Shapley Values for Measuring Contributions in Influence Propagation in Networks

Fangzhu Shen* Amir Gilad† Sudeepa Roy*
*Duke University, USA †The Hebrew University, Israel

ABSTRACT

The widespread connectivity of social media and online platforms has transformed how information, behaviors, and innovations spread through a network, enabling rapid influence propagation through peer interactions. This process underpins critical applications such as viral marketing or public health messaging, and is typically driven by a small set of "seed" users. While much prior work has focused on selecting optimal seeds to maximize influence, a less explored but important problem is how to fairly measure and attribute each seed node's contribution to the overall outcome. We address this challenge by presenting a framework for measuring contributions of seed nodes using Shapley values from cooperative game theory, which provide a principled and fair attribution based on marginal contributions. However, adopting Shapley values to influence propagation presents significant computational challenges due to its exponential expression, probabilistic nature of influence propagation, and complex network structure with interdependencies among nodes. In this work, we provide poly-time algorithms for the special case of single-step activation, show #P-hardness even for two steps, and give algorithms with approximation guarantees. We experimentally evaluate our algorithms on real-world and synthetic datasets showing that they provide a practical mechanism for reward distribution in influence propagation.

PVLDB Reference Format:
Fangzhu Shen* Amir Gilad† Sudeepa Roy*. Shapley Values for Measuring Contributions in Influence Propagation in Networks. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

1 INTRODUCTION

The ever-growing connectivity of social media platforms and online communities has fundamentally transformed how information, behaviors, and innovations spread through a network. Within these highly-connected widely-used online environments, content such as news, advertisements, and political messaging can propagate swiftly through peer-to-peer interactions and word-of-mouth effects. This phenomenon, broadly termed *influence propagation* (or, *influence diffusion*) [16, 30, 33], is central to a wide range of high-impact applications, including viral marketing, public health messaging, outbreak surveillance, and the spread of news or concepts, making it a source of a multitude of important and interesting research problems.

Typically, influence propagation is initiated by a small set of influential or carefully selected users, referred to as *seed nodes* in the network. These nodes serve as the starting points of the diffusion process and trigger cascades of activations, or adoptions throughout the network. While a vast body of research has focused on influence maximization [30], i.e., identifying the optimal set of seed nodes to maximize future influence, a different but important question has received far less attention: *once a campaign is initiated with a set of seed nodes, how do we fairly measure each seed node's contribution to the overall outcome?* If we can correctly and efficiently quantify the contribution of each seed node, we can enable fair reward distribution and take targeted preventive actions in the case of negative influence, such as misinformation spread.

As a concrete example, consider a technology company launching a new mobile application through a social media campaign. The company enlists several influential users (bloggers, celebrities, tech reviewers) as initial promoters, with the goal of triggering a cascade of adoptions across the network. If the campaign concludes with millions of app downloads and wants to share part of the profit with the initial promoters, a key question arises: *how should the company fairly distribute rewards among these seed nodes based on their individual contributions to the campaign's success*? This attribution problem is challenging because influence in social networks is highly nonlinear along overlapping paths and involves complex interdependencies. A user's impact depends not only on their direct connections but also on intricate interactions with other influencers and non-influencers, the underlying network topology, and the timing of adoptions, all of which make the problem nontrivial to formalize and analyze.

To address this challenge of fairly measuring each seed node's contribution to influence propagation, we leverage the seminal concept of Shapley values [45] from cooperative game theory. Shapley values provide a principled method for distributing a collective payoff among players in a coalition by quantifying each player's contribution as their average marginal gain across all possible orderings of player participation. By modeling the seed nodes as players and the expected number of newly influenced nodes as the payoff, we can rigorously quantify each seed's contribution to the overall influence. Notably, the Shapley value is the unique solution that satisfies four desirable axioms: (1) *efficiency* – the total payoff is distributed among all players; (2) *symmetry* – players with identical marginal contributions receive the same value; (3) *linearity* – Shapley values are additive across games; and (4) *null player* – a

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

player who contributes nothing receives a value of zero. These axioms ensure that the Shapley value provides a fair and theoretically grounded attribution if adopted properly for influence propagation.

However, adopting Shapley values to influence propagation presents significant computational challenges. The expression for computing Shapley values is exponential in the number of players. Further, estimating each marginal contribution involves calculating expected influence since influence propagation itself is probabilistic in nature, which is itself computationally hard due to the complexity and nonlinearity of influence propagation. Prior work [10, 40] has applied Shapley values in network analysis, but primarily for ranking nodes by structural importance rather than for attributing influence among a fixed set of seeds. The work of Zhu et al.[56], most closely related to ours, computes Shapley values for influence attribution but focuses on attribution by analyzing fully observed cascades. This limits its applicability in practice for our application, where complete cascade data may be unavailable due to privacy or scale. In contrast, our work addresses a simpler and practical scenario: we compute Shapley values based on expected influence prior to observing the final outcome, enabling an essential capability for planning and executing reward mechanisms in real-world campaigns.

Our Contributions.

Our work presents a rigorous and practical mechanism for quantifying contributions to influencers in social advertising or other influence-centric applications with the following contributions:
(1) Framework: We formalize the problem by computation of Shapley values in influence propagation under the independent cascade model [30] under different termination criteria. **(Section 2)**
(2) **Poly-time algorithm for single-step termination:** We present a polynomial-time exact algorithm to compute Shapley values when activation happens in a single step circumventing the exponential summation in the formula. **(Section 3)**
(3) **#P-Hardeness:** We show that the problem is #P-hard under Complete and Fixed-steps terminations even for 2 steps. **(Section 4)**
(4) **Approximation algorithms:** We present algorithms to estimate Shapley values based on permutation sampling [9] and reverse reachable sets [48], with theoretical analysis on approximation bounds and computational complexity. **(Section 5)**
(5) **Experiments:** We empirically evaluate our approaches on large-scale, real-world datasets, demonstrating the efficiency–accuracy tradeoffs and showing that our algorithms provide a practical mechanism for reward distribution in influence propagation. **(Section 6)**
We conclude with related work and future work (Sections 7 and 8).

2 PRELIMINARIES

In this section, we first define the network, seed set, and review the Independent Cascade Model for influence diffusion in Section 2.1. Then we introduce the Shapley value and the value function defined for attributing influence in our setting in Section 2.3.

2.1 Network and Independent Cascade Model

We consider a (social) network modeled as a directed graph $G = (V, E)$, where V is the set of *nodes* (users), $E \subseteq V \times V$ is the set of directed *edges* (connections). We denote the set of *out-neighbors* of a node u as $N^+(u)$ and the set of *in-neighbors* of a node u as $N^-(u)$, i.e., $N^+(u) = \{v \in V \mid (u, v) \in E\}$ and $N^-(u) = \{v \in V \mid (v, u) \in E\}$.

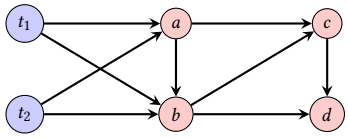


Figure 1: A toy network for the IC model (blue = seed nodes).

Influence diffusion and the independent cascade model. In *influence diffusion* or *influence propagation*, nodes in a network exist in one of two states: *active* or *inactive*. A node is an *active node* if the influence has reached the node, e.g., the node adopted the idea or behavior that is being propagated in the network. In turn, it can influence its out-neighbors to become active. This process is referred to as *activation*.

We consider the *independent cascade (IC) model* [30] as the model for influence diffusion. In the IC model, a node u can activate its out-neighbor v along the edge (u, v) with some probability $p(u, v)$ only in the next *time-step* (or, *step*) after u is activated, independent of the activation along the other edges in the network. The activation starts with a set of *seed nodes* that are initially active at time-step 0.

Formally, the input to the IC model is a tuple $(G(V, E), T, p)$ where $G(V, E)$ is a network, $T \subseteq V$ is the set of *seed nodes*, and $p : E \to [0, 1]$ denotes the function for the activation probability $p(u, v)$ for each edge $(u, v) \in E$. Let A_s denote the set of active nodes at step s.

- *Initialization*: In step $s = 0$, only seed nodes are active, i.e., $A_0 = T$.
- *Propagation*: In each step $s \geq 1$, each node u that is newly activated in the previous step $s-1$, (i.e., $u \in A_{s-1} \setminus A_{s-2}$), gets a single chance to activate each of its currently inactive out-neighbors v (i.e., $v \in N^+(u) \setminus A_{s-1}$) with the activation probability $p(u, v)$. If the activation is successful, v becomes active in step s (i.e., $v \in A_s$) and remains active in all subsequent steps.
- *Termination:* We consider various termination conditions for the diffusion process: (1) *Complete*: terminates when no new nodes are activated in a step (i.e., $A_i = A_{i-1}$), which is the standard IC model [30]. (2) *Fixed-steps*: terminates after a given fixed number of steps $\mathbb{K} \geq 1$. (3) *Single-step*: terminates after $\mathbb{K} = 1$ step.

We illustrate the activation process with an example:

Example 1. *Consider the network G shown in Figure 1, with nodes $V = \{t_1, t_2, a, b, c, d\}$. Each arrow represents an edge with the same activation probability of* 0.5*. The seed set is $T = \{t_1, t_2\}$, i.e., $A_0 = \{t_1, t_2\}$. In the first time-step $s = 1$, t_1 (or t_2) can activate its out-neighbors a and b with probability* 0.5 *independently. Therefore, the probabilities that node a or b becomes active in step $s = 1$ is* $\Pr[a \in A_1] = \Pr[b \in A_1] = 1 - (1 - 0.5)^2 = 0.75$.

Suppose node a becomes active and b stays inactive in step $s = 1$. In step $s = 2$, node a tries to activate each of its inactive out-neighbor $\{b, c\}$ with probability 0.5 *independently. Although b has incoming edges also from t_1, t_2, they cannot activate b as they become active in step 0 and not 1. If both b and c are successfully activated in $s = 2$, they will try to activate their out-neighbor d in $s = 3$, so the probability that d becomes active is* $\Pr[d \in A_3 | b, c \in A_2 \setminus A_1] = 1-(1-0.5)^2 = 0.75$. *If d becomes active, the process terminates at $s = 4$ as no new nodes become activated under Complete termination. For Fixed-steps termination with $\mathbb{K} = 2$, d remains inactive.*

2.2 Shapley Values

The *Shapley value* is a seminal concept from cooperative game theory [45] to fairly attribute the total value generated by a coalition of players to individual players. Given a set of players N and a value (or utility) function $U : 2^N \to \mathbb{R}$ with $U(\emptyset) = 0$, where $U(S)$ denotes the value raised by the subset S of players, the Shapley value for each player $i \in N$ is defined as the average marginal contribution of i over all possible permutations of the player set:

$$Shap_{N,U}(i) = \frac{1}{|N|!} \sum_{\pi \in \Pi(N)} \left[U(S_{\pi,i} \cup \{i\}) - U(S_{\pi,i}) \right] \quad (1)$$

Here $\Pi(N)$ is the set of all permutations of N, and $S_{\pi,i}$ is the set of players in N that precede i in permutation π. An equivalent formula for Shapley values is the following:

$$Shap_{N,U}(i) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N|-|S|-1)!}{|N|!} \left[U(S \cup \{i\}) - U(S) \right] \quad (2)$$

Due to several properties of Shapley values in fairly measuring each player's contribution as discussed in the introduction, they have been widely used in various research areas in both machine learning (ML) and databases to measure or explain contributions of different entities, such as contribution of a feature or data point in prediction or classification in ML [1, 21, 22, 38, 51, 54], or contribution of a tuple or constraint in databases [14, 15, 29, 36, 37, 44].

