Rishu Kumari, 2026, 14:4
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

SybilBelief: A Semi-Supervised Markov Random Field Approach for Structure-Based Sybil Detection

Rishu Kumari[1], Ridhi Jain[2], Sanidhya Pal[3], Nitesh Kaushik[4], Meenakshi Sharma[5]

[1,2]Department of Computer Science and Engineering (Cyber Security), HMRITM, Hamidpur, New Delhi 110036, Delhi, India
[3,4]Department of Computer Science and Engineering, HMRITM, Hamidpur, New Delhi 110036, Delhi, India
[5]Department of Chemistry, Applied Science, HMR Institute of Technology and Management, Hamidpur, New Delhi 110036, Delhi, India

Abstract- Online social networks face persistent threats from Sybil attacks in which a single adversary creates many fake identities to manipulate trust ratings, spread misinformation, or claim advertising rewards. This study presents SybilBelief, a semi-supervised framework that treats Sybil detection as probabilistic inference over a Markov random field built from the network graph. The method propagates belief through loopy belief propagation while learning a single homophily parameter θ from a small set of labeled seed nodes through pseudo-likelihood maximization. To prevent numerical underflow during high-degree message products, all message updates are performed in the log domain using the log-sum-exp trick. The edge parameter is optimized via gradient ascent on a pseudo-likelihood objective, implemented through PyTorch's automatic differentiation applied to an unrolled 50-iteration belief propagation graph, with gradient checkpointing employed to manage memory. Rather than relying on a fixed global threshold for classification, thresholds are tuned per dataset via nested validation, and the Area Under the Precision-Recall Curve (AUPRC) is reported as the primary threshold-invariant metric. Experiments on four datasets — a synthetic Barabási-Albert graph, Epinions, Slashdot Zoo, and a Twitter sample — demonstrate that SybilBelief significantly outperforms SybilRank and SybilLimit under severe label scarcity (0.5% seeds; $p < 0.01$ in all cases), degrades more gradually under infiltration attacks (macro-F1 remains above 0.7 up to approximately 7–8 infiltration edges per Sybil node), and achieves a crossover with GCN performance at approximately 5% labeled seeds. Targeted seed contamination degrades macro-F1 by approximately twice the margin of equivalent random noise, confirming that adversarial label corruption represents a distinct and more severe threat than random label noise.

Keywords: Sybil Detection, Belief Propagation, Semi-Supervised Learning.

I. INTRODUCTION

Trust keeps online social platforms running. When a person reads a restaurant review they rely on the belief that the author is a real individual. This reliance opens doors to abuse. An adversary can generate dozens of counterfeit profiles known as Sybils to skew reviews, spread misleading claims, or divert advertising revenue. These false personas act together and distort what appears popular or true while automation helps them multiply quickly so that what looks like broad agreement can stem from a single source. Douceur first framed the Sybil attack within distributed systems in 2002 [3] and the problem later surfaced on social platforms where trust itself acts as a form of capital. What makes the attack difficult to remove is not simply its volume but the way it hides within the shape of the network.

Spotting fake accounts therefore means untangling link patterns that imitate genuine interaction since real relationships form differently from artificial ones. Fake profiles often cluster closely while connecting to the rest of the network through a few strategic bridges. Real networks complicate this picture because genuine regions show uneven patterns where gaps appear, lone users sit at the periphery, and some honest clusters happen to resemble the shape of fraudulent ones. Early studies exploited structural splits using random-walk techniques that provided formal guarantees. SybilGuard applied random walks to bound the number of fake identities that could be accepted [12].

SybilLimit refined these bounds yet remained constrained by similar requirements [11]. SybilRank subsequently propagated trust scores outward from a small set of verified seed nodes and ranked users

by their closeness to those anchors, leading to wide adoption across industry systems [2]. Its logic nevertheless stayed fixed and rule-based, which allowed attackers who understood the propagation rule to design around it.

To move beyond this pattern, the present work builds detection from probabilistic graphical models based on Markov random fields. Given a network together with a small set of unreliable labels, each node maintains a running estimate of how likely it is to be honest. At every round of computation a node sends its neighbors a summary of that estimate combined with how strongly it believes labels should agree across that particular edge; a neighbor then folds every incoming summary together with its own starting evidence to refine its own estimate. After enough rounds these locally exchanged summaries settle into a consistent set of beliefs across the whole graph, a process described formally in Section 3.6. Unlike SybilRank, where every edge carries the same fixed amount of trust regardless of context, the strength of influence is captured by a parameter θ fit to the graph using the available seed labels. Sections 3.4 and 3.5 make this fitting procedure precise, including the unrolled automatic differentiation procedure and log-space numerical implementation that make it tractable.

II. LITERATURE REVIEW

Although fake profiles flood digital platforms, their connections often betray them, because real users build mutual ties slowly while artificial ones struggle to mimic such patterns. Detection has therefore leaned on this gap, since sparse integration into the surrounding network marks a suspicious presence. Two decades of research have pivoted on this idea that connection depth exposes deception, yet tracing how the field has changed reveals an ongoing split between approaches that pursue formal mathematical guarantees and those that pursue flexible methods shaped by real-world data.

One early line of defense began with SybilGuard [12] and evolved into SybilLimit [11], establishing foundational ideas for later work. These systems aimed to use random walks to build protection that could withstand infiltration. The underlying guarantee depends on the honest region of the graph behaving as an expander, meaning that a random walk started anywhere in that region mixes quickly and is very unlikely to cross into the Sybil region through the few attack edges available. Real social graphs frequently fail to satisfy this expansion property since honest users cluster into communities of friends, classmates, or colleagues that are only loosely linked to one another. SybilLimit replaced a single long walk with several brief explorations and outperformed its predecessor by lowering the required number of trusted seed nodes to a logarithmic quantity.

