new BP procedure is run on $\tilde{G}^{(t)}$ to infer $\tilde{b}_X$, and $d(b_X, \tilde{b}_X)$ is calculated as the quality of the candidate $\tilde{G}^{(t)}$. After exhausting all possible candidates, **GE-G** adds the k candidates with the smallest $d(b_X, \tilde{b}_X)$ to `Beam[t]`.

On the high-level, **GE-G** is similar to forward wrapper-style feature selection algorithms, where each feature is evaluated by including it to the set of selected features and running a target classification model on the new feature sets. The key difference here is that **GE-G** can't select any variable on G, but has to restrict itself to those that will result in an acyclic graph (guaranteed by Theorem 2).

One of the advantages of **GE-G** is that the objective function in the optimization problem Eq. (3) is minimized directly at each greedy step. However, as each candidate at each step requires a separate BP, it can be time-consuming. We analyze the time complexity below. To generate a subgraph of maximum size C for a variable, $C-1$ iterations are needed. At iteration $t = 2, \ldots, C$, one has to run BP for as many times as the number of neighboring nodes of the current explanation $\tilde{G}^{(t-1)}$. The number of candidates that need to be evaluated for one of the k candidates in `Beam[t-1]` equals the size of the number of neighboring nodes in $\partial\tilde{G}^{(t-1)}$. On graphs with a small diameter, this size grows quickly to the number of nodes in G. On the other extreme, if G is a linear chain, this size is no more than 2. For each BP run, it is known that BP will converge in the number of iterations same as the diameter of the graph that BP is operated on, which is upper-bounded by the size of the candidate subgraph $\tilde{G}^{(t)}$. During each BP iteration, $O(t)$ messages have to be computed. The overall time complexity of **GE-G** is $O(|V|k\sum_{t=2}^{C}|\partial\tilde{G}^{(t-1)}|t^2)$, where k is the beam size. Since the number of classes on the variables are fixed and usually small (relative to the graph size), here we ignore the factor $O(c^2)$, which is the time complexity to compute one message using Eq. (1).

B. Speeding up *GE-G* on large graphs

Graphical models in real-world applications are usually gigantic containing thousands of nodes. We propose parallelized search and a pruning strategy to speed up **GE-G**.

Parallel search The general GraphExp algorithm can be parallelized on two levels. First, the generation of explanations over multiple target variables can be executed on multiple cores. Second, in the evaluation of the next extensions of

$\tilde{G}^{(t-1)}$ during beam search, multiple candidates $\tilde{G}^{(t)}$ can be tried out at the same time on multiple cores.

Pruning candidate variables In Algorithm 1, all candidates in $Q = (\tilde{G}^{(t)}, \partial\tilde{G}^{(t)})$ have to be evaluated at step t, and we have to run BP as many times as $|Q|$. As we aim at explaining how BP infers the marginal of the target X, adding any variable that causes a larger $d(b_X, \tilde{b}_X)$ is not helpful but confusing. Since most candidates in $\partial\tilde{G}^{(t-1)}$ remains in future steps, removing confusing candidates from all future steps would speed up the algorithm. Thus, we run BP on $|Q|$ candidates and abandon the bottom $(100 - p)$ percent of the candidates ranked by $d(b_X, \tilde{b}_X)$ in ascending.

*C. GraphExp-Local (**GE-L**): search explanations via local message back-tracing*

Sometimes one may want to trade explanation faithfulness for speed during subgraph search. For example, in the exploratory phase, a user wants to identify mistakes caused by glitches on the graph before digging deeper into finer-grained explanations. We propose **GE-L** for this purpose to complement **GE-G**, which generates more faithful and detailed explanations using larger search space (with the expense of more searching time). **GE-L** is based on message back-tracing that follows the general beam search but with more constraints on the search space. At step t, the search adds an edge between $\tilde{G}^{(t-1)}$ and $\partial\tilde{G}^{(t-1)}$ that best explains a message or a belief in $\tilde{G}^{(t-1)}$. There are two cases.

- For a message $m_{W\to X}$ on an edge (W, X) already included in $\tilde{G}^{(t-1)}$, the search attempts to find a message $m_{Y\to W}$, where $Y \in \partial\tilde{G}^{(t-1)}$, so that the message $m_{Y\to W}$ contributes most to the message $m_{W\to X}$. We use the distance d defined in Eq. (3) to measure the contribution of $m_{Y\to W}$ to $m_{W\to X}$: the smaller the distance, the more similar two messages are and thus more contribution from $m_{Y\to W}$ to $m_{W\to X}$.
- For the belief b_X of the target node X, the search attempts to find a message $m_{W\to X}$ that best explains b_X, using the distance between $m_{W\to X}$ and b_X.