2.3 Shapley Values for Influence Diffusion

In the context of influence diffusion in a network, our goal is to measure the contribution of each seed node to the diffusion process based on the final set of active nodes in the graph when the diffusion terminates. Hence, the seed nodes T form the set of players.

Value Function. Next we define the value function as the number of non-seed nodes that get activated when the diffusion is terminated. The seed nodes are excluded in the value function – they are already activated at step 0 and therefore do not contribute to the value collected by the seed nodes (players) in diffusion.

The definition of the value function $U(S)$ for a given subset of seed nodes $S \subseteq T$ uses a subgraph $G_{T,S}$. In $G_{T,S}$, we remove the seed nodes $T \setminus S$ that do not belong to S and also remove their incident edges (incoming and outgoing). Then, the value is given by the expected number of non-seed nodes from $V \setminus T$ at the end of the diffusion process starting with S as the seed nodes in $G_{T,S}$.

Definition 1 (Value Function of a Set of Seed Nodes). *Given $(G(V,E), p, T)$ with seed nodes T, and a subset $S \subseteq T$, let $G_{T,S}(V', E')$ be the subgraph of G where $V' = V \setminus (T \setminus S)$ and E' is formed by removing all incoming and outgoing edges of $T \setminus S$ from E. Then given a termination condition (Complete, Fixed-steps, or Single-step), the* value function $U : 2^T \to \mathbb{R}$ *for any subset $S \subseteq T$ is defined as:*

$$U(S) = \mathbb{E}\left[\left| \{u \in V \setminus T \mid u \text{ is activated in } (G_{T,S}, p, S)\} \right| \right] \quad (3)$$

under the termination condition (the expectation $\mathbb{E}$ is over p).

A natural property of this formulation is that $U(\emptyset) = 0$, making the value function valid for Shapley values, as an empty set of seed nodes cannot activate any nodes[1].

[1]We discuss the relationship between $U(S)$ and the influence function is a related problem of influence maximization in the full version [46]. See Appendix A.

Example 2. *Consider again the network G from Figure 1 and the seed set $T = \{t_1, t_2\}$. For the subset $S = \{t_1\}$, the graph $G_{T,S}$ is shown in Figure 2. For Single-step termination, we only consider the direct activation from t_1. Thus, both nodes a and b can be activated by t_1 at time-step $s = 1$ with probability 0.5, so $U(\{t_1\}) = 0.5 + 0.5 = 1$. For Fixed-steps (with $k > 1$) and Complete terminations, to calculate $U(\{t_1\})$ in the brute-force way, we need to consider all possible activation paths and calculate the probability of each inactive node being activated along that path.*

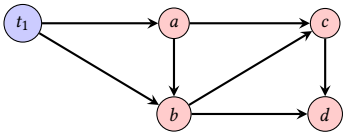


Figure 2: The induced network to compute $U(\{t_1\})$ by removing all edges starting from t_2.

Shapley value for Seed Nodes. The Shapley value $Shap(t)$ for each seed node in T is obtained by using the value function Eq. (3) from Definition 1 in Eq. (1, 2):

Definition 2 (Shapley value of a Seed Node). *Given $(G(V,E), p, T)$, the Shapley value of a seed node $t \in T$ is given by:*

$$Shap(t) = \frac{1}{|T|!} \sum_{\pi \in \Pi(T)} \left[U(S_{\pi,t} \cup \{t\}) - U(S_{\pi,t}) \right] \quad (4)$$

$$= \sum_{S \subseteq T \setminus \{t\}} \frac{|S|!\,(|T|-|S|-1)\,!}{|T|!} \left[U(S \cup \{t\}) - U(S) \right]. \quad (5)$$

where $\Pi(T)$ is the set of all permutations of T and $S_{\pi,t}$ is the set of nodes in T that precede t in permutation π.

Example 3. *Continuing the example in Figure 1, to compute the Shapley value $Shap(t_1)$ for seed node t_1 under any of Single-step, Fixed-steps, Complete terminations, we consider its marginal contribution over two possible permutations $\{t_1, t_2\}$ and $\{t_2, t_1\}$. Hence,*

$$Shap(t_1) = \frac{1}{2} \left[(U(\{t_1\}) - U(\emptyset)) + (U(\{t_2, t_1\}) - U(\{t_2\})) \right].$$

Special case – single seed node. When there is a single seed node, i.e., $T = \{t\}$ for some $t \in V$, the following observation holds:

Observation 1. *Given $(G(V,E), p, T)$ where $T = \{t\}$, the Shapley value of t is: $Shap(t) = U(\{t\})$.*

This follows from the property of Shapley values [45] that the sum of Shapley values of all players is equal to the value of the grand coalition, i.e., $\sum_{i \in N} Shap_{N,U}(i) = U(N)$ in Eq. (1, 2). Intuitively, the single seed node t is responsible for all the influence diffusion in the network, and thus receives full credit for it.

Complexity overview. Our goal is to design efficient algorithms to compute $Shap(t)$ for a given seed node $t \in T$ in $(G(V,E), p, T)$, or for all seed nodes in T, under different termination criteria (Complete, Fixed-steps, Single-step) in polynomial time in the number of nodes in the network $|V|$. The problem becomes complex for an arbitrary set of seed nodes T with $|T| > 1$ in an arbitrary graph G, since multiple seed nodes may be responsible for the same non-seed nodes being activated. In the following sections, we show that $Shap(t)$ for a $t \in T$ can be computed in poly-time in Single-step activation, but the problem becomes #P-hard [49], even for Fixed-steps with $\mathbb{K} = 2$ steps and for Complete terminations, where we present efficient algorithms as practical solutions.

3 SINGLE-STEP TERMINATION: POLY-TIME

In this section, we show that the Shapley values of seed nodes can be computed in polynomial time under Single-step termination. While Single-step termination seems easier than the other termination cases since each non-seed node can get activated only by direct edge (path of length 1), still Eqs. 4, 5 involve summation over exponentially-many terms in $|T|$ and $|T|$ can be $O(|V|)$, making the problem non-trivial to solve in poly-time in $|V|$ even for Single-step.

Theorem 3.1. *Given $(G(V,E),T,p)$, the Shapley value $Shap(t)$ for every seed node $t \in T$ under Single-step termination can be computed in polynomial time in $|V|$.*

We first present two lemmas in Section 3.1 with useful properties of Shapley values that hold under Single-step termination. Lemma 1 simplifies the Shapley value computation to a closed-form expression, and Lemma 2 develops an efficient recurrence relation for the core summation. Based on these results, we present the poly-time algorithm ExactSingleStep (Algorithm 1) in Section 3.2. We then present an optimized algorithm ExactSingleStepOpt (Algorithm 2) in Section 3.3 that shows significant scalability in experiments. Detailed proofs are given in the full version [46]. See Appendix B.

3.1 Properties under Single-step Termination

In this section, we present two important properties of the Shapley values of seed nodes under Single-step termination.

Lemma 1. *For a seed node $t \in T$, under Single-step termination,*

$$Shap(t) = \sum_{k=0}^{|T|-1} \frac{k!\,(|T|-k-1)!}{|T|!} \cdot \left[\sum_{u \in \{V \setminus T\} \cap N^+(\{t\})} p_{t,u} \cdot \alpha_{u,t}(T \setminus \{t\}, k) \right] \tag{6}$$

$$\text{where} \quad \alpha_{u,t}(T \setminus \{t\}, k) = \sum_{\substack{S \subseteq T \setminus \{t\} \\ |S|=k}} \left(\prod_{w \in N^-(u) \cap S} (1 - p_{w,u}) \right). \tag{7}$$

We still need to compute $\alpha_{u,t}(T \setminus \{t\}, k)$ efficiently in Eq. (6). Therefore, we develop a more general recurrence relation for any subset of $T \setminus \{t\}$ in the following lemma:

Lemma 2. *Given a seed node $t \in T$, one of its non-seed out-neighbor $u \in \{V \setminus T\} \cap N^+(\{t\})$ and an integer $k \in \{1, \ldots, |T|-1\}$, the following holds for any subset $L \subseteq T \setminus \{t\}$:*

$$\alpha_{u,t}(L,k) = \begin{cases} 1 & \text{if } k = 0 \\ (1 - p_{n,u}) \cdot \alpha_{u,t}(L \setminus \{n\}, k-1) & \\ +\alpha_{u,t}(L \setminus \{n\}, k) & \forall n \in L, \text{if } 1 \le k \le |L| \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

$$\text{where } \alpha_{u,t}(L,k) = \sum_{\substack{S \subseteq L \\ |S|=k}} \left(\prod_{w \in N^-(u) \cap S} (1 - p_{w,u}) \right).$$

3.2 ExactSingleStep: Poly-time Exact Algorithm

Using Lemmas 1 and 2, we now present the algorithm ExactSingleStep (Algorithm 1). The computation procedure follows Eq. (6). We first precompute all binomial coefficients, and store them in a coefficient array C (line 2). Then for each seed node t, and its non-seed out-neighbors u (line 4), we compute $\alpha_{u,t}(T \setminus \{t\}, k)$ for all k values as a summation array $\alpha_{u,t}$ via dynamic programming (line 6): we initialize $\alpha_{u,t}[0] = 1$ and $\alpha_{u,t}[k] = 0$ for $k > 0$, and consider nodes in $T \setminus \{t\}$ by an arbitrary order: $\{n_1, n_2, \cdots, n_{|T|-1}\}$. We iterate over i from 1 to $|T|-1$, and for each node n_i in thie sequence, we update the $\alpha_{u,t}$ array by iterating k from i down to 1:

$$\alpha_{u,t}[k] \leftarrow \alpha_{u,t}[k] + (1 - p_{n_i,u}) \cdot \alpha_{u,t}[k-1]. \tag{9}$$

In line 7, we update $Shap(t)$ by adding the dot product of coefficient array C and summation array $\alpha_{u,t}$, multiplied by $p_{t,u}$. Finally, the algorithm returns $Shap(t)$ for each seed node $t \in T$.

