Explaining the Inferences on Large Graph Flexibly via Subgraph Search

Chao Chen[1,*], **Yifei Liu**[2,+], **Xi Zhang**[3,+], **and Sihong Xie**[4,*]

[*]Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA 18015, USA
[+]Beijing University of Posts and Telecommunications, Beijing 100876, China
[1]chc517@lehigh.edu
[2]liuyifei199512@163.com
[3]zxwinner@gmail.com
[4]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. To close the gap, we formulate the problem of explaining probabilistic inferences as a constrained cross-entropy minimization problem to find simple subgraphs that faithfully approximate the inferences to be explained. We prove that the optimization is NP-hard, while the objective is not monotonic and submodular to guarantee efficient greedy approximation. We propose a general beam search algorithm to find simple trees to enhance the interpretability and diversity in the explanations, with parallelization and a pruning strategy to allow efficient search on large and dense graphs without hurting faithfulness. We demonstrate superior performance on 10 networks from 4 distinct applications, comparing favorably to other explanation methods. Regarding the usability of the explanation, we visualize the explanation in an interface that allows the end-users to explore the diverse search results and find more personalized and sensible explanations.

Introduction

Distilling knowledge from graphs is an important task found ubiquitously in applications, such as fraud detection (1), user interest modeling in social networks (2; 3; 4), and bioinformatics (5; 6; 7). The knowledge helps humans make high-stake decisions, such as whether to investigate a business or account for fraud detection 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 Bayesian networks, Markov Random Fields (MRF) (8), and Conditional Random Fields (CRF) (9), and distill knowledge from the graphs using inferences based on optimization (10; 11) and message passing (12). Different from predictive models on i.i.d. vectors, graphical models capture the dependencies among random variables and carry more insights for decision making. However, the inferences involve iterative and recursive computations, making the inference outcomes cognitively difficult to understand, verify, and ratify, locking away more applications of graphical models (EU law requires algorithmic transparency (13)) and more accurate models through debugging (14). We focus on MRFs and belief propagation (BP) inferences (15) that compute marginal distributions, aiming to make the inference outcomes more interpretable and thus cognitively easier for humans. Fig. 1 depicts the problem definition and the proposed solution.

Several challenges are due. First, simple but faithful explanations are desired (16) but have not been defined for inferences on MRFs. Prior work (17; 18) approximates a high-dimensional Gaussian through a sparse covariance matrix, which does not explain belief propagation. To explain the marginal distribution on MRFs, using sensitivity analysis, the authors in (19; 20) proposed to find influential parameters inherent in the model but not in any particular inference algorithms and computations. Explanation of parametric linear models and deep networks using surrogate models, differentiation, and feature selection (21; 22; 23; 24; 25) cannot be applied to graphical model inferences, although our proposal can explain inferences on deep graphical models (26). Explainable RNNs (27) handles linear chains but not general MRFs. In terms of usability, previous works have studied how to visualize explanations of other models and utilize the explanations in the end tasks such as model debugging (28). It is less know what's the best *graph* explanation complexity and faithfulness trade-off for the end-users, how to effectively communicate the probabilistic explanations, and how to utilize the explanations.

Second, algorithmically, an MRF can be large, densely connected, and cyclic, while simple and faithful explanations need to be found efficiently. BP computes each message using other messages iteratively and recursively until convergence (Eq.

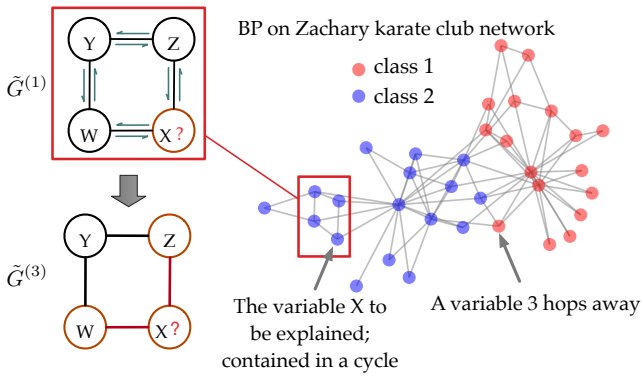

Figure 1. A cyclic graphical model G for the Zachary karate club network, with BP inference outcomes shown in two colors. We focus on explaining how BP calculates the belief on X, highlighted in the subgraph $\tilde{G}^{(1)}$. Due to the cycles and long-range dependencies on G, a complete explanation is recursive and long. With messages, beliefs, and priors, GraphExp extracts a limited-size *tree* $\tilde{G}^{(3)}$, on which X has a marginal similar to that on G.

(2)). As a result, a message is a function of other messages and a complete explanation requires the entire history of message computations, possibly from multiple iterations. Furthermore, on cyclic graphs, a converged message can be defined recursively by itself, generating explanations such as "a user is fraudulent because it is fraudulent". A desirable explanation should be without recursion, but cutting out the cycles may result in a different model and affect faithfulness.

We propose a new approach called "GraphExp" to address the above challenges. Given the full graphical model G and any target variable X (the "explanandum") on G, GraphExp finds an "explanans" (or the "explaining") 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 (29). Starting from the graph $\tilde{G}^{(1)}$ consisting of X only, GraphExp greedily includes the next best variable into the previous subgraph so that the enlarged subgraph has the lowest loss. Theoretically we prove that: **(1)** an exhaustive search for the optimal $\tilde{G}$ with highest faithfulness (lowest loss) 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 1.1); **(2)** GraphExp only generates acyclic graphs that are more explainable (Theorem 2.1).