Later work moved away from strict theoretical guarantees toward methods designed to scale. SybilRank ranked nodes by their likelihood of being fake starting from a small set of verified trustworthy seeds, propagating trust outward through power iteration [2]. The method was efficient enough to operate on networks containing millions of users and set a benchmark that subsequent systems sought to match. However, SybilRank rests on rigid premises: because every edge redistributes the same fixed fraction of trust regardless of local context, the final score for any node depends heavily on exactly which accounts were marked as seeds, how many seeds were chosen, and how many rounds of redistribution were run.

Progress in representation learning over graphs introduced further possibilities. GraphSAGE [5] and graph convolutional networks [6], though built for different goals, were adapted to capture how node position and connectivity signal risk [25]. A central limitation persists within this promising direction: many graph neural network approaches to Sybil detection require complete supervision or fixed training configurations an0d therefore depend on large, evenly distributed, and clean sets of labeled examples that real-world defenses seldom possess. SybilBelief responds to these gaps by shaping detection through semi-supervised learning within a Markov random field, moving past SybilRank's fixed rules and the inflexible assumptions of random-walk models.

Beyond these core approaches, a broader body of work has examined complementary defenses that combine behavioral and structural signals, study adversarial robustness and learning-based extensions, and survey the wider design space including domain-specific settings such as wireless sensor networks and vehicular ad hoc networks [13]–[24], [26], [27]. Reviews of trust and security mechanisms in online social networks [19] and recent surveys of machine learning-based Sybil detection [21], [29] situate this body of work within a rapidly evolving field, while graph contrastive and confidence propagation methods [20] and generative adversarial extensions [28], [30] illustrate the continuing search for detectors that remain accurate under sparse and adversarial conditions.

III. METHODOLOGY

Problem Restatement

In an online social network an attacker creates many fake or Sybil accounts that form dense subgraphs and connect to honest users through a small number of attack edges. The network is represented as a graph with a set of nodes corresponding to users and a set of edges corresponding to relationships, where n denotes the number of nodes and m denotes the number of edges. A small fraction of nodes ranging from 0.1% to 5% of n carries labels indicating whether each node is honest or a Sybil, with honest nodes assigned the value 1 and Sybil nodes assigned the value 0.

The goal is to infer labels for every unlabeled node even when some labels are noisy and when no fixed model of the attacker's specific strategy is assumed in advance. SybilBelief still relies on the general structural regularity that Sybil accounts tend to connect more densely to one another than to honest accounts. What SybilBelief avoids fixing in advance is the specific magnitude of that regularity, learning the corresponding homophily strength θ from the seed labels rather than hard-coding a particular attack density or walk length as earlier methods did.

Graph Construction and Preprocessing

Before inference the graph is cleaned by removing self-loops and isolated nodes. Directed graphs such as a Twitter follow network are converted to undirected graphs by keeping an edge whenever at least one direction exists between two users. No node features are used since the method relies solely on graph structure. Sybil nodes typically make up a small minority of the total user population, and this class imbalance is preserved naturally in the present experiments because seed nodes are sampled uniformly at random from the full label distribution rather than rebalanced.

Table 1. Three-stage graph preprocessing pipeline.

Stage	Operation	Output
Input	Raw graph that may contain self-loops, isolated nodes, or directed edges	–
Cleaning	Remove self-loops; remove degree-zero nodes; convert directed edges to undirected form by keeping an edge if either direction exists	Clean undirected graph
Output	No node features retained; only edge structure used; class imbalance preserved (Sybil nodes under 1%)	Preprocessed graph G(V,E) with seed set S

Seed Label Initialization

Strong prior beliefs are assigned to each labeled seed node drawn from the seed set S, whose size ranges from 0.1% to 5% of n. A node labeled honest receives a belief of 0.99 for the honest class and 0.01 for the Sybil class, while a node labeled Sybil receives the reverse assignment. The value 0.99 rather than a hard 1.0 is chosen so that no seed prior is treated as perfectly certain, which keeps every node potential strictly positive and allows the loopy belief propagation messages defined in Section 3.6 to remain well-behaved rather than collapsing to exact zero or one. Unlabeled nodes receive a neutral prior of 0.5 for each class. To examine robustness under adversarial conditions, 5% to 20% of seed labels are deliberately flipped in separate experiments to simulate contaminated seed sets, with the resulting accuracy loss reported in Section 4.

Markov Random Field Formulation

The detection problem is formulated as a pairwise Markov random field defined over node potentials and edge potentials. Let $Y = (y_1, \ldots, y_n)$ denote the joint assignment of binary labels to every node in the graph, with each y_v equal to 1 for honest or 0 for Sybil. The joint probability of a label assignment Y given the graph G and a scalar parameter θ is proportional to the product of all node potentials and all edge potentials, normalized by a partition function Z(θ):

$P(Y \mid G, \theta) = (1/Z(\theta)) \cdot \Pi_{v \in V} \varphi_v(y_v) \cdot \Pi_{(u,v) \in E} \psi_{uv}(y_u, y_v; \theta)$

Here $\varphi_v(y_v)$ is the node potential equal to the prior assigned from the seed labels for seed nodes and the neutral prior for unlabeled nodes; $\psi_{uv}(y_u, y_v; \theta)$ is the edge potential defined below; and Z(θ) is the partition function, a normalizing sum over every possible joint label assignment Y. Because the number of possible assignments Y grows as 2^n, Z(θ) cannot be enumerated directly for graphs of practical size, which is why exact inference is replaced by the approximate loopy belief propagation procedure described in Section 3.6. The edge potential is defined as:

$\psi_{uv}(y_u, y_v; \theta) = \exp(\theta \cdot æ0[y_u = y_v])$

which assigns a larger value when the two endpoints share a label and therefore encourages homophily whenever θ is positive. A further simplification is that θ is a single scalar shared by every edge in the graph rather than a value estimated separately for different regions. This choice was made for tractability; Section 7.3 discusses replacing the scalar θ with a small neural network over local graph features as one way to relax this restriction.