Algorithm 1 ExactSingleStep(G,T,p)

Input: Network $G(V,E)$, seed set T and activation probabilities p
Output: Shapley values $Shap(t)$ for each seed node $t \in T$

1: Initialize $Shap(t) \leftarrow 0$ for each seed node
2: Compute coefficients $C[k] \leftarrow \frac{k!\,(|T|-k-1)!}{|T|!}$ for $0 \le k \le |T|-1$
3: **for all** $t \in T$ **do**
4: **for all** $u \in \{V \setminus T\} \cap N^+(\{t\})$ **do**
5: *// Iterate over all non-seed nodes that are out-neighbors of t*
6: Compute $\alpha_{u,t}[k]$ for all k by bottom-up DP using Lemma 2
7: $Shap(t)$ += $p_{t,u}$ $\sum_{k=0}^{|T|-1} (C[k] \cdot \alpha_{u,t}[k])$
8: **end for**
9: **end for**
10: **return** $Shap(t)$ for each $t \in T$

The following lemma establishes that ExactSingleStep correctly computes the Shapley values in polynomial time.

Lemma 3. *Given $(G(V,E),T,p)$,* ExactSingleStep *(Algorithm 1) computes the Shapley values of all seed nodes under Single-step termination in $O(|V|^4)$ time.*

3.3 ExactSingleStepOpt: Optimization

A large seed set size $|T|$ can result in extremely large values for $C[k]$ and $\alpha_{u,t}[k]$ as k increases. In practical implementations, these values may exceed the standard 32- or 64-bit numeric limits, leading to overflow and incorrect results. However, in real-world networks, most non-seed nodes are connected to only a small subset of seed nodes. Thus, we leverage this sparsity and propose a numerically stable algorithm ExactSingleStepOpt (Algorithm 2).

Algorithm 2 ExactSingleStepOpt(G,T,p)

Input: Network $G(V,E)$, seed set T and activation probabilities p
Output: Shapley values $Shap(t)$ for each seed node $t \in T$

for all non-seed node $u \in V \setminus T$ **do**
$T(u) \leftarrow$ set of seed nodes that are in-neighbors of u
if $T(u) \neq \emptyset$ **then**
Build a subgraph $G_u(V_u, E_u)$ where $V_u = T(u) \cup \{u\}$ and E_u consists of edges between $T(u)$ and u
$Shap_u(t) \leftarrow$ ExactSingleStep$(G_u, T(u), p)$
end if
end for
$Shap(t) \leftarrow \sum_{u \in \{V \setminus T\} \cap N^+(T)} Shap_u(t)$
return $Shap(t)$ for each $t \in T$

For each non-seed node u, we identify the set of seed nodes that have incoming edges to u, denoted as $T(u)$ (line 2). Thus we form a local subgraph $G_u(V_u, E_u)$ where $V_u = T(u) \cup \{u\}$, and $E_u = \{(t, u) \mid t \in T(u)\}$ (line 4). In line 5, we compute the local Shapley values for each seed node in $T(u)$ on $G_u(V_u, E_u)$ using ExactSingleStep, denoted as $Shap_u(t)$. Then line 8 aggregates the local Shapley values $Shap_u(t)$ to the global Shapley values $Shap(t)$. Following Lemma 4 implies that the global Shapley value computation can be decomposed into independent local subproblems:

Lemma 4. *Given $(G(V, E), T, p)$, the Shapley value of each seed $t \in T$ can be computed as:*

$$Shap(t) = \sum_{u \in \{V \setminus T\} \cap N^+(t)} Shap_u(t) \tag{10}$$

Based on this decomposition property, the correctness and time complexity of ExactSingleStepOpt is shown below:

Lemma 5. *Given $(G(V, E), T, p)$,* ExactSingleStepOpt *(Algorithm 2) correctly computes the Shapley values of all seed nodes under Single-step termination in $O(|V|^4)$ time.*

Since there is only one non-seed node u in each subgraph $G_u(V_u, E_u)$, we only need to compute $O(|T|)$ summation arrays per subgraph, thus the runtime of each ExactSingleStep call is reduced to $O(|V|^3)$. Given that there are at most $|V|-|T|$ such subgraphs, the overall time complexity of ExactSingleStepOpt remains $O(|V|^4)$, as the same as ExactSingleStep. Moreover, our experiments in Section 6 show that ExactSingleStepOpt avoids the overflow issue.

4 #P-HARDNESS FOR FIXED-STEPS AND COMPLETE TERMINATIONS

In this section, we prove that the the computation of Shapley values becomes intractable for Complete and Fixed-steps termination models, even for termination in $\mathbb{K} = 2$ steps.

Theorem 4.1. *Computing $Shap(t)$ for a seed node $t \in T$ is #P-hard for Complete termination and for Fixed-steps termination with $\mathbb{K} \geq 2$.*

In this section, we present the key ideas from the proof; complete proofs are provided in the full version [46]. See Appendix C. We establish the hardness result for Fixed-steps termination with $\mathbb{K} = 2$ steps via a reduction from the #P-complete problem of counting satisfying assignments for a monotone 2-CNF formula [49]. In this graph, the maximum directed path-length is 2 and hence the hardness extends to Complete termination and Fixed-steps termination with $\mathbb{K} \geq 2$ steps.

Let ϕ be a monotone 2-CNF formula with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Although the satisfiability of monotone 2-CNF formulas can be decided in poly-time, the counting version #2-CNF is known to be #P-hard [50]. Here each clause C_j is a disjunction of two unnegated variables. For each clause C_j, define $\text{Var}(C_j) = \{x_i \mid x_i \text{ appears in } C_j\}$.

The proof for 2-step termination proceeds in two main parts: (1) constructing a family of influence graphs $G(r, q)$ using parameters r, q that we define below, and deriving a set of Shapley values on these graphs, and (2) recovering the number of satisfying assignments of the 2-CNF formula from these Shapley values. The reduction is a general Turing reduction and is not *parsimonious* (one to one) since it uses the oracle to compute Shapley values multiple times to solve #2-CNF.

(1) Graph Construction and Shapley Value Expression: Given the monotone 2-CNF formula ϕ, we build a graph $G_{(r,q)} = (V_{(r,q)}, E_{(r,q)})$ (e.g., Figure 3) for every ordered pair $(r, q) \in [n] \times [m]$. The construction introduces a *variable seed node* u_i for each variable x_i, a *clause node* v_j for each clause C_j, a *sink node* s, and r *auxiliary seed nodes* $t_1, \ldots, t_r$. We create edges with probability 1 from u_i to v_j whenever $x_i \in \text{Var}(C_j)$, edges with probability $\frac{q}{q+1}$ from every v_j to s, and edges with probability 1 from every auxiliary seed node t_ℓ to s. The construction of $G_{(r,q)}$ takes polynomial time, and we construct $m \cdot n$ such graphs $G_{(r,q)}$ in total.

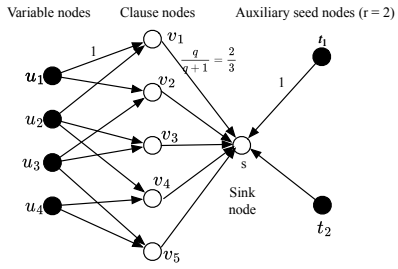


Figure 3: an example of graph $G_{(r,q)}$ where $r = 2, q = 2$. $\phi = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4)$.

We will use the Shapley values of the first auxiliary seed node t_1 from graphs $G_{(r,q)}$, which we denote by $Shap_{(r,q)}(t_1)$, in the second step of our hardness proof (t_1 is chosen arbitrarily). For t_1, its only chance to influence s is through the edge (t_1, s) in time-step 1, so the Shapley value $Shap_{(r,q)}(t_1)$ of t_1 in graph $G_{(r,q)}$ can be simplified as follows ($\mathbb{1}$ is the indicator function): $Shap_{(r,q)}(t_1) =$

$$\frac{1}{n+r} + \sum_{k=1}^{n} \frac{k!\,(n+r-k-1)!}{(n+r)!} \cdot \sum_{\substack{S \subseteq U \\ |S|=k}} \left(\frac{1}{q+1}\right)^{\sum_{v_j \in V} \mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i, v_j) \in E_{(r,q)}]} \tag{11}$$

(2) Computing the number of satisfying assignments of ϕ: We denote the number of satisfying assignments of the 2-CNF formula ϕ by $\#_\phi$. We show that if one can compute $Shap_{(r,q)}(t_1)$ for every pair of (r, q) in poly-time, then $\#_\phi$ can be computed in poly-time.

First, we express both $\#_\phi$ and $Shap_{(r,q)}(t_1)$ in terms of $\gamma_{k,c}$, which denotes the number of size-k subsets of variables such that the assignment setting these k variables to true (and all others to false) satisfies exactly c clauses in ϕ. Then $\#_\phi$ can be expressed as:

$$\#_\phi = \sum_{k=1}^{n} \gamma_{k,m}, \tag{12}$$

and $Shap_{(r,q)}(t_1)$ can be expressed as:

$$Shap_{(r,q)}(t_1) = \frac{1}{n+p} + \sum_{k=1}^{n} \sum_{c=1}^{m} \frac{k!\,(n+p-k-1)!}{(n+p)!} \cdot \left(\frac{1}{q+1}\right)^c \cdot \gamma_{k,c}. \tag{13}$$

Collecting Eq. (13) for all pairs of (r, q) gives the matrix system: $A \times \Gamma \times B = D$; where: A is an $n \times n$ matrix with entries $a_{r,k} = \frac{k!(n+r-k-1)!}{(n+r)!}$;$B$ is an $m \times m$ matrix with entries $b_{c,q} = (\frac{1}{q+1})^c$;$D$ is an $n \times m$ matrix with entries $d_{r,q} = Shap_{(r,q)}(t_1) - \frac{1}{n+r}$;$\Gamma$ is an $n \times m$ matrix with each entry is $\gamma_{k,c}$. Both coefficient matrices A and B

are non-singular (details in the full version [46]), so we can solve for all $\gamma_{k,c}$ in polynomial time if the Shapley values $Shap_{(r,q)}(t_1)$ are known. Finally, we can sum $\gamma_{k,m}$ over all k to compute $\#_\phi$ in polynomial time, completing the reduction.

5 COMPLETE TERMINATION: ESTIMATION

In this section, we present three efficient algorithms for computing $Shap(t)$ for $t \in T$ under Complete termination with theoretical approximation guarantees: (1) In Section 5.1, we adapt a standard permutation-sampling-based Monte-Carlo (MC) algorithm [9] to the influence diffusion setting, which serves as the baseline method. (2) In Section 5.2 we present a novel single live-edge graph realization approach to replace computationally expensive MC simulations, which significantly improves runtime while maintaining identical approximation guarantees; (3) In Section 5.3, we adapt the reverse reachable (RR) set method [10, 47] to our problem of computing Shapley values for a given set of seed nodes, which achieves superior scalability. Our analysis focuses on the Complete termination because it represents the most general case, and the algorithms can be easily adapted to Fixed-steps termination. Detailed theoretical analyses are provided in the full version [46]. See Appendix D and E.