There can exist multiple sensible explanations for the same inference outcome (30; 31) and an end-user can find the one that best fits her mental model. We equip GraphExp with beam search (32) 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 trivially parallelizable, we further propose a safe pruning strategy that retains the desirable candidates while cutting the search space down significantly (Fig. 4). Regarding usability, GraphExp does not commit to a single explanation but allows the users to select one or multiple most sensible explanations for further investigation (Section 3.7). We highlight the contributions as follows:

- We define the problem of explaining belief propagation to end-users and study the communication and utility of the explanations.

- We propose an optimization problem for finding simple, faithful, and diverse explanations. We prove the hardness of the problem and then propose GraphExp as a greedy approximation algorithm to solve the problem. We analyze the time complexity and the properties of the output graphs, Both parallel search and the proposed pruning strategy deliver at least linear speed-up on large datasets.

- Empirically, on 10 networks with up to millions of nodes and edges from 4 domains, GraphExp explains BP faithfully and significantly outperforms variants of GraphExp and other explanation methods not designed for graphs. We propose visualization that allows flexible, intuitive of the found explanations. To demonstrate the utility of the found explanations, we identify a security issue in Yelp spam detection using the found subgraphs.

1 Problem definition

Notation definitions are summarized in Table 2. Given a set of n random variables $V = \{X_1, \ldots, X_n\}$, each taking values in

Table 1. Comparing GraphExp with prior explanation methods: FS (Feature Selection), LIME, GSparse (graph sparsification), and GDiff (graph differentiation). (∗: "surely yes"; ○: "partially"; emptiness: "no").

	FS (23)	LIME (21)	GSparse (18)	GDiff (19)	GraphExp
Cycles handling			∗	∗	∗
Completeness			○	∗	∗
Interpretability	○	∗	○	○	∗
Diversity	○	○			∗
Scalability		○		○	∗
Flexibility		○		○	∗

Table 2. Notation Definitions

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

$\{1,\dots,c\}$ where c is the number of classes, an MRF G factorizes the joint distribution $P(V)$ as

$$P(V)=\frac{1}{Z}\prod_{i\in[n]}\phi_i(X_i)\prod_{i,j\in[n]}\psi_{ij}(X_i,X_j), \tag{1}$$

where Z normalizes the product to a probability distribution, ϕ_i is the prior distribution 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 and capture the dependencies between variables. The factorization can be represented by a graph consisting of $X_1,\dots,X_n$ as nodes and edges (X_i,X_j), as shown in Fig. 1. BP inference computes the marginal distributions (beliefs) $b_X, X\in V$, based on which human decisions can be made. The inference computes messages $m_{i\to j}(x_j)$ from X_i to X_j:

$$\frac{1}{Z_j}\sum_{x_i}\left[\psi_{ij}(x_i,x_j)\phi_i(x_i)\prod_{k\in\mathcal{N}(X_i)\setminus\{j\}}m_{k\to i}(x_i)\right], \tag{2}$$

where Z_j is a normalization factor so that $m_{i\to j}$ is a probability distribution of x_j. The messages in both directions on all edges $(i,j)\in E$ are updated until convergence (guaranteed when G is acyclic (33)). The marginal b_X is the belief

$$b(X)\propto\phi(X)\prod_{Y\in\mathcal{N}(X)}m_{Y\to X}(X), \tag{3}$$

where $\mathcal{N}(X)$ 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 and to completely explain how b_X is computed, one has to trace down each message in Eq. (3) 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 (12; 34). A complete explanation of BP can keep track of all these computations on a call graph (35). However, the graph easily becomes too large to be intuitive 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 follow 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 $\tilde{G}\subset G$ with $X\in\tilde{G}$ and 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*

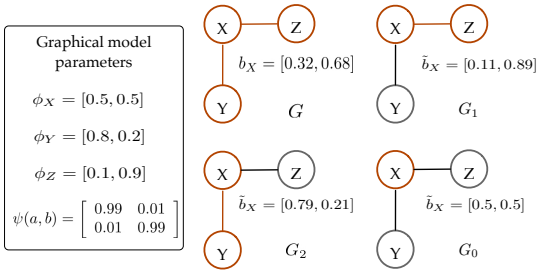

Figure 2. A graphical model on which submodularity and monotonicity of the objective function in Eq. (4) do not hold. The goal is to find the best subgraph from $\{G_0, G_1, G_2\}$ that best approximates the belief of X inferred on G. Edges and nodes in red are those included in the individual subgraphs.

following

$$\begin{aligned} &\min_{\tilde{G}} \; d(b_X, \tilde{b}_X) = \mathrm{KL}(b_X || \tilde{b}_X) + \mathrm{KL}(\tilde{b}_X || b_X) \\ &\text{s.t.} \;\; \tilde{G} \subset G, \quad |\tilde{G}| \leq C, \quad X \in \tilde{G}, \quad \tilde{G} \text{ acyclic}. \end{aligned} \tag{4}$$

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

$$\mathrm{KL}(P||Q) = \sum_{x=1}^{c} P(x) \log[P(x)/Q(x)]. \tag{5}$$

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 $\tilde{b}_X$ (29). 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 less than C. Since a graphical model encodes a joint distribution of a set of variables, the above problem is equivalent to searching a joint distribution of a smaller number of variables with fewer variable dependencies, so that the two joint distributions lead to similar marginals of the variable X to be explained.

If the negation of the above objective function is submodular and monotonically increasing, then a greedy algorithm that iteratively builds $\tilde{G}$ by adding one variable at a time to increase the negation of objective function (namely, to decrease $d(b_X, \tilde{b}_X)$) can generate a solution whose value is within $(1-1/e)$ of the optimum (36).