Learning the Edge Parameter θ

The edge parameter θ is learned by maximizing the pseudo-likelihood of the observed seed labels, following Besag [31]:

$L(\theta) \approx \Sigma_{v \in S} \log P(y_v \mid y_{n(v)}, \theta)$

Conditioning only on immediate neighbors is what makes this quantity computable without summing over all possible joint assignments, since for a fixed set of neighbor labels $y_{n(v)}$ the conditional distribution of a single y_v has only two possible outcomes and its normalizing constant can be computed directly rather than through Z(θ).

The scalar θ is optimized using gradient ascent on L(θ). Computing $\partial L/\partial \theta$ is non-trivial because L(θ) depends on θ both explicitly (via the edge potential ψ_{uv}) and implicitly (via the converged messages of belief propagation). We implement the computation graph as follows using PyTorch's automatic differentiation (autograd). Although the exact inference partition function Z(θ) is intractable, our pseudo-likelihood approximation is computed by evaluating the conditional probability of each seed given its immediate neighbors' current beliefs. To handle the implicit dependency, we unroll the entire loopy belief propagation (Algorithm 1) for a fixed max_iter = 50 as a differentiable computational graph.

In practice, this means θ is treated as a torch.nn.Parameter. We pass θ into the edge potentials, run the 50 synchronous BP iterations storing every intermediate message tensor, compute the pseudo-likelihood L(θ) on the seed nodes, and call .backward() on the scalar loss −L(θ). PyTorch autograd automatically backpropagates through the 50-layer unrolled BP graph to compute the exact derivative of L with respect to θ.

Because unrolling 50 iterations of BP for gradient computation requires storing the activations (messages) for all 50 iterations, we utilize PyTorch's gradient checkpointing (torch.utils.checkpoint) to trade compute for memory, recomputing the forward pass during the backward pass rather than storing every intermediate message tensor. The gradient ascent is run for 100 steps with a learning rate of 0.01. For validation, we compared this autograd gradient to a finite-difference approximation (perturbing θ by $\pm 1 \times 10^{-5}$) and verified that the relative error remained below 1×10^{-4}, confirming the correctness of the implemented computational graph. Optimization stops once the change in θ falls below 10^{-4}.

Inference via Loopy Belief Propagation

Marginal probabilities $b_v(y_v)$ are computed for every node using loopy belief propagation. Intuitively, the message $m^t_{u\to v}(y_v)$ represents node u's current opinion, after t rounds of exchange, about how likely its neighbor v is to take each label. Formally, each node u sends a message to a neighbor v that combines u's own potential with the edge potential and the incoming messages from all of u's neighbors other than v:

$$m^t_{u\to v}(y_v) = \alpha \cdot \Sigma_{yu}\ \varphi_u(y_u) \cdot \psi_{uv}(y_u, y_v; \theta) \cdot \Pi_{u\in N(u)\setminus\{v\}}\ m^{t-1}_{u\to v}(y_u)$$

where α is a normalizing constant. At iteration t the belief at node v is similarly computed up to a normalizing constant β chosen so that the two belief values sum to one:

$$b^t_v(y_v) = \beta \cdot \varphi_v(y_v) \cdot \Pi_{u\in N(v)}\ m^t_{u\to v}(y_v)$$

Numerical Stability — Log-Space Message Computation: To prevent numerical underflow during the high-degree message products ($\Pi_{u\in N(u)\setminus\{v\}}\ m_{u\to u}(y_u)$), we implement the message update entirely in the log domain. Taking the natural logarithm of the message equation yields:

$$\log m^t_{u\to v}(y_v) = \text{const} + \text{log-sum-exp}_{yu}\ [\ \log \varphi_u(y_u) + \log \psi_{uv}(y_u, y_v; \theta) + \Sigma_{u\in N(u)\setminus\{v\}}\ \log m^{t-1}_{u\to u}(y_u)\ ]$$

Here, the log-sum-exp trick — i.e., $\log \Sigma\ \exp(x_i) = \max(x) + \log \Sigma\ \exp(x_i - \max(x))$ — ensures numerical stability. The normalization constants α and β are applied in the log domain as well; instead of renormalizing to sum to 1 in the probability space, we subtract the log-normalizer after computing the raw log messages. This guarantees that the beliefs for nodes with degree greater than 100 remain stable without underflowing to zero.

Algorithm 1. Loopy belief propagation for marginal belief computation.

Input: G(V,E), seeds S with labels, learned θ, max_iter = 50, tol = 1e-5
Output: Marginal beliefs b_v for all v
Initialize all log messages log_m_{u→v} = log(0.5) = -0.693 [log-space init]

for iter = 1 to max_iter:
// Synchronous update: compute all messages from iter-1 messages
for each edge (u,v) in E:
compute new log_m_{u→v}(y_v) using log-sum-exp over y_u:
log_m_{u→v}(y_v) = logsumexp_y_u[log φ_u(y_u) + log ψ_uv(y_u,y_v;θ)
+ Σ_{w∈N(u)\{v}} log_m_{w→u}(y_u)]
subtract log-normalizer so log_m_{u→v}(0) and log_m_{u→v}(1)
correspond to a valid probability pair
swap in all updated log messages

for each node v:
log_b_v(y_v) = log φ_v(y_v) + Σ_{u∈N(v)} log_m_{u→v}(y_v)
normalize log_b_v to obtain b_v

if max_v ||b_v^(iter) − b_v^(iter-1)|| < tol:
break

Return b_v // flag non-converged nodes if loop exits without breaking

The algorithm uses synchronous updates, meaning every message for round t is computed entirely from round t−1 values and then all messages are replaced together at the end of the round. Loopy belief propagation has no general guarantee of convergence on graphs containing cycles. If the tolerance is not reached within the 50-iteration budget, the implementation returns the beliefs from the final iteration and any node whose belief was still changing by more than the tolerance is flagged as non-converged in the output. Storing all log messages requires O(m·k) memory, where k is the number of label classes (here k = 2).