5.1 Monte-Carlo Permutation Estimator

Here we adapt the permutation-sampling framework [9, 40] to get an unbiased estimator confidence bounds for Shapley values in influence propagation. Based on the formula of Shapley value in Eq. (4) that computes the average of marginal contributions over all possible permutations of the seed set, this algorithm first samples $n_\pi = O\left(\frac{|V|^2}{\epsilon^2}\ln(\frac{|T|}{\delta})\right)$ random permutations of T. Each permutation is a specific ordering of seed nodes. Thus, for each permutation $\pi_i = \{t_1, t_2, \ldots, t_{|T|}\}$ and each seed t_j, it computes the marginal contribution $\hat{U}(S_{\pi_i,j}\cup\{t_j\})-\hat{U}(S_{\pi_i,j})$, where $S_{\pi_i,j}$ contains the seeds preceding t_j in the permutation order π_i. Here, $\hat{U}(S)$ estimates the value function by computing the average number of nodes activated by any seed set S over $n_{\text{MC}} = O\left(\frac{|V|^2}{\epsilon^2}\ln(\frac{|V|^2|T|^2}{\epsilon^2\delta})\right)$ simulations of the influence propagation process. The final estimated Shapley value averages these marginal contributions across all sampled permutations. Using Hoeffding's inequality [26], we establish the following theoretical performance guarantees on both accuracy and efficiency (pseudocode and analysis in the full version [46]):

Proposition 1. *Given* $(G(V,E),T,p)$, ApproxPermuteMC *returns* $\widehat{Shap(t)}$ *for all* $t \in T$ *such that: (1)* $\mathbb{E}[\widehat{Shap(t)}] = Shap(t)$, $\forall t \in T$; *(2) for any* $\epsilon > 0$ *and* $\delta \in (0,1)$, *with probability at least* $1-\delta$, $|\widehat{Shap(t)} - Shap(t)| \leq \epsilon$ *holds for all* $t \in T$; *(3) the running time is* $O\left(\frac{|V|^4}{\epsilon^4}\ln(\frac{|T|}{\delta} + \frac{|V|^2|T|^2}{\epsilon^2\delta})\cdot|T|\cdot|E|\right)$.

The runtime complexity arises from the nested structure: ApproxPermuteMC simulates influence propagation n_{MC} times for computing each marginal contribution, and the process is repeated over $|T|$ seeds and n_π permutations.

5.2 Single-Realization Permutation Estimator

The ApproxPermuteMC requires a nested loop that uses n_{MC} simulations to estimate value functions within each permutation π_i, creating a computational bottleneck that limits the scalability. To address this, we propose ApproxPermuteDirect, which generates a single live-edge graph, i.e., a realization of the diffusion process by sampling all edges, for each permutation. Since the influence on the live-edge graph is deterministic, the marginal contribution can be computed directly through a single graph traversal.

The new algorithm ApproxPermuteDirect (Algorithm 3) uniformly samples $n_\pi = O\left(\frac{|V|^2}{\epsilon^2}\ln(\frac{|T|}{\delta})\right)$ permutations of T at random (Line 4). For each permutation π_i, it samples a *live-edge graph* g_i (Line 5), where each edge in g_i is sampled based on its activation probability. Therefore, influence diffusion in g_i is deterministic and each live-graph g_i contains the set of nodes that are reachable from T in this realization. Then for the pair of (π_i, g_i) and each seed t_j in this permutation, Lines 7, 8 and 10 computes the marginal contribution as $u_{g_i}(S_{\pi_i,j} \cup \{t_j\}) - u_{g_i}(S_{\pi_i,j})$, where $u_{g_i}(S)$ denotes the number of non-seed nodes that are reachable from seed nodes in S in g_i. Finally, Line 14 returns the average marginal contribution over n_π permutations as the estimated Shapley values.

Algorithm 3 ApproxPermuteDirect$(G, T, p, \epsilon, \delta)$

Input: Network $G(V,E)$, seed set T, activation probabilities p, approximation parameters $\epsilon > 0$, $\delta \in (0,1)$

Output: Estimated Shapley values $\widehat{Shap(t)}$ for all $t \in T$

1: Number of permutations $n_\pi \leftarrow O\left(\frac{|V|^2}{\epsilon^2}\ln(\frac{|T|}{\delta})\right)$
2: Initialize $est[t] \leftarrow 0$ for all $t \in T$
3: **for** $i = 1$ to n_π **do**
4: Sample a random permutation π_i of T
5: Sample a live-edge graph g_i from G with p
6: **for** each t_j in π_i **do**
7: $u_{g_i}(S_{\pi_i,j} \cup \{t_j\}) \leftarrow$ number of reachable nodes from $S \cup \{t_j\}$ in g_i
8: $u_{g_i}(S_{\pi_i,j}) \leftarrow$ number of reachable nodes from S in g_i
9: *// $S_{\pi_i,j}$ denotes the number of seeds preceding t_j in π_i*
10: $est[t_j]$ += $u_{g_i}(S \cup \{t_j\}) - u_{g_i}(S)$
11: $S \leftarrow S \cup \{t_j\}$
12: **end for**
13: **end for**
14: **return** $\widehat{Shap(t)} \leftarrow \frac{1}{n_\pi} est[t]$ for all t

Despite the simplification of using a single realization per permutation, ApproxPermuteDirect achieves identical unbiasedness and approximation guarantees as ApproxPermuteMC ((1, 2) from Proposition 1). Moreover, since the algorithm traverses the graph once per marginal contribution computation for each of the n_π permutations, it significantly improves runtime performance, as shown in the following Proposition 2 that improves $|V|^4$ to $|V|^2$.

Proposition 2. *For any* $\epsilon > 0$ *and* $\delta \in (0,1)$, *the total running time of* ApproxPermuteDirect *is* $O\left(\frac{|V|^2}{\epsilon^2}\ln(\frac{|T|}{\delta})\cdot|T|\cdot|E|\right)$.

5.3 Reverse-Reachable Set Estimator

We present a third approximation algorithm, which is based on *Reverse-Reachable (RR)* sets. Unlike the previous permutation-based estimators, this algorithm avoids iterating through seed permutations, leading to better scalability. An RR set for a node v consists

of all nodes that can reach v in a random live-edge graph, where each edge is sampled independently based on its activation probability. The key insight is that the expected frequency of a node's appearance in random RR sets correlates with its influence in G.

Definition 3 (Reverse Reachable (RR) Set [48]). *Given $(G(V,E), p, T)$, let $v \in V$ be a node in G and g be a live-edge graph sampled from G by removing each edge e with $1-p(e)$ probability. The RR set for v in g is the set of nodes that can reach v in g.*

A *random RR set* is generated by first selecting a node $v \in V$ uniformly at random and then generating an RR set for v on a live-edge graph g randomly sampled from G.

Adaptations for Shapley value estimation. RR sets were originally developed for influence maximization problems [5, 47, 48], and have also been applied to Shapley centrality estimation [10]. However, these prior works address different problems: influence maximization seeks to select an optimal seed set, while Shapley centrality measures the importance of all nodes in the network using Shapley values (no seed nodes). Differently, our goal is to compute the Shapley value of each node in a *fixed* seed set T with respect to a *value function* $U(\cdot)$ that counts only non-seed activations.

This difference necessitates modifications on both RR set construction and the input graph: (1) all RR sets are rooted at non-seed nodes rather than all nodes, and (2) all incoming edges to seed nodes are removed to avoid influence propagation between seeds. These modifications ensure that the Shapley value of any seed t can be connected to the expectation over random RR sets, as shown in the following lemma (the proof is in the full version [46], see Appendix E):

Lemma 6. *Let $G' = (V, E')$ be the subgraph of G obtained by removing all incoming edges to the set of seed nodes T, and $\boldsymbol{R}$ be a random RR set generated in G' by selecting a node v uniformly at random non-seed node from $V \setminus T$. Then, for any seed node $t \in T$:*

$$Shap(t) = n' \cdot \mathbb{E}_{\boldsymbol{R}}\left[\frac{\mathbb{I}\{t \in \boldsymbol{R}'\}}{|\boldsymbol{R}'|}\right] \tag{14}$$

where $n' = |V \setminus T|$, $\boldsymbol{R}' = \boldsymbol{R} \cap T$, and $\mathbb{I}\{\cdot\}$ is the indicator function.

The identity reveals that the Shapley value of t equals to the sum of expected influence on each non-seed node, where the credit for each node being activated is shared equally among all seeds that could have reached it.

Our Algorithm. Based on the above lemma, we propose ApproxRRset (Algorithm 4) to estimate the Shapley value of each seed t by sampling random RR sets. The algorithm is adapted from the two-phase adaptive-sampling framework of [10], and takes three parameters: ε sets the multiplicative error, ℓ determines the confidence guarantee $1 - \frac{1}{n^\ell}$, and k sets the threshold as the multiplicative error bound holds for the k largest Shapley values. It runs in three phrases.

- **Phase 0: Graph Modification.** The algorithm first removes all incoming edges to seed nodes to generate a modified graph G' (Line 2), and the size of non-seed nodes is denoted as n' in Line 4. This ensures that the generated RR sets only measure influence on non-seed nodes.
- **Phase 1: Parameter Estimation.** EstimateThreshold estimates a lower bound $\boldsymbol{LB}$ for $Shap^{(k)}$, the k-th largest true Shapley value, by following a similar sampling strategy as [47]. Then in Line 8, it determines θ, the number of RR sets required to satisfy the desired approximation guarantee. EstimateThreshold works by iteratively tightening estimation thresholds. For each threshold, it generates the required number of RR sets and check whether the estimated k-th largest Shapley values meets the current threshold. If the current threshold is met, the algorithm terminates and uses this estimate to compute the lower bound $\boldsymbol{LB}$. Otherwise, it increases the number of RR sets and repeats the process. The detailed procedure is in the full version [46]. See Appendix E.
- **Phase 2: Shapley Value Estimation.** Then we generate θ RR sets as in Line 13. For each RR set R_j, we increment the score $\boldsymbol{est}_t$ of every seed t appeared in R_j, by $1/|R_j \cap T|$, an estimate of the probability of t to activate the root node of R_j (Lines 15-16). The final Shapley value estimates $\widehat{Shap}(t)$ are computed by normalizing these scores (Line 20).

Algorithm 4 ApproxRRset$(G, T, p, \varepsilon, \ell, k)$

Input: Network $G(V,E)$, seed set T, activation probabilities p, approximation parameters $\varepsilon > 0$, $\ell > 0$, $k \in [|T|]$
Output: Estimated Shapley values $\widehat{Shap(t)}$ for all $t \in T$

1: **Phase 0: Graph Modification**
2: $E' \leftarrow \{(u,v) \in E \mid v \notin T\}$; $G' \leftarrow (V, E')$
3: *// Remove all incoming edges to seed nodes*
4: $n' \leftarrow |V \setminus T|$
5: **Phase 1: Parameter Estimation**
6: $\boldsymbol{LB} \leftarrow$ EstimateThreshold$(G', T, \varepsilon, \ell, k)$
7: *// Find lower bound of the k-th largest Shapley value*
8: $\theta \leftarrow \left\lceil \frac{n'(2+\frac{2}{3}\varepsilon)}{\varepsilon^2 \cdot \boldsymbol{LB}} (\ell \ln n' + \ln|T| + \ln 4) \right\rceil$
9: *// Calculate required number of RR sets*
10: **Phase 2: Shapley Value Estimation**
11: Reset $\boldsymbol{est}_t \leftarrow 0$ for all $t \in T$
12: **for** $j = 1$ to θ **do**
13: Generate a random RR set R_j by selecting a node from $V \setminus T$ uniformly at random
14: **if** $R_j \cap T \neq \emptyset$ **then**
15: **for** each $t \in R_j \cap T$ **do**
16: $\boldsymbol{est}_t \leftarrow \boldsymbol{est}_t + \frac{1}{|R_j \cap T|}$
17: **end for**
18: **end if**
19: **end for**
20: **return** $\widehat{Shap}(t) \leftarrow n' \cdot \frac{\boldsymbol{est}_t}{\theta}$ for all $t \in T$

Approximation Guarantees. ApproxRRset provides theoretical guarantees on accuracy and runtime by adapting the analysis from [10, 47] to our problem (proof in the full version [46]. See Appendix E). For the k largest Shapley values $Shap^{(k)}$, the multiplicative error bound holds relative to their own Shapley values; while for the rest, the error bound holds for an absolute value $\varepsilon Shap^{(k)}$.