Definition 1.1 (Submodularity)**.** *Let $\mathscr{V}$ be a set and $2^{\mathscr{V}}$ be the power set of $\mathscr{V}$. A set function $f : 2^{\mathscr{V}} \to \mathbb{R}$ is submodular if $\forall \mathscr{A} \subset \mathscr{A}' \subset \mathscr{V}$ and any $Y \notin \mathscr{A}'$, $f(\mathscr{A} \cup \{Y\}) - f(\mathscr{A}) \geq f(\mathscr{A}' \cup \{Y\}) - f(\mathscr{A}')$.*

Definition 1.2. *A set function $f : 2^{\mathscr{V}} \to \mathbb{R}$ is monotonically increasing if $\forall \mathscr{A} \subset \mathscr{A}' \subset \mathscr{V}$, implies $f(\mathscr{A}) \leq f(\mathscr{A}')$.*

However, we prove that both properties do not hold on $d(b_X, \tilde{b}_X)$, which cannot be efficiently approximated well.

Theorem 1.1. *The objective function in Eq. (4) is not submodular nor monotonically increasing.*

Proof. We find a counterexample that violates submodularity and monotonicity. As shown in Fig. 2, the full graph G has 3 variables X, Y and Z, with X connected to Y and Z, respectively. The priors are $\phi(X) = [0.5, 0.5]$, $\phi(Y) = [0.8, 0.2]$, and $\phi(Z) = [0.1, 0.9]$, and both edges have the same homophily-encouraging potential $\psi_{ij}(a,b) = 0.99$ if $a = b$ and 0.01 otherwise.

Let subgraphs $G_0 = \{X\}$, $G_1 = \{X, Z\}$ and $G_2 = \{X, Y\}$ be as shown in the figure. One can run BP on $G_i, i = 0, 1, 2$ to find $\tilde{b}_X$ and that $(-d_{G_2}(b_X, \tilde{b}_X)) - (-d_{G_0}(b_X, \tilde{b}_X)) < (-d_G(b_X, \tilde{b}_X)) - (-d_{G_1}(b_X, \tilde{b}_X))$, with the subscriptions G_i indicating on which subgraph is $\tilde{b}_X$ computed. However, $G_0 \subset G_1$ and the gain through adding Y to G_1 is greater than that adding Y to G_0. On the same example, we can see that adding Y to G_0 can increase the objective. □

2 Methodologies

The optimization problem Eq. (4) 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 (23; 37), similar to exhaustive feature selection. Greedy algorithms are a common solution to approximately solve NP-hard problems. Since finding multiple

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.

alternative sensible explanations is one of our goals, we adopt beam search in the greedy search (32), maintaining in a beam several top candidates ranked by faithfulness and succinctness throughout the search.

A general greedy beam search framework is presented in Algorithm 1. The algorithm finds multiple explaining subgraphs $\tilde{G}^{(C)}$ of size C in $C-1$ iterations, where C is the maximum subgraph size (which roughly equals to human working memory capacity (38), or the number of cognitive chunks (39)). 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 the desired subgraphs are found and right before the algorithm exits, BP will be run again on $\tilde{G}$ to compute $\tilde{b}_X$ so that we can use the converged messages on $\tilde{G}$ to explain to an end-user how b_X is approximated on $\tilde{G}$. Since $\tilde{G}$ is small and contains no cycle, the explanation is significantly simpler than the original computations on G. We substantiate the general framework in the following two sections, with two alternative ways to rank candidate extensions of the subgraphs during the beam search. Before we discuss the two concrete algorithms, a theoretical aspect related to interpretability is characterized below.

Theorem 2.1. *The output $\tilde{G}$ from Algorithm 1 is a tree.*

Proof. We prove this by induction. Base case: $\tilde{G}^{(0)}$ is a single node so it is a tree. Inductive step: Assume that $\tilde{G}^{(t)}$ is a tree. By adding one more variable and edge (X^* and e^*) to $\tilde{G}^{(t)}$, there is no cycle created, since we are only attaching X^* to $\tilde{G}^{(t)}$ through a single edge e^*. □

2.1 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 the objective Eq. (4). 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 $\tilde{b}_X$, the marginal of X, and the distance $d(b_X,\tilde{b}_X)$ is calculated as the quality of the candidate $\tilde{G}^{(t)}$. After exhausting all possible candidates, **GE-G** adds the k candidates with the least distances to `Beam[t]`.

One search step in Fig. 3 demonstrates the above ideas. The search attempts to extend the middle $\tilde{G}^{(2)}$ in `Beam[2]` by adding a new variable from $\partial\tilde{G}^{(2)}=\{Y,Z\}$ to the subgraph, so the new belief $\tilde{b}_X$ on the larger subgraph is closer to b_X. In this example, **GE-G** keeps the top 3 extensions of $\tilde{G}^{(2)}$ (beam size is 3), but only two options are legitimate and are the only two candidates included in `Beam[3]`. The middle subgraph consisting of $\{W,X\}$ is generated by the algorithm **GE-L** to be discussed later (see Section 2.3). When the search extends the bottom right subgraph $\tilde{G}^{(3)}$, variable $Y\in\partial\tilde{G}^{(3)}$ can be connected to $\tilde{G}^{(3)}$ in two ways, through edges $Y\to Z$ and $Y\to W$, but both GraphExp variants include only one link to avoid cycles in the exlaining subgraphs.

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 (which is guaranteed by Theorem 2.1).