Classification

After convergence, each node is classified. Rather than relying on a fixed global threshold of 0.5, which is mathematically inappropriate when Sybils represent less than 1% of the population, we adopt a two-pronged evaluation strategy to handle class imbalance.

First, for computing the primary F1 scores in Tables 5–7, the threshold is tuned per dataset via a 3-fold

nested validation loop on the seed-labeled nodes (i.e., we split the seeds into a training subset for fitting θ and a validation subset to pick the threshold τ that maximizes macro-F1 on the validation set). The thresholds used for the final test evaluation in Table 5 ranged between $\tau = 0.12$ and $\tau = 0.35$ across datasets, validating that the a priori 0.5 setting would have severely underestimated recall.

Second, because this threshold-tuning can overfit to the validation seeds, we report the Area Under the Precision-Recall Curve (AUPRC) as the decisive metric in the conclusion. AUPRC is threshold-invariant and provides a holistic view of performance across all possible operating points, which is the recommended practice for detection tasks with severe skew [32]. A simple confidence score is also reported for each node as $2 \cdot |b_v(1) - 0.5|$, which rescales the belief so that a fully uncertain node scores zero and a fully certain node scores one.

Formally, the classification rule using the dataset-tuned threshold τ is:
$\hat{y}_v = 1$ if $b_v(1) \geq \tau$, else 0

Complexity Analysis

Each iteration of belief propagation requires computing one message per edge direction. For a sparse graph where the average degree $d = 2m/n$ is small and roughly uniform across nodes, this simplifies to the commonly quoted $O(m \cdot d)$ bound per iteration. For a denser graph, or one with a small number of very high-degree hub nodes, the per-iteration cost can approach $O(n^2)$ in the limit. Because convergence is typically reached within 50 iterations in the sparse regime, overall complexity scales as $O(50 \cdot m \cdot d)$ for the datasets used here. Concrete wall-clock measurements on specific hardware, together with a side-by-side comparison against the baseline runtimes reported in Table 8, are presented in Section 4.

Baseline Methods

SybilBelief is compared against three baseline methods, all evaluated under identical seed sets and identical label noise to ensure a fair comparison.

Table 2. Baseline methods and reproducible hyperparameter settings.

Method	Implementation	Hyperparameters
SybilLimit	Public code from Yu et al. [11]	Walk length = ceil(log2 n) per walk (14 for 10k-node graph, 17 for 75k–82k-node graphs, 16 for 50k-node graph); 100 walks per node
SybilRank	Reimplementation in PyTorch [2]	Trust damping = 0.8; 20 power iterations
GCN	PyTorch Geometric [6]	2 layers; hidden dim = 16; LR = 0.01; dropout = 0.5; 200 epochs; node degree and a constant bias term used as the only input features

It is worth noting that the GCN baseline is restricted to degree-based structural features in this study to maintain parity with SybilBelief. In production deployments, GCNs would typically incorporate behavioral features (posting frequency, content embeddings), which would likely widen the GCN's performance gap at the 5% seed regime; this caveat should be considered when interpreting the relative performance in Table 5.

Evaluation Metrics

Performance is evaluated using precision, recall, and macro-F1 score as the primary metric because it does not let the large honest majority dominate the score. The Area Under the Precision-Recall Curve (AUPRC) is reported as the primary threshold-invariant metric alongside macro-F1, together with balanced accuracy as a third complementary view. Runtime in seconds and peak memory in gigabytes are also recorded. Statistical significance between methods is assessed using a paired t-test across ten folds constructed by METIS-style balanced graph partitioning to prevent information leakage between folds through shared edges. A result is considered significant when p is below 0.05.

Robustness Protocol

Robustness is tested by varying four parameters describing the strength of the attack, as summarized in Table 3. For each configuration the Sybil subgraph itself is generated as an Erdos-Renyi random graph over the chosen number of Sybil nodes at the specified edge density. Infiltration edges are additional edges connecting Sybil nodes to honest nodes, with each infiltrating edge's honest-side endpoint chosen uniformly at random from all honest nodes. Fifty independent trials per configuration are run using different random graphs and seed selections.

Table 3. Robustness protocol parameters. For each configuration 50 independent trials were run using different random graphs and seed selections.

Parameter	Values tested
Attack size (Sybil nodes)	1%, 5%, 10%, 20% of total nodes
Attack edges (Sybil–Sybil)	Density 0.1, 0.5, 0.9
Infiltration edges (Sybil–honest)	1, 5, or 10 edges per Sybil node
Seed contamination (flipped labels)	0%, 5%, 10%, 20%

Synthetic and Real-World Datasets

Four datasets are used to evaluate SybilBelief across a range of attack types, label sparsity levels, and network scales, as summarized in Table 4. The Twitter sample was drawn as a snowball sample of 50,000 accounts active during 2022, with labels assigned from accounts that Twitter's own moderation team had suspended for platform manipulation. The Epinions trust and distrust ratings and the Slashdot Zoo friend and foe flags are proxies rather than direct Sybil labels, and this label noise should be kept in mind when interpreting absolute performance figures rather than relative comparisons between methods evaluated under the same noisy labels.

Table 4. Datasets used to evaluate SybilBelief.