In **GE-L**, we define the **end point** of $\tilde{G}^{(t)}$ as either the target node in $\tilde{G}^{(1)}$ or the node last added to the subgraph in $\tilde{G}^{(t)}$ ($t \geq 2$). In the example in Figure 1, the end point in $\tilde{G}^{(1)}$ is X, and it is W in $\tilde{G}^{(2)}$. If the prior of the target node (ϕ_X) best explains its belief (b_X) in $\tilde{G}^{(1)}$, or the prior of the non-target end point (ϕ_W) best explains the message emitting from the end point (e.g., $m_{W\to X}$) in $\tilde{G}^{(t)}$ ($t \geq 2$), the search of **GE-L** will stop at the end point and no extension at this branch: the priors are fixed input to BP and the priors at the end points best explain the previous message or the belief.

Variants of GE-L To further speed up **GE-L** (see Figure 2), especially on graphs that a user has prior knowledge about the topology of the explaining subgraphs, its search space can be further constrained. On the one hand, we can only extend a candidate on the end-point that is added most recently, creating a chain of variables so that one is explaining the other. This aligns with the conclusion that causal explanations are easier to be understood by the end-users [10], and our explanations of the inference are indeed causal: how $\tilde{b}_X$ is computed by a smaller subgraph will be presented to the end-users. On the other hand, when a target variable has many incoming messages (which is the case on social and scientific networks), it is best to spend the explaining capacity on direct neighbors. In the experiments, we adopt these two constraints over **GE-L** on all networks, respectively. [4] shows more details for **GE-G** and **GE-L**.

Table III
Ten networks from four application domains.

Datasets	Classes	Nodes	Edges	edge/node
YelpZip	2	873,919	2434,392	2.79
PubMed	3	1,9717	44,324	2.25
BlogCatalog	39	10,312	333,983	32.39
Bioinformatics	144	13,682	287,916	21.04

IV. Experiments

We examine the explanation faithfulness, interpretability and scalability of **GE-L**, **GE-G**, and the state-of-the-art baselines, **LIME**, in four domains. More experiments can be found in [4], including sensitivity analysis, and case-study.

A. Datasets

We drew datasets from four applications. First, we adopt the Yelp review networks from [16] for spam detection tasks. We represent reviewers, reviews, and products and their relationships by an MRF. BP can infer the labels (suspicious or normal) of reviews. Second, in collective classification, we construct an MRF for the citation network (PubMed) that contain papers as nodes and undirected edges as paper citation relationships [13]. BP can infer the distributions of paper topics of an unlabeled paper. Third, we represent blogs (BlogCatalog) as nodes and behaviors, including subscription and tagging, as edges [20]. BP infers the preferences of users. Lastly, in biological networks, we adopt the networks analyzed in [21], which denotes nodes as protein-protein pairs and the subordination relations of protein pair as the class. Explaining BP inference is important in all these applications: the MRFs are in general large and cyclic for a user to thoroughly inspect why a review is suspicious, or why a blog is under a specific topic. The statistics of the datasets are shown in Table III.

B. Experimental setting

On Yelp review network, a review has two and only two neighbors (a reviewer posts the review and the product receives the review), while a reviewer and a product can be connected to multiple reviews. On the remaining networks, nodes are connected to other nodes without constraints over the number of neighbors and the type of nodes. We apply the two variants of **GE-L** on Yelp and other networks, respectively. Psychology study shows that human can process

about seven items at a time [12]. To balance the faithfulness and interpretability, both **GE-L** and **GE-G** search trees that are of maximum size of five starting from the target node.

On all networks, we assume homophily relationships between any pair of nodes. For example, a paper is more likely to be on the same topic of the neighboring paper. On Yelp, we set node priors and compatibility matrices according to [16]. On other networks, we assign 0.9 to the diagonal and $\frac{0.1}{c-1}$ to the rest of the compatibility metrics. As for priors, we assign 0.9 to the true class of a labeled node, and $\frac{0.1}{c-1}$ to the remaining classes, where c is the number of classes in data. For unlabeled nodes, we set uniform distribution over classes. We set the ratio of labeled data as 50%. With consideration the size of the large networks, we sample 20% of unlabeled nodes as target nodes on BlogCatalog and Bioinformatics networks.

C. Baselines

Random It ignores outcomes from BP and selects a node in $\partial\tilde{G}^{(t-1)}$ randomly when extending $\tilde{G}^{(t-1)}$. For fair comparison, **Random** searches subgraphs of the same structures as those found by **GE-L** and **GE-G**, respectively.