One of the advantages of **GE-G** is that the objective function in the optimization problem Eq. (4) 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

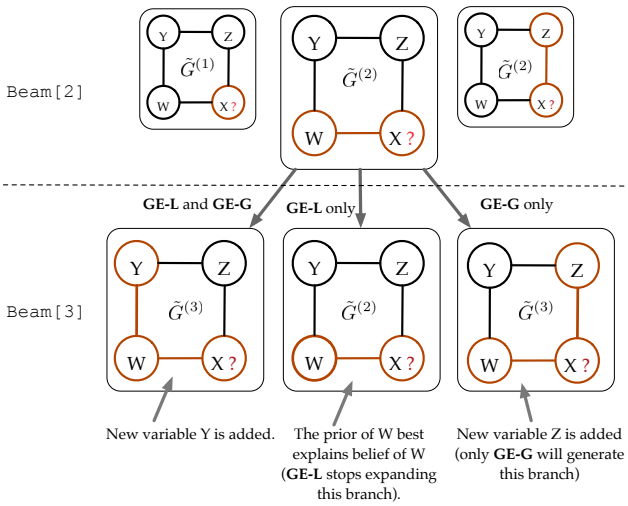

Figure 3. Explaining BP on a graphical model with four variables (X, Y, Z, and W), four edges, and a cycle. We show one step of beam search, extending a candidate $\tilde{G}^{(2)}$ in `Beam[2]` to three candidates of $\tilde{G}^{(3)}$ in `Beam[3]` (with beam size 3). The rightmost candidate in `Beam[2]` is also extended but the extensions are not shown to avoid clutters. After all extensions are generated, only the top 3 are kept in `Beam[3]`. The edges and variables in red are included in the explaining subgraphs. Note that **GE-G** and **GE-L** have different behaviors when dealing with the extensions.

$t = 2, \ldots, C$, 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]` in the iteration 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.

2.2 Speeding up GE-G on large graphs

Graphical models in real-world applications are usually gigantic containing tens or hundreds of thousands of nodes. GraphExp can take a long time to finish on such large graphs, especially when the graph diameter is small. Slowness can harm the usability of GraphExp in applications requiring interpretability, for example, when a user wants to inspect multiple explanations of BP for counterfactual analysis, or statistics of the errors needs to be computed over explanations of many nodes (14). 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. Particularly for **GE-G**, during the BP inference over each candidate $\tilde{G}^{(t)}$, there are existing parallel algorithms that compute the messages (40) asynchronously. As the subgraphs $\tilde{G}^{(t)}$ are bounded by the human cognitive capacity and are small, a parallel inference can be an overkill. We evaluate the reduction in search time using the first level of parallelism (Section 3.5).

Pruning candidate variables In Algorithm 1, all candidates $e \in C = (G \setminus \partial\tilde{G}^{(t)}, \partial\tilde{G}^{(t)})$ have to be evaluated to select (X^*, e^*) and we have to run BP as many times as $|C|$. When the cut C is large, this can be costly. As we aim at explaining how BP infers the marginal of the target Y, adding any variable that has a distribution that deviates much from the distribution of Y is not helpful but confusing. Considering that a subgraph $\tilde{G}^{(t-1)}$ has $|C|$ candidates at step-t, we run BP on these $|C|$ candidates and abandon the bottom $(100-p)$ percent of them based on Eq. 4 in the following steps.

2.3 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 get a quick feeling of how the inferences are made or 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** that can generate more faithful and detailed explanations (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 iteration t, the search adds

an additional edge between $\tilde{G}^{(t-1)}$ and $\partial\tilde{G}^{(t-1)}$ that best explains a message or a belief in $\tilde{G}^{(t-1)}$, which can be computed using information that is not currently in $\tilde{G}^{(t-1)}$. There are two cases.

- For a message $m_{Y\to W}$ on an edge (Y,W) already included in $\tilde{G}^{(t-1)}$, the search attempts to find a message $m_{Z\to Y}$, where $Z\in\partial\tilde{G}^{(t-1)}$, so that the message $m_{Z\to Y}$ contributes most to the message $m_{Y\to W}$. We use the distance d defined in Eq. (4) to measure the contribution of $m_{Z\to Y}$ to $m_{Y\to W}$: the smaller the distance, the more similar two messages are and thus more contribution from $m_{Z\to Y}$ to $m_{Y\to W}$.
- 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 both cases, we define the **end points** of $\tilde{G}^{(t-1)}$ as those nodes that are in $\tilde{G}^{(t-1)}$ but can be connected to those in $\partial\tilde{G}^{(t-1)}$. In the example in Fig. 1, Y and X are end points of the subgraph $\tilde{G}^{(2)}$ in the middle. In **GE-L**, if the prior of an end-point best explains the message emitting from the end-point (e.g., $m_{Y\to X}$ in Fig. 1) or belief of the end-point (b_X in the same figure), the prior is added to the subgraph and no extension can be made to the end point: the prior is a parameter of the graphical model and not computed by BP, and no other BP distribution can further explain the prior. The search of **GE-L** will stop at this branch, although the other branches in the beam can further be extended. Using the same example in Fig. 1, the prior ϕ_W best explains $m_{W\to X}$) and this branch is considered finished requiring no more extension.

Analysis of GE-L We analyze what the above search is optimizing. We unify the two cases where the search explains how a message and belief are computed. Assuming that the potential function ψ_{ij} for all edges (X_i,X_j) are identity matrices, which is the case when a graph exhibits strong homophily. Then the message going from X_i to X_j in Eq. (2) is proportional to

$$\phi_i(x_i)\prod_{k\in\mathcal{N}(X_i)\setminus\{j\}} m_{k\to i}(x_i), \tag{6}$$

which is in the same form of Eq. (3) when a belief is computed. Therefore, both messages and beliefs can be written in the form of $P(x)=\prod_{\ell=1}^{L}p_\ell(x)$, where L is the number of factors in the product. Let $Q=\prod_{m=1}^{M}q_m(x)$ be a distribution that explains P with M factors and with X being an end-point to be explained. If $Q=P$, then the distance is 0 but many edges (factors) are included in $\tilde{G}$. Starting from a uniform distribution, the search each time finds an approximating distribution $Q(x)$ by including an additional factor in P to minimize $d(P,Q)$ over all distributions P representing the so-far included messages and beliefs computed by BP on G, and over all factors (messages or priors) that have not been included but are contributing to P. Therefore, the algorithm does not directly attempt to optimize the objective $d(b_X,\tilde{b}_X)$ for the target X, but does so locally: it keeps adding more factors to best explain one of the next end-points, which can already explain the target variable to some extent.