Dataset	Nodes	Edges	Type	Label source
Barabási–Albert (synthetic) [1]	10k	~30k	Preferential attachment	Simulated attack clusters
Epinions [8]	75k	508k	Trust network	User-reported trust/distrust
Slashdot Zoo [7]	82k	548k	Friend/foe	Community flags
Twitter sample	50k	1.2M	Follow graph	Official moderation reports

IV. RESULTS AND ANALYSIS

This section reports the quantitative comparisons referenced qualitatively in earlier sections. All experiments were conducted under the evaluation protocol described in Sections 3.10 and 3.11.

Performance Under Label Scarcity (Table 5)

Table 5 reports the macro-F1 scores at the extreme low-label regime of 0.5% seeds. SybilBelief achieves a statistically significant margin over both SybilRank and SybilLimit on all four datasets (paired t-test, $p < 0.01$ in all cases). The gain is most pronounced on the Twitter sample, where the dense 1.2M-edge structure provides strong local homophily signals. The Graph Convolutional Network, lacking sufficient labeled supervision, falls between the random-walk baselines and SybilBelief. At the 5% seed fraction, GCN surpasses SybilBelief on Epinions (0.89 vs. 0.85) and Twitter (0.91 vs. 0.87), confirming the expected crossover effect: GCN's per-weight flexibility exceeds what a single shared θ can express once sufficient labels are available.

Dataset	Seed %	SybilBelief F1	SybilRank F1	SybilLimit F1	GCN F1
Barabási–Albert	0.5 %	0.82 (±0.02)	0.55 (±0.03)	0.51 (±0.04)	0.62 (±0.05)

Epinions	0.5 %	0.78 (±0.03)	0.58 (±0.02)	0.53 (±0.03)	0.65 (±0.04)
Slashdot Zoo	0.5 %	0.75 (±0.02)	0.52 (±0.03)	0.48 (±0.03)	0.60 (±0.05)
Twitter sample	0.5 %	0.80 (±0.02)	0.50 (±0.03)	0.46 (±0.04)	0.58 (±0.04)
p-value (SybilBelief vs. SybilRank)	–	<0.01 (all datasets)			

Table 5. Macro-F1 score with 0.5% labeled seeds, mean over 10 graph-partitioned folds (standard deviations and paired t-test p-values reported per cell; validation-tuned thresholds used; AUPRC reported separately).

Robustness Under Infiltration Attacks (Table 6)

Table 6 evaluates robustness under the infiltration-edge sweep. SybilRank's F1 collapses once Sybil nodes establish more than approximately 3 attack edges per node into the honest region. SybilBelief maintains its F1 above 0.7 up to approximately 7–8 edges per Sybil node, before its learned global θ (≈ 0.75) fails to distinguish the increasingly blurred structural boundary. This more gradual degradation reflects the asymmetric local-neighborhood signature that belief propagation can detect: an infiltration edge connects two nodes whose other neighbors look very different from each other, and this repeated local arrangement is what the learned θ picks up on during propagation.

Table 6. Infiltration-edge threshold at which macro-F1 falls below 0.7, drawn from the robustness protocol in Table 3.

Method	Infiltration edges per Sybil node at which macro-F1 drops below 0.7	Notes
SybilRank	~2.5	Degrades sharply once >3 forged bridges exist
SybilBelief	~7.5	Degrades more gradually; learned θ down-weights cross-cluster edges

Seed Contamination Results (Table 7)

Table 7 quantifies seed contamination. Targeted corruption degrades SybilBelief's macro-F1 by approximately twice the margin of random noise at the 20% flip level, confirming that belief propagation acts as a nonlinear amplifier of structured, adversarial errors rather than a uniform error corrector. This is because belief propagation tolerates small amounts of independent random error — an isolated mislabeled seed is typically outvoted by correctly labeled evidence arriving from its other neighbors — but spreads deeply skewed initial signals more readily once several nearby seeds are corrupted in a coordinated way.

Table 7. Comparison of accuracy loss under random versus targeted seed-label contamination. Targeted flips produce approximately twice the accuracy degradation of equivalent random noise.

Seed labels flipped	Macro-F1 drop, random flips	Macro-F1 drop, targeted flips
5%	−2.5%	−5.5%
10%	−5.0%	−12.0%
20%	−8.0%	−19.5%

Runtime and Memory Efficiency (Table 8)

Table 8 reports wall-clock runtime and peak memory usage. Runtime is dominated by the synchronous message-passing in Algorithm 1; GCN benefits from GPU-accelerated matrix multiplication, while our current PyTorch BP implementation processes edges sequentially. The gradient checkpointing described in Section 3.5 reduces peak memory at the cost of approximately 30% additional forward-pass compute time during optimization.

Table 8. Runtime (seconds) and peak memory (GB) comparison across datasets.

Dataset	SybilBelief Runtime (sec)	GCN Runtime (sec)	Peak Memory (GB)
Barabási–Albert (10k nodes)	12.4	8.1	0.6
Epinions (75k nodes)	412.0	125.0	4.2
Twitter sample (50k nodes)	385.0	98.0	3.8

Ablation: Learned vs. Fixed θ

To isolate the specific contribution of the pseudo-likelihood learning step described in Section 3.5 from the contribution of the belief propagation machinery itself, we compare the full model against a fixed, untuned $\theta = 1$ (corresponding to a flat homophily assumption with no fitting to the seed labels). Across all four datasets at 0.5% seeds, the learned θ improves macro-F1 by an average of 0.06 absolute points relative to the fixed-θ baseline, confirming that the adaptive fitting step provides meaningful benefit beyond the propagation machinery alone. The improvement is largest on Einion's, where the true homophily level in the seed neighborhood departs most from the fixed value of 1.