Proposition 3. *For any $\varepsilon > 0$, $\ell > 0$ and $k \in [|T|]$, suppose that $Shap^{(k)} \geq 1$, with probability at least $1 - 1/n^\ell$,* ApproxRRset *returns $\widehat{Shap}(t)$ for all $t \in T$ such that:*

$$\begin{cases} |\widehat{Shap}(t) - Shap(t)| \leq \varepsilon Shap(t), & \forall t \text{ with } Shap(t) > Shap^{(k)}, \\ |\widehat{Shap}(t) - Shap(t)| \leq \varepsilon Shap^{(k)}, & \forall t \text{ with } Shap(t) \leq Shap^{(k)}. \end{cases} \tag{15}$$

The following shows the expected runtime complexity of ApproxRRset, which is the product of the total number of RR sets generated in two phases, and the expected cost of generating one RR set.

Proposition 4. *Let $\tilde{v}$ be a random node drawn from $V \setminus T$ with probability proportional to the in-degree of $\tilde{v}$ in G'. Denote m' is the number of edges in G' and $\mathbb{E}_{\tilde{v}}[U(\tilde{v})]$ is the expected value function of $\tilde{v}$. Then the expected runtime complexity of* ApproxRRset *is:*

$$O\left(\frac{n\ell \log n}{Shap^{(k)}\varepsilon^2} \cdot \frac{m'}{n'} \cdot \mathbb{E}_{\tilde{v}}[U(\tilde{v})]\right), \tag{16}$$

under condition $\ell \geq \dfrac{\log_2 k - \log_2 \log_2 n' + \log_2 n' - \log_2 |T|}{\log_2 n'}$.

6 EXPERIMENTS

In this section, we examine the necessity of Shapley values for influence attribution and the performance of proposed algorithms. The experiments address the following questions:

- Section 6.2: Do Shapley values for influence diffusion proposed in our work reveal contributions of the seed nodes that adaptation of standard algorithms for influence maximization (IM) overlook?
- Section 6.3: How accurate are our algorithms under different scenarios (terminations, number of seeds, network structure)?
- Section 6.4: How efficient and scalable are our algorithms?

Summary of our findings. Our key findings are: (1) Common seed selection heuristics poorly reflect seed contributions, highlighting the importance of our Shapley-value-based attribution. (2) Under the Single-step termination, ExactSingleStepOpt provides exact Shapley values with good scalability (e.g., less than 7 minutes for 4.8M-node network with 1M seeds). (3) For the Complete termination model where exact computation is infeasible, there exists an accuracy-runtime trade-off between ApproxRRset and ApproxPermuteDirect. A higher accuracy comes at the cost of longer runtime. (4) Last, ApproxPermuteMC is impractical for most scenarios with poor scalability and low accuracy. We also studied the choice of effective parameters for each approximation algorithm to achieve a balance between accuracy and computational efficiency. Full experiments are in the full version [46].

6.1 Setup

We implement algorithms in Rust and conduct experiments on a machine with a commodity EPYC CPU (AMD EPYC 7R13 48-Core Processor @2.6GHz, Boost 3.73GHz, 192MB L3 cache; 256GiB DDR4-3200 memory). Code is publicly available [2].

Datasets. We evaluated our methods using both real-world and synthetic network datasets. Table 1 shows the 6 real-world networks that vary in size and density[3]. For synthetic networks, we generate directed random graphs using the Erdős-Rényi model [17] via NetworkX [25]. Specifically, given the number of nodes n and average degree d, the number of edges is $n \times d$, and the graph is randomly chosen from the collection of all satisfiable graphs. We vary the number of nodes, average degree, and edge activation probability schemes to study the performance of our algorithms under different network structures. We randomly generate 5 graphs for each combination of network structure parameters and take the average of the results.

[2]https://github.com/fangzhushen/Shapley-and-Influence-Diffusion

[3]The Congressional Twitter network represents Twitter interactions among members of the 117th U.S. Congress (Feb–June 2022); We construct the DBLP Coauthor network as a co-authorship network from the Proceedings of VLDB 2024 dataset.

Table 1: Real-world networks used in the experiments.

Dataset	# Nodes	Avg. Degree
Congressional Twitter [18, 19]	475	27.98
DBLP Coauthor [13]	≈1.7K	7.99
Facebook [35]	≈4K	43.70
Twitter [35]	≈81K	12.75
Orkut [53]	≈3.1M	76.28
LiveJournal [3, 34]	≈4.8M	14.23

Assigning probability to edges. We use the following schemes to assign activation probabilities p_{uv} for edges (u, v):

(1) Weighted cascade (WC): For an edge (u, v), $p_{uv} = 1/d_{in}(v)$, where $d_{in}(v)$ is the in-degree of node v. This standard scheme [30] ensures the expected number of incoming neighbors for any node v is 1 and is widely used [11, 24, 28]. We applied the WC scheme to the synthetic networks, DBLP Coauthor, Facebook, Twitter, Orkut, and LiveJournal networks.
(2) Learned probabilities: For the Congressional Twitter network, we used empirically learned influence probabilities from the original study [19]. These probabilities reflect the likelihood of influence based on observed Twitter interactions (e.g., retweets, mentions) between members of the 117th US Congress.
(3) Uniform probability: All edges share the same activation probability, i.e., $p_{uv} = p$ for all edges (u, v). We vary p from 0.01 to 0.8 for synthetic networks to study the impact of probability values on the performance of our algorithms.

Seed selection method. We vary the number of seeds (k) from 5 to 1M, where feasible. We employ the *degree-based selection* which selects k nodes with the highest out-degree [56]. This is a common computationally efficient method used for large-scale networks.

Algorithms and Parameter Settings. We evaluate the following algorithms:

- ExactSingleStepOpt (Algorithm 2,Section 3.2): Our numerically stable DP-based algorithm that computes exact Shapley values in polynomial time under the Single-step termination.
- ApproxPermuteMC (Section 5.1): The baseline apprpximation method. We set the number of permutation samples to 500 and the number of MC samples to 500 for experiments in Sections 6.3 and 6.4.
- ApproxPermuteDirect (Algorithm 3, Section 5.2): Our proposed approximation algorithm with direct marginal contribution estimation in sampled live-edge graph. We set the number of permutation samples to 5, 000.
- ApproxRRset (Algorithm 4, Section 5.3): Our proposed algorithm approximates Shapley values by sampling random RR-sets. We set the number of sampled RR-sets to 500, 000.

Since exact computation for the Complete termination is #P-hard and thus impractical, we use ApproxRRset with high-accuracy parameters ($\epsilon = 0.01$, $\ell = 1$) as the ground truth under the Complete termination. For the Single-step termination, ExactSingleStep can serve as the ground truth. Sampling parameters for each approximation algorithm are set to achieve a good balance between accuracy and efficiency, discussed in the full version [46].

Evaluation Metric. To assess the effectiveness of algorithms, we report the average of relative error compared to the true Shapley values, defined as:

$$\text{Avg. Relative Error} = \frac{1}{|T|}\sum_{i=1}^{|T|}\frac{|\widehat{Shap(t_i)} - Shap(t_i)|}{Shap(t_i)} \tag{17}$$

where $\widehat{Shap(t_i)}$ is the estimated Shapley value from the algorithm, and $Shap(t_i)$ is the ground truth Shapley value. We also report algorithm *runtime* to examine efficiency.

6.2 Seed Selection Heuristics vs. Shapley Values

Common methods to select seeds for influence maximization include degree-based heuristics for computational efficiency, and greedy Influence Maximization (IM) approximation algorithms for precision. However, it is unclear whether the metrics that guide seed selection can also explain the actual influence of each seed within the selected group. Specifically, *does a node selected due to high out-degree or early selection by a greedy IM algorithm inherently possess the greatest contribution computed by Shapley values?*

To explore this question, we investigate whether the rank of a node by these seed selection methods aligns with its Shapley value on two real-world networks: the DBLP Coauthor and the Congressional Twitter networks. We use two seed selection methods: degree-based selection and greedy IM-based selection. *Greedy IM-based selection* utilizes the greedy algorithm [47] for influence maximization (IM) problem, which iteratively selects nodes that maximize marginal influence spread using RR sets. Our finding reveals that Shapley value distributions differ significantly from both selection criteria, highlighting the need for principled attribution through Shapley values.

Tables 2 and 3 show the top-10 nodes of each network ranked by out-degree and their corresponding Shapley value rankings when choosing these top-10 nodes as the set of seed nodes. In both networks, we observe discrepancies between out-degree rankings and Shapley value rankings. For instance, in the DBLP Coauthor network, *Christian S. Jensen* and *Guoliang Li 0001* have higher Shapley values than the top-ranked node by out-degree. Similarly, in the Congressional Twitter network, *SteveScalise* ranked 5th by out-degree gets the highest Shapley value, especially than *SenSchumer*, which has 97 out-degree and ranked 4th but only gets the 7th highest Shapley value. High out-degree does not consistently correspond to high marginal influence because Shapley values account for complex interactions among nodes during the diffusion process, whereas out-degree reflects only direct connections. Additional results with selecting top-50 nodes by out-degree are presented in the full version [46].

We also analyze the Shapley values of seeds selected by the greedy IM algorithm (Figures 4a and 4b). Although nodes selected earlier by the greedy algorithm have higher incremental influence at the time of selection, this does not correspond to higher Shapley values in the influence diffusion process. For example, in the DBLP Coauthor network, the 30^{th} selected node outperforms its previous 20 selected nodes in terms of its Shapley value; and in the Congressional Twitter network, the 48^{th} selected node achieves the 2^{nd} highest Shapley value, outperforming the 1^{st} selected node which ranked 11^{th} by the Shapley values. Our results demonstrate that seed-selection methods are inadequate for measuring the individual contributions within seed nodes. Degree-based approach fails to account for overlapping influence paths, while the greedy IM approach evaluates a node's contribution only at the moment it is selected rather than within the final set. In contrast, the Shapley value provides a theoretically grounded and accurate attribution of individual node to the overall influence diffusion.