Variants of GE-L To further speed up **GE-L** (see Fig. 4), especially on graphs that a user has prior knowledge about the topology of the explaining subgraph, 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 (16), 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 the review and remaining networks, respectively.

3 Experiments

In this section, we examine the explanation faithfulness and interpretability of **GE-L**, **GE-G**, and the state-of-the-art baselines, including **LIME**, in ten different networks from four domains (spam detection, citation analysis, social networks, bioinformatics). We also evaluate the scalability and sensitivity analysis of these methods. Moreover, we conduct user study to demostrate the usability of **GE-G**.

3.1 Datasets

We drew datasets from four applications. First, we adopt the same three Yelp review networks (YelpChi, YelpNYC, and YelpZip) from (1) for spam detection tasks. We represent reviewers, reviews, and products and their relationships (reviewer-review and review-product connections) by an MRF. BP can infer the labels (suspicious or normal) of reviews on the MRF given no labeled data but just prior suspiciousness probability computed from meta-data (1). Second, in collective classification, we construct an MRF for each one of three citation networks (Cora, Citeseer, and PubMed) that contain papers as nodes and undirected edges as paper citation relationships (4). As a transductive learning algorithm, BP can infer the distributions of paper topics of an unlabeled paper, given labeled nodes in the same network. Third, we represent blogs (BlogCatalog), videos (Youtube), and users (Flickr) as nodes and behaviors including subscription and tagging as edges (2). BP infers the preferences of users. The

Table 3. Ten networks from four application domains.

Datasets	Classes	Nodes	Edges	edge/node
YelpChi	2	105,659	269,580	2.55
YelpNYC	2	520,200	1436,208	2.76
YelpZip	2	873,919	2434,392	2.79
Cora	7	2,708	10,556	3.90
Citeseer	6	3,321	9,196	2.78
PubMed	3	1,9717	44,324	2.25
Youtube	47	1,138,499	2,990,443	2.63
BlogCatalog	39	10,312	333,983	32.39
Flickr	195	80,513	5,899,882	73.28
Bioinformatics	144	13,682	287,916	21.04

goal is to classify social network nodes into multiple classes. Lastly, in biological networks, we adopt the networks analyzed in (7) 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 paper belongs to a certain area, or why a review is suspicious, or why a blog is under a specific topic, or why two proteins connect to each other. It is much easier for the user to verify the inference outcome on much smaller explaining graphs. The statistics of the datasets are shown in Table 3.

3.2 Experimental setting

On Yelp review networks, 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 of 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 beings can process about seven items at a time (38). To balance the faithfulness and interpretability, both **GE-L** and **GE-G** search subgraphs that are of maximum size five starting from the target node. In the demon, we explore larger explaining subgraphs to allow users to select the smallest subgraph that makes the most sense.

On all ten 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, and two nodes are more likely to be suspicious or normal at the same time. On Yelp, we set node priors and compatibility matrices according to (1). 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. On Yelp, there is no labeled node, and on Youtube network, 5% are labeled nodes. For the remaining networks, we set the ratio of labeled data as 50%. With consideration the size of the large networks, we sample 1% from the unlabeled nodes as target nodes on Youtube and Flickr datasets, and sample 20% of unlabeled nodes as target nodes on BlogCatalog and Bioinformatics networks.

Table 4. 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
Chi	0.058[5.0]	5.300	0.053[3.9]•	0.036[3.9]	0.022[5.0]	0.0012[5.0]	0.0012[5.0]•	**0.0006**[6.5]
NYC	0.084[5.0]	5.955	0.043[4.1]•	0.028[4.1]	0.017[5.0]	0.0012[5.0]	0.0011[5.0]•	**0.0006**[6.2]
Zip	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]
Cora	0.527[4.8]	1.321	0.362[3.6]•	0.181[3.6]	0.594[4.8]	0.137[4.8]	0.132[4.9]•	**0.084**[6.4]
Citeseer	0.305[4.4]	1.221	0.243[2.9]•	0.108[2.9]	0.340[4.4]	0.077[4.4]	0.075[4.4]•	**0.048**[5.7]
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]
Youtube	0.340[5.0]	-	0.376[2.8]	0.321[2.8]	0.343[5.0]	0.263[5.0]	0.264[5.0]	**0.225**[6.7]
Flickr	5.903[5.0]	-	6.259[4.7]	6.232[4.7]	6.018[5.0]	4.654[5.0]	4.652[5.0]	**4.111**[7.4]
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]

3.3 Baselines

Random It ignores the messages computed by BP and selects a node in $\partial\tilde{G}^{(t-1)}$ randomly when extending $\tilde{G}^{(t-1)}$. To compare fairly, **Random** searches the subgraph of the same structure 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(41) to obtain node embeddings, based on which top candidate nodes similar to the target are included to explain the target variable.

LIME (21) It is the state-of-the-art black-box explanation method that works for classification models when input data 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. The feature vector is defined either as (1) (on Yelp review networks) or bag-of-words (on the citation networks). A binary/multiclass logistic regression model is then fitted on the sample and used to approximate the decision boundary around the target. **LIME** is less efficient than the subgraph search-based approaches since a new classification model has to be fitted for each target variable. **LIME** explains 30% of all review nodes, randomly sampled on Yelp review datasets, and explain all unlabeled variables on the citation networks. It cannot explain nodes in the remaining four networks due to the lack of feature vectors, which is one of the drawbacks of **LIME**.

Comb It aggregates all candidate subgraphs from `Beam[C]` into a single graph as an explanation. This method can aggregate at most k (beam size) subgraphs 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 under **GE-G** (k=1) and **GE-G** (k=3).