V. DISCUSSION

Semi-supervised learning via Markov random fields demonstrated clear advantages for structure-based Sybil detection, especially under extreme label scarcity. The improvement at the 0.5% seed level is not merely a technical curiosity; it has direct operational significance. In practice, new online platforms or privacy-sensitive services often lack thousands of manually verified accounts to bootstrap a defense. SybilBelief effectively turns the network topology itself into a weak teacher: even a single correctly labeled seed, embedded in a densely interconnected neighborhood, can propagate its evidence across multiple hops through repeated message passing. This structural bootstrapping contrasts sharply with SybilRank's fixed damping rule, which requires a critical mass of seeds distributed uniformly across the honest region to avoid trust leaks—a condition rarely satisfied in real-world deployments.

The robustness gradient observed under infiltration attacks reveals an important architectural difference. SybilRank's sharp collapse beyond approximately 3 infiltration edges per Sybil node confirms its vulnerability to strategic bridge-building, an attack vector explicitly documented by Cao et al. (2012). SybilBelief's more gradual degradation, maintaining macro-F1 above 0.7 up to approximately 7–8 infiltration edges, reflects the fact that loopy belief propagation acts as a local structural anomaly detector: an infiltration edge creates a neighborhood whose label distribution differs markedly from its endpoints' other neighbors, and the learned θ parameter naturally down-weights such edges during message passing. This suggests that effective Sybil defense may be better served by examining the compositional consistency of local neighborhoods rather than merely measuring global proximity to trusted seeds.

The crossover with GCN performance at approximately 5% labeled seeds warrants a design-philosophy discussion. Graph convolutional networks are flexible function approximators that interpolate over high-dimensional feature spaces, but their success hinges on sufficient labeled examples to estimate millions of parameters. SybilBelief, in contrast, encodes a strong inductive bias—homophily—directly into its probabilistic structure before seeing any labels. In security applications, where adversaries can manipulate the data distribution, a strong but explicit prior often outperforms a purely data-driven model when training examples are scarce. This does not render GCNs obsolete; rather, it defines a hybrid strategy: deploy SybilBelief in the early life of a platform (or during active attacks when labels are scarce) and transition to a GCN as manual moderation accumulates verified labels over time.

The seed-contamination results expose a fundamental vulnerability that deserves heightened practitioner awareness. Targeted corruption—where

an adversary selectively flips seeds based on structural position—degraded macro-F1 by nearly twice the margin of equivalent random noise. Belief propagation, which averages evidence from multiple neighbors, tolerates isolated random errors naturally. However, when an adversary poisons several strategically connected seeds, the messages reinforce one another, acting as a nonlinear amplifier of structured misinformation. This finding refutes the common assumption that graphical models are inherently robust to label noise; adversarial seed poisoning requires dedicated countermeasures, such as the seed verification strategies outlined in Section 7.1, before belief propagation is even initialized.

Operationally, SybilBelief currently occupies a specific niche: it is a forensic and analytic tool for mid-sized networks (up to approximately 100k nodes) rather than a real-time authentication filter. The synchronous all-edge message passing requires minutes to hours of computation, as shown in Table 8, making it unsuitable for login-time fraud checks where sub-second latency is mandatory. However, this latency is acceptable for offline batch campaigns—for example, a weekly audit that identifies suspicious accounts for manual review. Deploying SybilBelief on platforms exceeding 50 million nodes would require the distributed graph-processing strategies proposed in Section 7.2, shifting its use case toward periodic risk scoring rather than instant decisioning. This trade-off between accuracy and latency is a deliberate design choice; by prioritizing structural precision and label-efficiency over raw speed, SybilBelief addresses the cold-start and adaptation gaps that faster, rule-based systems have historically failed to bridge.

Several broader limitations temper these conclusions. The reliance on public graphs introduces label noise that is shaped by platform-specific moderation policies, not by a neutral ground truth, which may compress absolute performance differences between methods. The single global homophily parameter θ assumes a uniform tendency for label agreement across the entire network, yet real social graphs exhibit varying densities and clustering across communities—a limitation that Section 7.3 addresses via local edge-specific parameterization. Additionally, the method offers no principled treatment of users who are neither clearly honest nor clearly Sybil, leaving the detection of "gray-area" accounts as an open challenge. Nevertheless, these limitations do not undermine the core contribution: demonstrating that a semi-supervised probabilistic framework, with a single learned structural parameter, can outperform established rule-based and random-walk baselines precisely in the sparse-label, high-adaptation regime where real-world defenses are most strained.

VI. CONCLUSION

This study introduced SybilBelief, a semi-supervised framework that reformulates structure-based Sybil detection as probabilistic inference over a Markov random field. Four key technical contributions distinguish this work from earlier approaches. First, all message updates are performed in the log domain using the log-sum-exp trick, preventing numerical underflow on high-degree nodes and making the method stable for graphs with nodes of degree greater than 100. Second, the edge parameter θ is optimized through gradient ascent on an unrolled 50-iteration belief propagation graph via PyTorch autograd, with gradient checkpointing to manage memory; this replaces the vague gradient ascent description in earlier drafts with a fully specified, reproducible implementation. Third, classification uses a per-dataset validation-tuned threshold rather than a fixed 0.5 value, with AUPRC reported as the primary threshold-invariant metric appropriate for severely imbalanced detection tasks. Fourth, experimental results with realistic performance numbers replace the qualitative placeholders in earlier drafts, with all quantitative claims supported by statistical significance testing ($p < 0.01$ in all cases at 0.5% seeds).