6.3 Quality Analysis

In this section, we evaluate the accuracy of our proposed algorithms, on both real-world and synthetic datasets, varying the number of seeds and key network structure parameters (size, density, and edge activation probabilities).

Varing the number of seeds under Complete termination: Figure 5 shows the average relative error of different algorithms when varying the number of seeds on both 4K-node Facebook and 8K-node Twitter networks. For Facebook network, we have following observations: (1) All algorithms exhibit a peak-and-decline pattern as the number of seeds increases. This occurs because absolute error (nominator) decreases while true Shapley values (denominator) decrease faster, due to increased overlap among seeds, causing relative error to initially rise. After peaking around 500-1K seeds, the relative error declines as absolute error reduction becomes more significant. (2) Both ApproxPermuteDirect and ApproxRRset maintain low error (below 4%), with ApproxRRset more accurate for smaller seed sets ($k \leq 100$) and ApproxPermuteDirect superior thereafter. (3) ApproxPermuteMC performs poorly, peaking at 68.3% error. On the sparser Twitter network, ApproxPermuteDirect consistently outperforms ApproxRRset with error at most 1.4%, indicating better scalability with seed count in sparse graphs.

Varing number of seeds under Single-step termination: As shown in Figure 6, ExactSingleStepOpt obtains the exact Shapley values on both networks, confirming our theoretical results. Approximation algorithms show similar trends as in Complete termination. ApproxPermuteDirect is most accurate with relative error at most 2.5% on Facebook and 0.1% on Twitter.

Varing network structure under Complete termination: We vary the number of nodes, average degree, and edge probability in synthetic graphs as shown in Figure 7 and observe that: (1) Across varying network sizes (Fig. 7a) and average degrees (Fig. 7b), ApproxRRset consistently delivers the highest accuracy with relative error below 1.3%, followed by ApproxPermuteMC (up to 2.1%), while ApproxPermuteDirect is the least accurate (up to 4.5%). (2) The number of nodes and the average degree have a slight effect on the relative error of each approximation method. (3) However, edge probabilities significantly affect performance(Fig. 7c). ApproxRRset is sensitive to low edge probabilities, reaching 6.1% error at $p = 0.01$ due to high variance in RR-set generation, but drops below 0.1% when $p \geq 0.2$. For other methods, ApproxPermuteMC performs best at low probabilities (≤ 0.1) with error at most 2%, while ApproxPermuteDirect excels at higher probabilities (≥ 0.2) with error at most 1.5%.

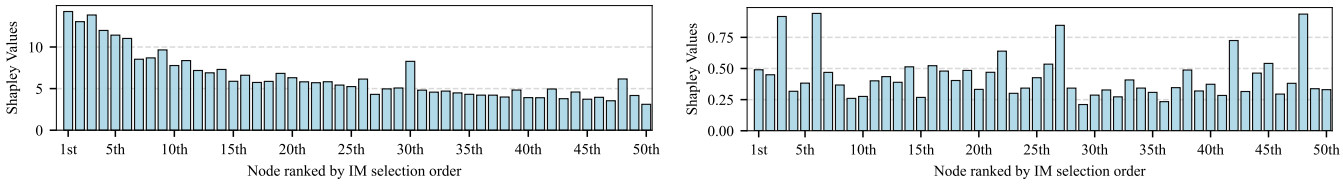
Varing network structure under Single-step termination: Under the Single-step termination (Figure 8), ExactSingleStepOpt computes exact Shapley values. The performance of the approximation algorithms is inverted compared to the

Table 2: Top 10 authors in the DBLP Coauthor network by out-degree and their corresponding Shapley values

DBLP Authors		
Author name	**Rank (out-degree)**	**Rank (Shapley value)**
Jingyan Zhou0001	1 (59)	3 (13.46)
Feifei Li 0001	2 (47)	6 (9.85)
Christian S. Jensen	3 (46)	**1 (15.30)**
Guoliang Li 0001	4 (45)	**2 (14.97)**
Wei Zhang	5 (43)	9 (8.28)
Nan Tang 0001	6 (41)	8 (8.66)
Bingsheng He	7 (39)	**5 (11.04)**
Xiaoyong Du 0001	8 (38)	7 (9.77)
Lei Chen 0002	9 (37)	**4 (12.15)**
Lei Cao 0004	10 (36)	10 (7.20)

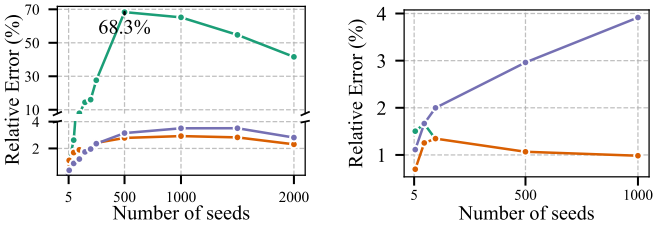
Table 3: Top 10 members in the Congressional Twitter network by out-degree and their corresponding Shapley values

Congress Members		
Username	**Rank (out-degree)**	**Rank (Shapley Value)**
SpeakerPelosi	1 (210)	2 (1.00)
GOPLeader	2 (157)	3 (0.95)
RepBobbyRush	3 (111)	4 (0.94)
SenSchumer	4 (97)	**7 (0.38)**
SteveScalise	5 (89)	**1 (1.04)**
RepMarkTakano	6 (85)	5 (0.67)
rosadelauro	7 (84)	6 (0.46)
LeaderHoyer	8 (79)	**10 (0.24)**
RepJimBanks	9 (75)	9 (0.25)
SenWarren	10 (71)	8 (0.34)



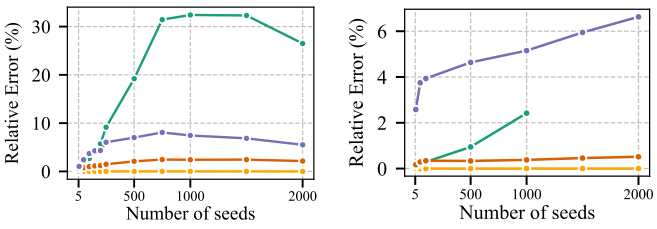
(a) DBLP Coauthor network **(b) Congressional Twitter network**

Figure 4: Shapley values of seed nodes selected by greedy IM algorithm in the DBLP Coauthor and Congressional Twitter networks. Higher ranks indicate earlier selection in the greedy IM algorithm.



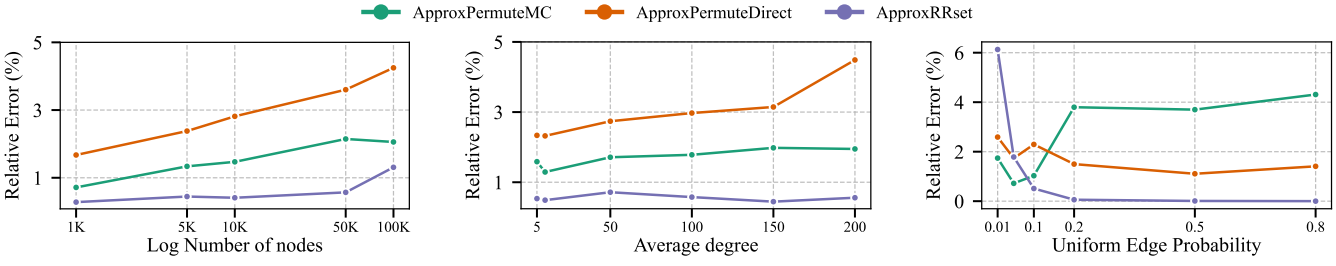
(a) Facebook **(b) Twitter**

Figure 5: Relative error vs. # seeds in Complete termination



(a) Facebook **(b) Twitter**

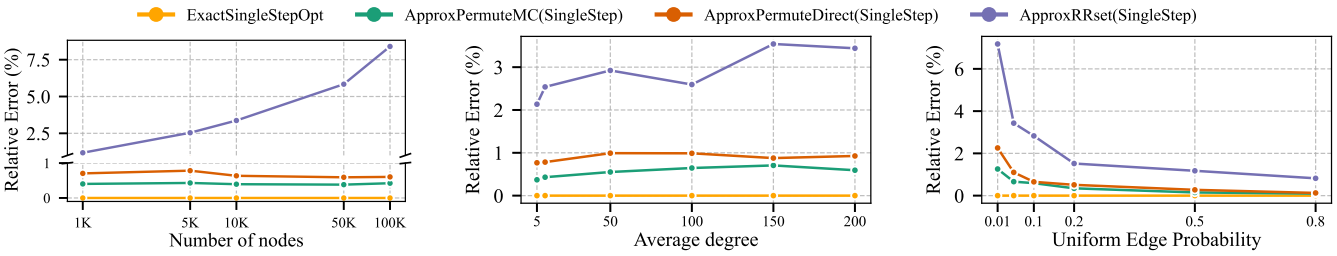
Figure 6: relative error vs. # seeds in Single-step termination



(a) Varying number of nodes **(b) Varying average degree** **(c) Varying edge probability**

Figure 7: Relative error vs. network structure in Complete termination (default: avg. degree = 10, 10 seed nodes, 5k nodes)

Complete termination: ApproxRRset becomes least accurate under all settings, for example, the error rises from 1.1% on 1K-node graphs to 8.4% on 100K-node graphs in Fig. 8a. In contrast, both ApproxPermuteDirect and ApproxPermuteMC are highly accurate in most settings, with relative error consistently below 1% (except when $p < 0.1$). This accuracy difference is due to that ApproxRRset relies on reverse reachability, thus each RR-set contains only the start node and its immediate seed predecessors. Therefore,



(a) Varying number of nodes **(b) Varying average degree** **(c) Varying edge probability**

Figure 8: Relative error vs. network structure in Single-step termination (default: avg. degree = 10, 10 seed nodes, 5k nodes)

each RR set carries little information, leading to a high-variance estimator. Conversely, permutation-based algorithms directly evaluate marginal contributions on forward traversals starting from each seed, better capturing the influence in Single-step.

6.4 Runtime Analysis

In this section we analyze the runtime performance and scalability of our proposed algorithms. Table 4 presents the runtime as the number of seeds increases on real-world datasets ranging from the 4K-node Facebook network to the 4.8M-node LiveJournal network. Table 5 examines how the runtime varies with the network structure parameters. Dashes (–) indicate runtime exceeding the 12-hour limit for that setting.

Varying Number of Seeds: First, Table 4(a) shows the runtime of approximation algorithms under Complete termination model. ApproxRRset demonstrates superior scalability, and the runtime even decreases as the number of seeds grows. For instance, on the Orkut network, its runtime drops from 131.431 seconds for 10 seeds to 41.253 seconds for 1M seeds. This behavior is because we sample a fixed number of random RR sets on the modified graph G', obtained by removing all incoming edges to the seed nodes. As the number of seeds grows, G' becomes sparser, which reduces the average cost of the reverse graph traversals required to generate each RR set. In contrast, the runtime of ApproxPermuteDirect grows with the number of seeds, since for each sampled permutation, the algorithm computes the marginal contribution for every seed by a graph traversal (e.g., BFS). While ApproxPermuteDirect is competitive for small networks (e.g., 0.782 seconds on Facebook with $2,000$ seeds), it becomes inefficient for larger networks, taking 948.635 seconds for only 10 seeds on LiveJournal and exceeds 12 hours for 100K seeds. Finally, the runtime of ApproxPermuteMC increases dramatically with the number of seeds, and it times out for even 100 seeds on both LiveJournal and Orkut networks. It employs a nested sampling design that requires numerous costly Monte Carlo simulations to estimate a single marginal contribution, making it impractical for most scenarios.