3.4 Explanation Accuracy

Overall Performance Quantitative evaluation metrics 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 the objective function in Eq. (4). In Table 4, we report the mean of the performance metric overall 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 4, 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** in all cases. **LIME** has the worst performance, as it is not designed to explain BP and cannot take the network connections into account. 4) Faithfulness is positively correlated to subgraph size.

Spam detection explanations On Yelp review networks, **GE-L** generates chain-like subgraphs. The average subgraph size is around four, even 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** versions 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 and also generates larger subgraphs.

Collective classification tasks In these tasks, **GE-L** constructs star-like subgraphs centered at the target node. On Cora and Citeseer, the performance of **GE-L** is closer to (but still inferior to) **GE-G** with both beam sizes, compared to their performance difference on Yelp. This is because the Cora and Citeseer networks consist of many small connected components, most of which contain less than five nodes, essentially capping **GE-G**'s ability to add more explaining nodes to further bring up the faithfulness (evidenced by the average size of subgraphs found by **GE-G** smaller than 5). Compared with Yelp review networks, interestingly, **Random** imitating **GE-G** generates larger subgraphs but performs worse than **Random** imitating **GE-L**. The reason is that **GE-G** can add nodes far away from the target node and the random imitation will do the same. However, without the principled guidance in **GE-G**, **Random** can add nodes those are likely in the other classes than the class of the target node.

3.5 Scalability

We run **GE-G** (k=1) on Yelp review networks which have the most nodes to be explained. The scalability of the algorithm is demonstrated in two aspects in Fig. 4. First, the search can be done in parallel on multiple cores. As we increase the number of 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 about $\times 1.7$ speedup at most. 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 objective).

3.6 Sensitivity analysis

There are two hyperparameters for the subgraph search and we study the sensitivity of the explanation faithfulness with respect to these two parameters. First, when there are labeled nodes on G, the explanation subgraphs may include a labeled node in the

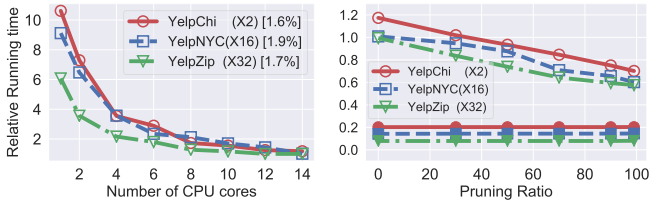

Figure 4. **Left** the computing time reduces superlinearly as the number of cores increases from 1 to 14. The parentheses enclose the number of hour(s) per unit. The square brackets enclose the running time of **GE-L**. For example, **GE-G** takes about 192 hours using one core on YelpZip, while **GE-L** 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 horizontal lines with solid markers), while the running time is reduced by 1/3.

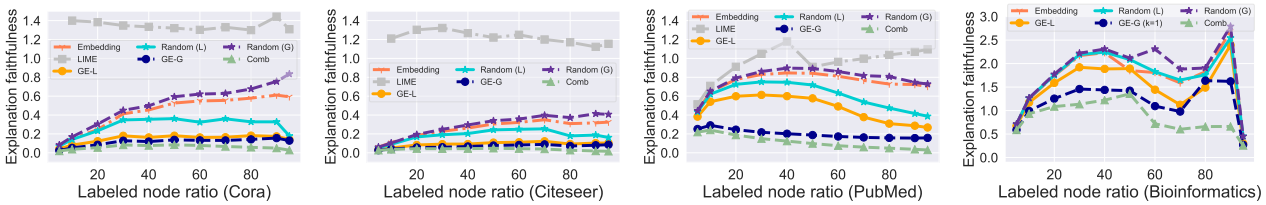

Figure 5. Explanation faithfulness (the smaller the better) on citation networks while the ratio of labeled nodes increases. **Random (L)** is the random baseline that imitate **GE-L**. Similar for **Random (G)**.

explanation, and the more labeled nodes, the more likely the explanation subgraphs will include these nodes. The question here is whether including such labeled nodes will improve explanation faithfulness, and does our method require a large number of labeled nodes. To answer the questions, on the four networks (Cora, Citeseer, PubMed, and Bioinformatics), we vary the ratio of labeled nodes in the ranges of {5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 95%}. The explanation performances are shown in Fig. 5. Overall different ratios on all four networks, **Comb** is the best and **GE-G** is runner-up. On the Cora and Citeseer networks, the performances of all methods are not very sensitive to the ratio of labeled nodes. On the PubMed network, there are improvements in most baselines except **LIME** becomes worse. One can also conclude that **LIME**, designed for parametric models including deep models, is outperformed by methods designed specifically for MRF. On the Bioinformatics, **LIME** is not available and **GE-G** and **Comb** have better performance.

Second, subgraphs containing more nodes perform better since they have more contexts of the target. To evaluate the sensitivity of faithfulness with respect to the subgraph size, we downsize the subgraphs found by other methods (**embedding**, **GE-G** (k=1), **GE-G** (k=3), **Random** imitating **GE-L** and **GE-G**) to the same size of those found by **GE-L**, which may stop growing subgraphs before reaching the cap. The results obtained on four networks are shown in Fig. 6. On Yelp, **GE-L**, **GE-G**, and **Random** perform the same when the size is two since, with this size, the imitating **Random** has to adopt the same subgraph topology and node type as the subgraph found by **GE-L** and **GE-G**. As we downsize the best subgraphs found by **GE-G** ($k = 3$), a better subgraph with a large size may not be optimal when trimmed to size two, lacking the optimal substructures that facilitate dynamic programming variable can alter the target belief, leading to more insight into and trust on the BP.