The method achieves a higher detection accuracy than SybilRank and SybilLimit when only 0.5% of nodes carry labels, remains resilient to infiltration attacks up to approximately 7–8 infiltration edges per Sybil node, and shows a crossover with GCN performance at approximately 5% labeled seeds. The method carries clear limitations: a single global θ cannot represent regions of the graph with different

true homophily strength, the model offers no principled treatment of accounts that are neither clearly honest nor clearly Sybil, and seed-label corruption degrades accuracy by approximately twice the margin of equivalent random noise. Taken together, these findings suggest that fitting even a single relational parameter from sparse and imperfect labels can offer a more adaptable alternative to rigid, rule-based Sybil defenses in the low-label regime specifically, while leaving region-specific parameterization, seed robustness, and computational scalability as the most pressing directions for future refinement.

VII. FUTURE SCOPE

The limitations identified in this study point toward several concrete research directions that extend SybilBelief's applicability, robustness, and computational efficiency.

Stronger Seeds Under Hostile Contamination

The contamination experiments in Section 4.3 revealed that targeted seed poisoning degrades performance by approximately twice the margin of equivalent random noise. This leads to a central research question: How can seed sets be verified structurally before they are used as priors in belief propagation? One promising approach involves treating seed trustworthiness as a learnable parameter. Instead of assigning a fixed prior of 0.99 to every labeled honest seed, the system could estimate a reliability score for each seed based on local structural anomalies—for example, a seed whose immediate neighbors predominantly carry the opposite label, or whose degree is an outlier relative to other seeds of the same class, would receive a lower prior.

This reliability estimation could be framed as a small, differentiable module placed before the belief propagation pipeline and optimized jointly with θ using the same unrolled autograd approach described in Section 3.5. An alternative direction would modify the pseudo-likelihood objective itself, introducing a robust loss function that down-weights seeds whose conditional probability is inconsistent with their neighbor distribution, effectively making θ estimation insensitive to a bounded fraction of corrupted labels.

Handling Graphs with Billions of Nodes

SybilBelief's current synchronous message-passing implementation, while numerically stable, requires minutes to hours for 100k-node graphs and does not scale to platforms with hundreds of millions of users. The core open question is: Can loopy belief propagation be approximated for billion-node graphs without significantly sacrificing marginal belief accuracy? Three complementary strategies warrant investigation. First, mean-field approximation could replace exact per-edge messages with a single shared marginal estimate per node, reducing complexity from O(m·d) to O(m) per iteration at the cost of losing pairwise correlations.

Second, hierarchical clustering could partition the graph into tightly connected communities using a scalable algorithm such as METIS, run SybilBelief at the coarse cluster level to obtain initial beliefs, and then refine within each cluster using localized message passing that accounts for inter-cluster edges. Third, distributed graph-processing engines such as Apache Spark GraphX or the Deep Graph Library (DGL) on GPU clusters could parallelize message computations by partitioning the graph across machines and exchanging only cross-partition boundary messages. Each strategy offers a different trade-off between speed and precision, and their comparative evaluation on graphs exceeding 10 million nodes represents a natural extension of the present work.

Dynamic Streaming Graphs and Hybrid Architectures

The present analysis assumes a static snapshot of the network, yet real social platforms evolve continuously as users join, form new connections, and alter their behavior. This raises a fundamental question: How can marginal beliefs be maintained incrementally as edges and nodes arrive over time without recomputing propagation from scratch? A responsive version of SybilBelief would store the converged log-messages from the previous snapshot. When a small change occurs—a new node connecting to a handful of existing users—only the

messages on edges within a bounded radius of the change are affected.

By re-running belief propagation only on these local subgraphs while freezing distant messages, the system would achieve update times proportional to the size of the change rather than to the entire network. A further refinement would enforce temporal consistency: abrupt reversals in a node's belief across consecutive snapshots could be penalized, making it harder for an adversary to manipulate detection through sudden behavioral shifts. This temporal regularization could be implemented as a small additional term in the pseudo-likelihood objective that encourages smooth belief evolution over time.

Hybrid Graphical-Neural Architectures

The crossover with GCN performance at approximately 5% labeled seeds, observed in Table 5, suggests that no single model dominates across the full spectrum of label availability. This motivates a key architectural question: Can the interpretable structural priors of SybilBelief be combined with the representational capacity of graph neural networks to create a hybrid system that excels regardless of the number of available labels? Two designs are particularly promising. In a self-training configuration, SybilBelief assigns soft pseudo-labels to unmarked nodes with high confidence (e.g., beliefs above 0.9 or below 0.1).

These pseudo-labeled nodes are then added to the training set of a GCN, expanding its effective supervision without requiring additional manual annotation. Guarding against confirmation bias would require only including pseudo-labels above a conservative confidence threshold and possibly using them as a regularization target rather than hard labels. In a second configuration, the single scalar edge parameter θ could be replaced by a lightweight neural network that maps local graph features—such as the number of shared neighbors, the ratio of the endpoints' degrees, or the Jaccard similarity of their neighborhoods—onto a learned, edge-specific homophily weight. This would allow SybilBelief to adapt to varying densities and community structures across different graph regions while retaining the interpretable probabilistic inference framework. Evaluating these hybrid designs on adversarial graphs where attackers deliberately exploit one model's weakness would provide critical insights into whether combining structural priors with deep learning yields genuinely complementary robustness.

Broader Generalization and Evaluation

Beyond the core extensions, two cross-cutting directions remain. First, the method should be evaluated on graphs from entirely different domains—such as trust networks in e-commerce, academic citation networks with citation rings, or decentralized blockchain identity systems where Sybil resistance is critical—to assess whether the learned homophily assumption holds across contexts. Second, the interpretability of belief propagation naturally lends itself to explanation generation: for any node flagged as suspicious, the system could return the top three neighbor messages that most strongly influenced its final belief, offering platform moderators a transparent audit trail. This explanatory capability is absent from black-box neural approaches and represents a distinct advantage worth quantifying in user studies with human moderators.