Embedding It constructs subgraphs with the same size as those found by **GE-G**. However, it utilizes DeepWalk[15] to obtain node embeddings, based on which it selects top similar candidate nodes to explain the target variable.

LIME [17] It is the state-of-the-art black-box explanation method that works for classification models when input are vectors rather than graphs. We randomly select 200 neighbors of each target node in the node feature vector space, with sampling probability weighted by the cosine similarity between the neighbors and the target. A binary/multiclass logistic regression model is then fitted on the sample and used to approximate the decision boundary around the target. It cannot explain nodes without feature vectors.

Comb It aggregates all trees from `Beam[C]` into a single graph as an explanation. This method can aggregate at most k trees and have at most kC variables. Here, we set k=3 and report the performance of the combined subgraph. The performances of the top candidate in `Beam[C]` with k=1 and k=3 are reported as **GE-G** (k=1) and **GE-G** (k=3).

D. Explanation Accuracy

Overall Performance For each method, we run BP on the extracted subgraph $\tilde{G}$ for each target variable X to obtain a new belief $\tilde{b}_X$ (except for **LIME** that does not construct subgraphs). Explanation faithfulness is measured by Eq. (3). In Table IV, we report the mean of the performance metric over all target variables, and the best methods are boldfaced. We also report the average size of explaining subgraphs in the square brackets after individual means.

From Table IV, we can conclude that: 1) **Comb** always constructs the largest subgraph and performs best, due to multiple alternative high-quality explanations from the beam branches. 2) **GE-G** (k=3) is the runner up and better than **GE-G** (k=1), because the search space of **GE-G** (k=3) is larger than **GE-G**(k=1)'s. 3) The performance of **Embedding** is not very good, but still better than **LIME**. **LIME** has the worst performance, as it is not designed for graphs and cannot take the network connections into account.

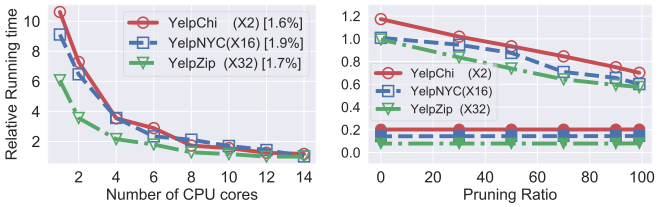

Figure 2. **Left** the computing time reduces as the number of cores increases from 1 to 14. The parentheses enclose the number of hour(s) per unit. The square brackets enclose the running time of **GE-L**. For example, **GE-G** takes about 192 hours using one core on YelpZip, while **GE-L** costs 3 hours. **Right** Effect of pruning: as we increase the pruning rate to 99%, the subgraph explanation faithfulness is not degraded (shown by the 3 lines with solid markers), while the running time is reduced by 1/3.

Spam Detection On Yelp review network, **GE-L** generates chain-like subgraphs. The average subgraph size is around four, though the maximum capacity is five. This is because **GE-L** focuses on local information only and stops early when the prior of the last added node best explains the previously added message. Both **GE-G** (k=1) and **GE-G** (k=3) extend the subgraph to the maximum size to produce a better explanation. Notice that **Random** performs better when imitating **GE-G** (k=1) than when imitating **GE-L**. The reason is that there are only two types of neighboring nodes of the target node and **Random** imitating **GE-G** has a higher chance to include the better neighbor.

Collective Classification In these tasks, **GE-L** constructs star-like subgraphs centered at the target node. Interestingly, compared with the Yelp review network, **Random** imitating **GE-G** generates larger subgraphs but performs worse than **Random** imitating **GE-L**. The reason is that **Random** can add nodes far away from the target node as **GE-G**. However, without the principled guidance in **GE-G**, **Random** can add nodes that are likely in the other classes than the same class of the target node.

E. Scalability

We run **GE-G** (k=1) on the Yelp review network. The scalability of the algorithm is demonstrated in two aspects in Figure 2. First, the search can be parallelized on multiple cores. The running time goes down super-linearly as the number of cores increases from 1 to 14. Second, candidate pruning plays a role in speeding up the search. We use 14 cores and increase the pruning ratio from 0 to 99% and obtain $\times 1.7$ speedup. Importantly, the explanation faithfulness is not affected by the pruning, shown by the three lines at the bottom of the right figure (200 $\times$ mean of $d(b_X, \tilde{b}_X)$).

Table IV
Overall performance: in general **Comb** is the best method that significantly outperforms all other methods on all networks. We use • to indicate whether statistically **GE-L** significantly (pairwise t-test at 5% significance level) outperforms **Random** and whether **Comb** outperforms **GE-G** (k=3), respectively.