3.7 Explanation visualization and utility

Explanation for Ratification One of the goals of explanations is to help end-users understand why the probabilistic beliefs are appropriate given the MRF, so that they can develop confidence in the beliefs and BP. This process is called "ratification" (29). To attain this goal, Fig. 7 displays multiple explanations (trees) generated by **Comb** on the Karate Club network, with increasing size in Fig. 7 along with the faithfulness measured by the distance from Eq. (4). One can see that the distance decreases exponentially fast as more nodes are added to the explanation. On each explaining subgraph (tree), we display beliefs found by BP on G and $\tilde{G}$ so that the user is aware of the gap. Since insight is personal (29), the interface is endowed with flexibility for a user to navigate through alternatives, and use multiple metrics (distributional distance, subgraph size and topology) to select the more sensible ones. This design also allows the user to see how adding or removing an variable can alter the target belief, leading to more insight into and trust on the BP.

Explanation for Verification MRF designers can use the explanations for verification of their design, including network parameters and topology. We demonstrate how to use the generated subgraphs to identify a security issue in spam detection. Specifically, we desire that a spam review should be detected with equal probability by the same spam detector, regardless of

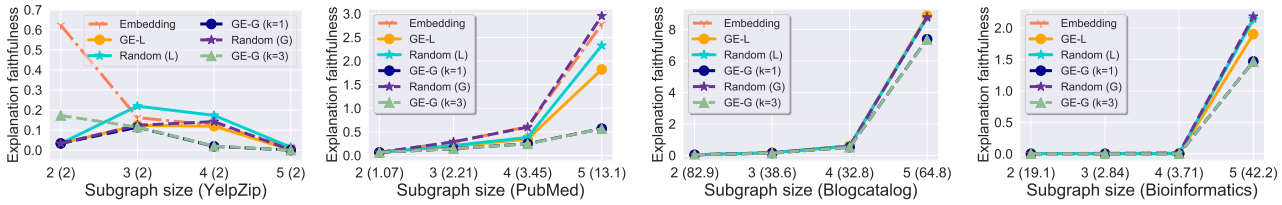

Figure 6. Relationship between size of subgraph and faithfulness

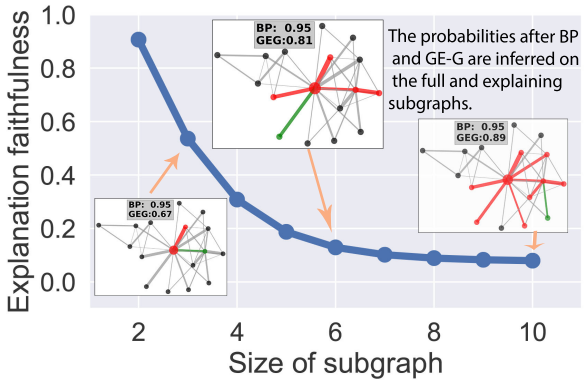

Figure 7. Larger red nodes are the nodes explained. Other red nodes are the explaining nodes. The green nodes are the newly added ones.

how many reviews their authors have posted. On the one hand, it has been found that well-camouflaged elite spammer accounts can deliver better rating promotion (42), a fact that has been exploited by dedicated spammers (43). On the other hand, a spam detector can be accurate in detecting fake reviews for the wrong reason, such as the prolificacy of the reviewers (31).

We gather all reviews from the YelpChi dataset, and from the generated subgraph explanations, create four features, including whether a review is connected to its reviewer and target product, and the degree of its two potential neighbors. We then train a logistic regression model on these features to predict the probability of a review being a false positive (4,501), a false negative (6,681), and mis-classified (11,182), using predictions based on SpEagle (1). The point is that the above mistakes should not depend on the prolificacy of the connected reviewers and products. However, we found that a sizable subset of false negatives (1,153 out of 6,681) are due to the influence from the review node only based on our explanations. Moreover, the logistic model shows that the influence of the degree of the neighboring reivewer contributes the most to the probability of FN. This is a serious security issue: a spam from an prolific reviewers is more likely to be missed by SpEagle.

4 Related work

To explain differentiable parametric predictive models, such as deep networks (22) and linear models (24; 44), the gradients of the output with respect to the parameters and input data (45) can signify key factors that explain the output. However, graphical models aim to model long range and more complicated types of interaction among variables. If a model is too large or complex to be explained, an approximating model can provide certain amount of transparency to the full model. In (21; 46), parametric or non-parametric models are fitted to approximate a more complex model locally. The idea of approximation is similar to that in GraphExp, with different approximation loss functions. We have seen in the experiments that a parametric model won't deliver good explanations to the inference outcomes on a graphical model. In (47), HMM is trained to approximate an RNN model, resembling the idea of using a simpler grapical model to approximate a more complex graphical 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 (29; 48), driven by the needs to verify and communicate the inference outcomes of Bayesian networks in expert systems. It has long been recognized that human users are less likely to adopt expert systems without interpretable and probable explanations (49). More recently, Bayesian networks were formulated as a multi-linear function so that explanations can be facilitated by differentiation (20). 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 (19) 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 a BP procedure on a larger graph. Interpretable graphical

models are also studied under the hood of topic models (50), where the focus is to communicate the meaning of the inference outcomes through measures or prototypes (which words belong to a topic), rather than explaining how the outcomes are arrived at.