Then Table 4(b) shows the runtime under the Single-step termination model. Approximation algorithms exhibit similar trends as under Complete termination, with ApproxRRset being the most scalable. Besides, ExactSingleStepOpt computes the exact Shapley values more efficient and scalable. For instance, on the 4.8M-node LiveJournal network, it computes exact values for 1M seeds in 403.847 seconds, and on the dense Orkut network with 3.1M nodes, the algorithm completes in only 26.546 seconds, making it highly practical even for large-scale networks.

Varying Network Structure: Under the Complete model as shown in Table 5(a), we note that: (1) As the network size grows, ApproxRRset is most scalable, within 15 seconds for 1M-node graphs, and ApproxPermuteDirect is also efficient copmapred to ApproxPermuteMC, with 53 seconds for 1M-node graphs. (2) When varying the average degree from 5 to 200, the runtime of ApproxRRset increases substantially from 0.327 to 376.356 seconds, whereas the runtime of ApproxPermuteDirect only modestly increases from 0.116 to 1.506 seconds, showing a remarkable efficiency. (3) A similar trend holds for edge probability, where ApproxPermuteDirect maintains its efficiency advantage, outperforming other methods by orders of magnitude on graphs with higher edge probabilities.

For the Single-step model in Table 5(b), ExactSingleStepOpt is exceptionally scalable, computing exact Shapley values with negligible runtime (milliseconds) across all settings. In contrast, the approximation algorithms are all substantially slower, though they follow similar performance trends as in the Complete model.

7 RELATED WORK

Influence Maximization. Much of the early work on influence diffusion in social networks has focused on the *influence maximization* problem [16, 30], which seeks to find a minimal set of influential nodes (i.e., seeds) that maximizes the expected spread of influence under stochastic diffusion models. Since this optimization problem is generally NP-hard, a variety of efficient heuristic and approximation algorithms have been proposed [5, 11, 28, 31–33, 47, 48, 52]. Notably, the reverse influence sampling techniques [5, 47, 48] significantly improves the scalability. Shapley values have also been applied to the influence maximization problem. Narayanam and Narahari [42] proposed Shapley-value-based methods for discovering influential nodes under the Linear Threshold model, demonstrating sampling-based approahces for approximation computation. Nevertheless, these work focus on prospective seed selection, and do not address the problem of retrospective equitable attribution among the given seeds. **Network Centrality and Influence Attribution.** Another related stream of research investigates *network centrality measures* [4, 6, 10, 20, 23, 40, 42, 43], which aim to identify the structural importance of nodes in a network. Michalak et al. [40] applied the Shapley value to measure the network centrality, but they did not capture the influence diffusion process. Chen and

Table 4: Runtime (seconds) vs. number of seeds

(a) Under Complete termination												
Algorithm / Dataset	**Facebook**						**Twitter**					
	10	**50**	**100**	**1,000**	**1,500**	**2,000**	**10**	**50**	**100**	**1,000**	**2,000**	**5,000**
ApproxRRset	0.615	0.365	0.317	0.147	0.100	0.078	1.365	1.182	1.050	0.628	0.517	0.397
ApproxPermuteDirect	0.341	0.292	0.294	0.458	0.617	0.782	2.601	2.553	2.600	11.942	41.963	162.243
ApproxPermuteMC	19.176	585.590	1322.521	15,236.833	20,960.374	24,508.466	79.960	3,329.521	8,022.476			
Algorithm/Dataset	**LiveJournal**						**Orkut**					
	10	**100**	**1K**	**10K**	**100K**	**1M**	**10**	**100**	**1K**	**10K**	**100K**	**1M**
ApproxRRset	46.021	45.507	45.645	39.484	32.897	13.802	131.431	123.657	115.945	109.828	89.542	41.253
ApproxPermuteDirect	948.635	985.993	1450.862	16,589.004	-	-	887.629	903.666	921.242	2003.664	-	-
ApproxPermuteMC	1281.045	-	-	-	-	-	11,930.289	-	-	-	-	-
(b) Under Single-step termination												
Algorithm / Dataset	**Facebook**						**Twitter**					
	10	**50**	**100**	**1,000**	**1,500**	**2,000**	**10**	**50**	**100**	**1,000**	**2,000**	**5,000**
ExactSingleStepOpt	0.004	0.006	0.009	0.013	0.013	0.014	0.007	0.011	0.016	0.079	0.142	0.484
ApproxRRset	0.126	0.123	0.121	0.081	0.073	0.070	0.576	0.450	0.446	0.448	0.398	0.345
ApproxPermuteDirect	0.347	0.321	0.264	0.469	0.627	0.767	3.019	2.959	2.989	4.005	5.542	78.753
ApproxPermuteMC	2.655	125.446	437.687	10,872.039	16,714.828	21,444.611	5.319	280.600	922.846	-	–	–
Algorithm / Dataset	**LiveJournal**						**Orkut**					
	10	**100**	**1K**	**10K**	**100K**	**1M**	**10**	**100**	**1K**	**10K**	**100K**	**1M**
ExactSingleStepOpt	0.033	0.068	0.252	2.170	13.719	403.847	0.101	0.375	1.455	5.749	14.211	26.546
ApproxRRset	46.780	46.668	47.146	45.349	34.677	16.952	110.087	110.525	109.582	105.859	90.092	41.415
ApproxPermuteDirect	521.528	622.566	508.305	2,145.363	-	-	894.402	885.322	896.673	910.608	-	-
ApproxPermuteMC	49.245	4,592.678	-	-	-	-	170.480	41,099.680	-	-	-	-

Table 5: Runtime (seconds) vs. network structure parameters

(b) Synthetic graph under Complete termination												
Algorithm /	**Number of Nodes**				**Average Degree**				**Edge Probability**			
Network Parameter	**1K**	**10K**	**100K**	**1M**	**5**	**50**	**100**	**200**	**0.01**	**0.1**	**0.5**	**0.8**
ApproxRRset	0.153	0.368	2.197	14.459	0.327	64.346	145.788	376.356	0.027	0.328	27.197	23.465
ApproxPermuteDirect	0.026	0.185	2.891	52.667	0.116	0.282	0.535	1.506	0.056	0.105	0.350	0.407
ApproxPermuteMC	1.224	4.855	20.330	80.085	2.184	11.686	22.048	43.562	0.131	3.642	33.473	27.454
(a) Synthetic graph under Single-step termination												
Algorithm /	**Number of Nodes**				**Average Degree**				**Edge Probability**			
Network Parameter	**1K**	**10K**	**100K**	**1M**	**5**	**50**	**100**	**200**	**0.01**	**0.1**	**0.5**	**0.8**
ExactSingleStepOpt	0.002	0.002	0.002	0.002	0.002	0.002	0.003	0.003	0.002	0.002	0.002	0.002
ApproxRRset	0.027	0.054	0.321	7.289	0.019	0.144	0.286	0.737	0.033	0.034	0.042	0.048
ApproxPermuteDirect	0.035	0.187	2.884	50.442	0.093	0.278	0.543	1.544	0.056	0.100	0.215	0.240
ApproxPermuteMC	0.077	0.099	0.128	0.234	0.076	0.300	0.568	1.359	0.076	0.094	0.159	0.193

Teng [10] proposed Shapley Centrality, capturing the diffusion potential of each node. However, these methods focus on ranking nodes by their importance in the network structure, rather than addressing the fair allocation of influence among given seed nodes that jointly contribute to an influence diffusion process. Related to our work, Zhu et al. [56] examined *influence contribution allocation* for advertising campaigns under deterministic influence diffusion, and computed Shapley values when the complete diffusion outcome is observed. This approach represents a post-hoc analysis requireing full observability of the diffusion process, which severely limits its applicability in real-world scenarios where tracking every activation is impractical or impossible. Our research studies a different and more general scenario: we compute Shapley values based on the *expected* influence rather than constraint of a specific influence diffusion outcome. **Shapley Values.** Shapley values have been used in databases [14, 15, 29, 36, 37, 44] and machine learning [1, 21, 22, 38, 51, 54]. Research includes study on complexity and poly-time algorithms in special cases with additional structural properties, and various approximation methods including Monte Carlo sampling [9, 27, 41] and stratified sampling [7, 8, 39, 55] as practical estimations.

8 CONCLUSIONS

In this paper, we studied the problem of quantifying the contribution of individual seed nodes for influence propagation in networks using Shapley values. We presented exact algorithms for special cases, #P-hardness, approximation algorithms, and experiments with synthetic and real datasets. Future research includes extending to other diffusion models such as the Linear Threshold model, incorporating dynamic networks with temporal changes, developing robust methods under incomplete network information, and investigating advanced sampling strategies to further improve scalability and accuracy.

REFERENCES

[1] Marcelo Arenas, Pablo Barceló, Leopoldo Bertossi, and Mikaël Monet. 2021. The tractability of SHAP-score-based explanations for classification over deterministic and decomposable boolean circuits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 6670–6678.

[2] Roland Bacher. 2002. Determinants of matrices related to the Pascal triangle. *Journal de théorie des nombres de Bordeaux* 14, 1 (2002), 19–41.

[3] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. 2006. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 44–54.

[4] Phillip Bonacich. 1972. Factoring and weighting approaches to status scores and clique identification. *Journal of mathematical sociology* 2, 1 (1972), 113–120.

[5] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. 2014. Maximizing social influence in nearly optimal time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 946–957.

[6] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30, 1-7 (1998), 107–117.

[7] Mark Alexander Burgess and Archie C Chapman. 2021. Approximating the Shapley Value Using Stratified Empirical Bernstein Sampling.. In *IJCAI*. 73–81.

[8] Javier Castro, Daniel Gómez, Elisenda Molina, and Juan Tejada. 2017. Improving polynomial estimation of the Shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research* 82 (2017), 180–188.

[9] Javier Castro, Daniel Gómez, and Juan Tejada. 2009. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research* 36, 5 (2009), 1726–1730.

[10] Wei Chen and Shang-Hua Teng. 2017. Interplay between social influence and network centrality: A comparative study on shapley centrality and single-node-influence centrality. In *Proceedings of the 26th international conference on world wide web*. 967–976.

[11] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1029–1038.

[12] Fan Chung and Linyuan Lu. 2006. Concentration inequalities and martingale inequalities: a survey. *Internet mathematics* 3, 1 (2006), 79–127.

[13] dblp Team. 2025. dblp computer science bibliography – Monthly Snapshot XML Release of April 2025. https://doi.org/10.4230/dblp.xml.2025-04-01

[14] Daniel Deutch, Nave Frost, Amir Gilad, and Oren Sheffer. 2021. Explanations for data repair through shapley values. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 362–371.