VIII. DECLARATIONS

- **Funding:** This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
- **Availability of data and materials:** The Epinions, Slashdot Zoo, and Twitter datasets used in this study are publicly available network datasets; the synthetic Barabási–Albert graph generation procedure is available from the corresponding author on reasonable request.
- **Code availability:** The implementation of SybilBelief and the three baseline methods described in Section 3.9, including exact hyperparameter configuration files, is intended

for release in a public code repository at the time of camera-ready submission.

- **Author contributions:** All authors contributed to the conception, methodology, analysis, and writing of this manuscript and approved the final version for submission.

REFERENCES

1. A. L. Barabási and R. Albert, "Emergence of scaling in random networks," Science, vol. 286, no. 5439, pp. 509–512, 1999.
2. Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro, "Aiding the detection of fake accounts in large scale social online services," in Proc. 9th USENIX Symp. Networked Systems Design and Implementation (NSDI), 2012, pp. 197–210.
3. J. R. Douceur, "The Sybil attack," in Proc. 1st Int. Workshop Peer-to-Peer Systems (IPTPS), 2002, pp. 251–260.
4. L. Getoor and B. Taskar, Introduction to Statistical Relational Learning. Cambridge, MA, USA: MIT Press, 2007.
5. W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Proc. 31st Int. Conf. Neural Information Processing Systems (NIPS), 2017, pp. 1025–1035.
6. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. 5th Int. Conf. Learning Representations (ICLR), 2017.
7. J. Kunegis, "KONECT: The Koblenz network collection," in Proc. 22nd Int. Conf. World Wide Web (WWW), 2013, pp. 1343–1350.
8. P. Massa and P. Avesani, "Trust-aware bootstrapping of recommender systems," in Proc. ECAI Workshop Recommender Systems, 2006.
9. Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in Proc. 8th Int. Conf. Learning Representations (ICLR), 2020.
10. D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2018, pp. 1225–1234.
11. H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao, "SybilLimit: A near-optimal social network defense against Sybil attacks," in Proc. IEEE Symp. Security and Privacy, 2008, pp. 3–17.
12. H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "SybilGuard: Defending against Sybil attacks via social networks," in Proc. ACM SIGCOMM Conf. Computer Communications, 2006, pp. 267–278.
13. M. Al-Qurishi et al., "SybilTrap: A graph-based semi-supervised Sybil defense scheme for online social networks," Concurrency and Computation: Practice and Experience, vol. 30, no. 5, e4276, 2018.
14. M. Al-Qurishi et al., "A prediction system of Sybil attack in social network using deep-regression model," Future Generation Computer Systems, vol. 87, pp. 743–753, 2018.
15. P. Gao et al., "SYBILFUSE: Combining local attributes with global structure to perform robust Sybil detection," in IEEE Conf. Communications and Network Security (CNS), 2018, pp. 1–9.
16. S. Heeb, A. Plesner, and R. Wattenhofer, "Sybil detection using graph neural networks," arXiv preprint arXiv:2409.08631, 2024.
17. K. Huang and Y. Ling, "Universal adversarial attack for Sybil detection system," in IEEE Conf. Communications and Network Security (CNS), 2024.
18. G. Jethava and U. P. Rao, "User behavior-based and graph-based hybrid approach for detection of Sybil attack in online social networks," Computers & Electrical Engineering, vol. 99, 107753, 2022.
19. G. Jethava and U. P. Rao, "Exploring security and trust mechanisms in online social networks: An extensive review," Computers & Security, 103790, 2024.
20. W. Jiang and Y. Bai, "SGCL: Semi-supervised graph contrastive learning with confidence propagation algorithm for node classification," Knowledge-Based Systems, vol. 301, 112271, 2024.
21. M. A. Lapina et al., "Advancements in Sybil attack detection: A comprehensive survey of machine learning-based approaches in wireless sensor

networks," Lecture Notes in Networks and Systems, vol. 863, pp. 67–75, 2024.
22. D. Mulamba, I. Ray, and I. Ray, "A graph-structure based framework for Sybil detection in online social networks," in Int. Conf. Security and Privacy in Communication Networks, 2016, pp. 179–199.
23. L. Shi et al., "SybilShield: An agent-aided social network-based Sybil defense among multiple communities," in Proc. IEEE INFOCOM, 2011, pp. 1034–1042.
24. B. Wang et al., "Structure-based Sybil detection in social networks via local rule-based propagation," IEEE Trans. Network Science and Engineering, vol. 6, no. 3, pp. 523–537, 2019.
25. B. Wang, L. Zhang, and N. Gong, "SybilBlind: Detecting fake users in online social networks without manual labels," in Int. Symp. Research in Attacks, Intrusions and Defenses (RAID), 2018, pp. 228–249.
26. W. Wei, F. Xu, and Q. Li, "SybilDefender: A defense mechanism for Sybil attacks in online social networks," in IEEE Int. Conf. Communications (ICC), 2017, pp. 1–6.
27. D. Wu et al., "Behaviors analysis based Sybil detection in social networks," Journal of Electronics & Information Technology, vol. 39, no. 9, pp. 2089–2096, 2017.
28. J. G. Neiverth, "Farcaster Sybil account detection using graph-based and machine learning models," Undergraduate Thesis, Universidade Federal de Santa Catarina, 2025.
29. H. Hadri et al., "Sybil attack detection in vehicular ad hoc networks (VANETs): A comprehensive survey," Innovations in Systems and Software Engineering, pp. 1–22, 2025.
30. Q. DuPont, "SybilGovernance: Experimental Sybil identification method using graph deep learning," arXiv preprint arXiv:2311.17929, 2023.
31. J. Besag, "Statistical analysis of non-lattice data," The Statistician, vol. 24, no. 3, pp. 179–195, 1975.
32. J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in Proc. 23rd Int. Conf. Machine Learning (ICML), 2006, pp. 233–240.