References

1. Rayana, S. & Akoglu, L. Collective Opinion Spam Detection: Bridging Review Networks and Metadata. KDD (2015).
2. Tang, L. & Liu, H. Relational learning via latent social dimensions. KDD (2009).
3. Tang, W., Zhuang, H. & Tang, J. Learning to infer social ties in large networks. ECML PKDD (2011).
4. Namata, G. M., London, B. & Getoor, L. Collective Graph Identification. *ACM Trans. Knowl. Discov. Data* (2016).
5. Li, M.-H., Lin, L., Wang, X.-L. & Liu, T. Protein protein interaction site prediction based on conditional random fields. *Bioinformatics* (2007).
6. Huang, Q., Wu, L.-Y. & Zhang, X.-S. An efficient network querying method based on conditional random fields. *Bioinformatics* (2011).
7. Zitnik, M. & Leskovec, J. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* **33**, i190–i198 (2017).
8. Kinderman, R. & Snell, S. *Markov random fields and their applications* (American mathematical society, 1980).
9. Sutton, C. & McCallum, A. An Introduction to Conditional Random Fields. *Foundations Trends Mach. Learn.* (2011).
10. Yanover, C., Meltzer, T. & Weiss, Y. Linear Programming Relaxations and Belief Propagation – An Empirical Study. *JMLR* (2006).
11. Hazan, T. & Shashua, A. Norm-Product Belief Propagation: Primal-Dual Message-Passing for Approximate Inference. *IEEE Transactions on Inf. Theory* (2010).
12. Jo, S., Yoo, J. & Kang, U. Fast and scalable distributed loopy belief propagation on real-world graphs. WSDM (2018).
13. Goodman, B. & Flaxman, S. European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI Mag.* **38**, 50–57 (2017).
14. Wachter, S., Mittelstadt, B. D. & Russell, C. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *CoRR* **abs/1711.00399** (2017).
15. Koller, D. & Friedman, N. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning* (The MIT Press, 2009).
16. Lombrozo, T. The structure and function of explanations. *Trends Cogn. Sci.* **10**, 464–470 (2006).
17. Honorio, J., Samaras, D., Rish, I. & Cecchi, G. Variable Selection for Gaussian Graphical Models. In *AISTAT* (2012).
18. Friedman, J., Hastie, T. & Tibshirani, R. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* **9**, 432–441 (2008).
19. Chan, H. & Darwiche, A. Sensitivity Analysis in Markov Networks. In *IJCAI* (2005).
20. Darwiche, A. A Differential Approach to Inference in Bayesian Networks. *J. ACM* (2003).
21. Ribeiro, M. T., Singh, S. & Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. KDD (2016).
22. Koh, P. W. & Liang, P. Understanding Black-box Predictions via Influence Functions. In *ICML* (2017).
23. Guyon, I. & Elisseeff, A. An introduction to variable and feature selection. *JMLR* **3** (2003).
24. Lou, Y., Caruana, R. & Gehrke, J. Intelligible Models for Classification and Regression. In *KDD* (2012).

25. Caruana, R. *et al.* Case-based explanation of non-case-based learning methods. *AMIA* (1999).

26. Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P. & Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, 2946–2954 (2016).

27. Lei, T., Barzilay, R. & Jaakkola, T. Rationalizing neural predictions. In *EMNLP* (2016).

28. Stumpf, S. *et al.* Integrating rich user feedback into intelligent user interfaces. IUI (2008).

29. Suermondt, H. J. *Explanation in Bayesian Belief Networks*. Ph.D. thesis (1992).

30. Russell, C. Efficient Search for Diverse Coherent Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT (2019).

31. Ross, A. S., Hughes, M. C. & Doshi-Velez, F. Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations. In *IJCAI* (2017).

32. Batra, D., Yadollahpour, P., Guzman-Rivera, A. & Shakhnarovich, G. Diverse M-Best Solutions in Markov Random Fields. In *ECCV* (2012).

33. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988).

34. Kang, U., Chau, D. H. & Faloutsos, C. Inference of Beliefs on Billion-Scale Graphs. *The 2nd Work. on Large-scale Data Mining: Theory Appl.* (2010).

35. Ryder, B. G. Constructing the call graph of a program. *IEEE Transactions on Softw. Eng.* (1979).

36. Nemhauser, G. L., Wolsey, L. A. & Fisher, M. L. An analysis of approximations for maximizing submodular set functions. *Math. Program.* **14**, 265–294 (1978).

37. Amaldi, E. & Kann, V. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theor. Comput. Sci.* **209** (1998).

38. Miller, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychol. review* **63**, 81 (1956).

39. Narayanan, M. *et al.* An evaluation of the human-interpretability of explanation. *ArXiv* **abs/1902.00006** (2018).

40. Gonzalez, J., Low, Y. & Guestrin, C. Residual Splash for Optimally Parallelizing Belief Propagation. In *AISTAT*, vol. 5, 177–184 (2009).

41. Perozzi, B., Al-Rfou, R. & Skiena, S. Deepwalk: Online learning of social representations (ACM, 2014).

42. M, L. Reviews, reputation, and revenue:The case of yelp.com. In *Harvard business school working papers, Harvard Business School* (2011).

43. Zheng, H. *et al.* Smoke screener or straight shooter: Detecting elite sybil attacks in user-review social networks (2018).

44. Wojnowicz, M. *et al.* Influence sketching: Finding influential samples in large-scale regressions. *Big Data* (2016).

45. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR* (2013).

46. Baehrens, D. *et al.* How to Explain Individual Classification Decisions. *JMLR* (2010).

47. Krakovna, V. & Doshi-Velez, F. Increasing the Interpretability of Recurrent Neural Networks Using Hidden Markov Models. In *ICML Workshop on Human Interpretability in Machine Learning* (2016).

48. Norton, S. W. An explanation mechanism for bayesian inferencing systems. In *UAI*, 165–174 (1986).

49. Teach, R. L. & Shortliffe, E. H. An analysis of physician attitudes regarding computer-based clinical consultation systems. *Comput. Biomed. Res.* (1981).

50. Chang, J., Gerrish, S., Wang, C., Boyd-graber, J. L. & Blei, D. M. Reading Tea Leaves: How Humans Interpret Topic Models. In *NIPS* (2009).