Method	Embedding	LIME	Random	GE-L	Random	GE-G (k=1)	GE-G (k=3)	Comb
YelpZip	0.084[5.0]	6.036	0.040[4.2]•	0.025[4.2]	0.010[5.0]	0.0014[5.0]	0.0013[5.0]•	**0.0008**[6.1]
PubMed	0.842[5.0]	0.910	0.718[3.1]	0.577[3.1]	0.893[5.0]	0.188[5.0]	0.185[5.0]•	**0.098**[7.1]
BlogCatalog	7.887[5.0]	-	7.899[4.8]	8.054[4.8]	7.867[5.0]	6.621[5.0]	6.702[5.0]	**6.343**[7.9]
Bioinformatics	2.065[5.0]	-	2.085[4.9]	1.893[4.9]	2.116[5.0]	1.423[5.0]	1.508[5.0]	**1.356**[5.6]

V. Related work

To explain differentiable parametric predictive models, such as deep networks [8] and linear models [11], the gradients of the output with respect to the parameters and input data [18] can signify key factors that explain the output. However, graphical models aim to model long range and more complicated types of interaction among variables. In [17], parametric or non-parametric models are fitted to approximate a more complex model locally, while a parametric model does not deliver good explanations to the inference outcomes on a graphical model. In [9], HMM is trained to approximate an RNN model. However, both HMM and RNN are linear while GraphExp focuses on graphs with more general topology, including cycles.

Explainable bayesian networks were studied in the 1980's and 1990's [19], driven by the needs to verify and communicate the inference outcomes of Bayesian networks in expert systems. More recently, Bayesian networks were formulated as a multi-linear function so that explanations can be facilitated by differentiation [5]. The fundamental difference between GraphExp and these prior works is that we handle MRFs with cycles while they handled Bayesian networks without cycles. The differentiation-based explanation of MRFs in [2] finds a set of important netowrk parameters (potentials) to explain changes in the marginal distribution of a target variable without explaining any inference procedure. GraphExp generates graphical models consisting of prominent variables for reproducing the inference of BP on larger graphs. Interpretable graphical models are also studied under the hood of topic models [3].

VI. Conclusion

We propose a general framework to explain the inference outcome of the target node from a MRF by searching smaller trees. Specifically, we substantiate the framework with **GE-G** and **GE-L** to demonstrate the faithfulness and simplicity of the generated trees in the experiment. Besides, we provide parallelization and a safe pruning strategy to speed up the search. Further work will explore how a model learns to generate the subgraphs with limited domain knowledge.

Acknowledgement

Chao Chen and Sihong Xie are supported by Lehigh young faculty startup. Yifei Liu and Xi Zhang are supported by NSFC (No. 61976026).

References

[1] Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. Diverse M-Best Solutions in Markov Random Fields. In *ECCV*, 2012.

[2] Hei Chan and Adnan Darwiche. Sensitivity Analysis in Markov Networks. In *IJCAI*, 2005.

[3] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-graber, and David M Blei. Reading Tea Leaves: How Humans Interpret Topic Models. In *NIPS*. 2009.

[4] Chao Chen, Yifei Liu, Xi Zhang, and Sihong Xie. Scalable explanation of inferences on large graphs. *arXiv:1908.06482*, 2019.

[5] Adnan Darwiche. A Differential Approach to Inference in Bayesian Networks. *J. ACM*, 2003.

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9:432–441, 2008.

[7] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *JMLR*, 3, March 2003.

[8] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. In *ICML*, 2017.

[9] Viktoriya Krakovna and Finale Doshi-Velez. Increasing the Interpretability of Recurrent Neural Networks Using Hidden Markov Models. In *ICML Workshop on Human Interpretability in Machine Learning*, 2016.

[10] Tania Lombrozo. The structure and function of explanations. *Trends in Cognitive Sciences*, 10(10):464–470, 2006.

[11] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible Models for Classification and Regression. In *KDD*, 2012.

[12] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[13] Galileo Mark Namata, Ben London, and Lise Getoor. Collective Graph Identification. *ACM Trans. Knowl. Discov. Data*, 2016.

[14] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, Dec 1978.

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. ACM, 2014.

[16] Shebuti Rayana and Leman Akoglu. Collective Opinion Spam Detection: Bridging Review Networks and Metadata. KDD, 2015.

[17] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. KDD, 2016.

[18] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR*, 2013.

[19] Henri Jacques Suermondt. *Explanation in Bayesian Belief Networks*. PhD thesis, 1992.

[20] Lei Tang and Huan Liu. Relational learning via latent social dimensions. KDD, 2009.

[21] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.