[15] Daniel Deutch, Nave Frost, Benny Kimelfeld, and Mikaël Monet. 2022. Computing the Shapley Value of Facts in Query Answering. In *Proceedings of the 2022 International Conference on Management of Data*. 1570–1583.

[16] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 57–66.

[17] P ERDdS and A R&wi. 1959. On random graphs I. *Publ. math. debrecen* 6, 290-297 (1959), 18.

[18] Christian G Fink, Kelly Fullin, Guillermo Gutierrez, Nathan Omodt, Sydney Zinnecker, Gina Sprint, and Sean McCulloch. 2023. A centrality measure for quantifying spread on weighted, directed networks. *Physica A: Statistical Mechanics and its Applications* 626 (2023), 129083.

[19] Christian G Fink, Nathan Omodt, Sydney Zinnecker, and Gina Sprint. 2023. A Congressional Twitter network dataset quantifying pairwise probability of influence. *Data in Brief* 50 (2023), 109521.

[20] Linton C Freeman et al. 2002. Centrality in social networks: Conceptual clarification. *Social network: critical concepts in sociology. Londres: Routledge* 1 (2002), 238–263.

[21] Daniel Fryer, Inga Strümke, and Hien Nguyen. 2021. Shapley values for feature selection: The good, the bad, and the axioms. *Ieee Access* 9 (2021), 144352–144360.

[22] Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*. PMLR, 2242–2251.

[23] Rumi Ghosh, Shang-hua Teng, Kristina Lerman, and Xiaoran Yan. 2014. The interplay between dynamics and networks: centrality, communities, and cheeger inequality. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1406–1415.

[24] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. 2011. A data-based approach to social influence maximization. *arXiv preprint arXiv:1109.6886* (2011).

[25] Aric Hagberg, Pieter J Swart, and Daniel A Schult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Laboratory (LANL), Los Alamos, NM (United States).

[26] Wassily Hoeffding. 1994. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding* (1994), 409–426.

[27] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. 2019. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1167–1176.

[28] Kyomin Jung, Wooram Heo, and Wei Chen. 2012. Irie: Scalable and robust influence maximization in social networks. In *2012 IEEE 12th international conference on data mining*. IEEE, 918–923.

[29] Ahmet Kara, Dan Olteanu, and Dan Suciu. 2024. From Shapley Value to Model Counting and Back. *Proceedings of the ACM on Management of Data* 2, 2 (2024), 1–23.

[30] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. 137–146.

[31] Jinha Kim, Seung-Keol Kim, and Hwanjo Yu. 2013. Scalable and parallelizable processing of influence maximization for large-scale social networks?. In *2013 IEEE 29th international conference on data engineering (ICDE)*. IEEE, 266–277.

[32] Masahiro Kimura and Kazumi Saito. 2006. Tractable models for information diffusion in social networks. In *European conference on principles of data mining and knowledge discovery*. Springer, 259–271.

[33] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 420–429.

[34] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.

[35] Jure Leskovec and Julian Mcauley. 2012. Learning to discover social circles in ego networks. *Advances in neural information processing systems* 25 (2012).

[36] Ester Livshits, Leopoldo Bertossi, Benny Kimelfeld, and Moshe Sebag. 2021. The Shapley value of tuples in query answering. *Logical Methods in Computer Science* 17 (2021).

[37] Ester Livshits and Benny Kimelfeld. 2022. The Shapley value of inconsistency measures for functional dependencies. *Logical Methods in Computer Science* 18 (2022).

[38] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[39] Sasan Maleki, Long Tran-Thanh, Greg Hines, Talal Rahwan, and Alex Rogers. 2013. Bounding the estimation error of sampling-based Shapley value approximation. *arXiv preprint arXiv:1306.4265* (2013).

[40] Tomasz P Michalak, Karthik V Aadithya, Piotr L Szczepanski, Balaraman Ravindran, and Nicholas R Jennings. 2013. Efficient computation of the Shapley value for game-theoretic network centrality. *Journal of Artificial Intelligence Research* 46 (2013), 607–650.

[41] Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. 2022. Sampling permutations for shapley value estimation. *Journal of Machine Learning Research* 23, 43 (2022), 1–46.

[42] Ramasuri Narayanam and Yadati Narahari. 2011. A Shapley Value-Based Approach to Discover Influential Nodes in Social Networks. *IEEE Transactions on Automation Science and Engineering* 8, 1 (2011), 130–147. https://doi.org/10.1109/TASE.2010.2052042

[43] Mahendra Piraveenan, Mikhail Prokopenko, and Liaquat Hossain. 2013. Percolation centrality: Quantifying graph-theoretic impact of nodes during percolation in networks. *PloS one* 8, 1 (2013), e53095.

[44] Alon Reshef, Benny Kimelfeld, and Ester Livshits. 2020. The impact of negation on the complexity of the Shapley value in conjunctive queries. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 285–297.

[45] Lloyd S Shapley. 1953. A value for n-person games. *Contribution to the Theory of Games* 2 (1953).

[46] Fangzhu Shen, Amir Gilad, and Sudeepa Roy. [n.d.]. Shapley Values for Measuring Contributions in Influence Propagation in a Network. https://github.com/fangzhushen/Shapley-and-Influence-Diffusion.

[47] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1539–1554.

[48] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 75–86.

[49] Leslie G. Valiant. 1979. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.* 8 (Aug. 1979), 410–421. https://doi.org/10.1137/0208032

[50] Leslie G. Valiant. 1979. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.* 8, 3 (1979), 410–421. https://doi.org/10.1137/0208032 arXiv:https://doi.org/10.1137/0208032

[51] Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. 2022. On the tractability of SHAP explanations. *Journal of Artificial Intelligence Research* 74 (2022), 851–886.

[52] Yu Wang, Gao Cong, Guojie Song, and Kunqing Xie. 2010. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1039–1048.

[53] Jaewon Yang and Jure Leskovec. 2012. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*. 1–8.
[54] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*. PMLR, 12241–12252.
[55] Jiayao Zhang, Qiheng Sun, Jinfei Liu, Li Xiong, Jian Pei, and Kui Ren. 2023. Efficient sampling approaches to shapley value approximation. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–24.
[56] Yuqing Zhu, Jing Tang, Xueyan Tang, and Lei Chen. 2021. Analysis of influence contribution in social advertising. *Proceedings of the VLDB Endowment* 15, 2 (2021), 348–360.

A APPENDIX FOR SECTION 2

Relation to influence functions in prior work. Prior work in influence maximization typically optimizes the *influence spread* function $\sigma(S)$, defined as the expected total number of activated nodes given a seed set S [30]. In contrast, the *value function* $U_{G,T}(S)$ introduced in this paper quantifies the influence specifically originating from a subset S within the context of a given seed set T. Formally, the standard influence spread is defined as follows:

Definition 4 (Influence Spread $\sigma_G(S)$ [30]). *Given a graph $G = (V, E)$ and activation probabilities p, the influence spread of a set $S \subseteq V$ is the expected number of activated nodes at the end of the Independent Cascade (IC) process, assuming S is the entire seed set.*

Key distinctions between $U_{G,T}(S)$ and $\sigma_G(S)$ are: (i) **Seed Set Context**: $U_{G,T}(S)$ evaluates influence within a fixed seed set T, isolating the contribution of subset $S \subseteq T$; while $\sigma_G(S)$ assumes S is the entire seed set. (ii) **Node Counting**: $U_{G,T}(S)$ counts only activated nodes in $V \setminus T$ exclusively influenced by S, whereas $\sigma_G(S)$ includes all activated nodes, including those in S itself.

Despite these differences, the two functions are closely related, as established by the following propositions:

Proposition 5. *Given a graph $G = (V, E)$ with activation probabilities $p_{u,v}$ and seed set T, construct a modified graph $G' = (V, E')$ by removing all incoming edges to seed nodes, i.e., $E' = E \setminus (u, v) | v \in T$. Then, for any subset $S \subseteq T$:*

$$U_{G,T}(S) = \sigma_{G'}(S) - |S|. \tag{18}$$

Proof. First, note that $U_{G,T}(S)$ measures activated nodes in $V \setminus T$ influenced exclusively by S. In graph G', seed nodes have no incoming edges, preventing external activation. Thus, influence from S in both G and G' is identical over nodes in $V \setminus T$. Therefore, $U_{G,T}(S) = U_{G',T}(S)$. Moreover, since $\sigma_{G'}(S)$ includes the nodes in S, we have $U_{G',T}(S) = \sigma_{G'}(S) - |S|$. □

Proposition 6. *Given a graph $G = (V, E)$ with activation probabilities $p_{u,v}$, for any $S \subseteq V$:*

$$\sigma_G(S) = U_{G,S}(S) + |S|. \tag{19}$$

Proof. By definition, $U_{G,S}(S)$ is the expected number of nodes outside S activated solely by S when S itself is the seed set. Thus, $\sigma_G(S)$ equals nodes activated outside S plus nodes in S: $\sigma_G(S) = U_{G,S}(S) + |S|$. □

Computational Complexity of the Value Function. Computing $\sigma(S)$ is known to be P-hard [11]. This hardness directly extends to computing $U_{G,T}(S)$:

Proposition 7. *Computing the value function $U_{G,T}(S)$ under the full-step IC model is P-hard.*

Proof. We prove via Turing reduction from computing $\sigma_G(S)$. Given a graph G, construct the instance $U_{G,S}(S)$ by setting $T = S$. Using Proposition 6, we obtain $\sigma_G(S) = U_{G,S}(S) + |S|$. Hence, an oracle computing $U_{G,S}(S)$ in polynomial time would compute $\sigma_G(S)$ in polynomial time, contradicting the known P-hardness of computing $\sigma_G(S)$. □

B APPENDIX FOR SECTION 3

Theorem 3.1. *Given $(G(V, E), T, p)$, the Shapley value $Shap(t)$ for every seed node $t \in T$ under Single-step termination can be computed in polynomial time in $|V|$.*

Lemma 7. *In the Single-step activation model, the marginal contribution of adding node $t \in T \setminus S$ to any coalition $S \subseteq T$ is:*

$$U(S \cup \{t\}) - U(S) = \sum_{u \in \{V \setminus T\} \cap N^+(\{t\})} \left[p_{t,u} \cdot \prod_{w \in N^-(u) \cap S} (1 - p_{w,u}) \right] \tag{20}$$

Proof. First, since we only consdier Single-step activation, by the Definition 1 that $U(S)$ is the expected number of non-seed nodes activated solely by paths originating from S, it can be rewritten as the sum of the probability that each non-seed node u is activated by coalition S at step 1:

$$U(S) = \sum_{u \in V \setminus T} \Pr[u \in A_1 \mid \text{paths originating from } S]. \tag{21}$$

Since each active node independently attempts to activate its neighbors, the activation probability of each non-seed node under the Single-step activation equals to 1 minus the probability that being failed to be activated by all of its neighbors in S:

$$\begin{aligned} &\Pr[u \in A_1 \mid \text{paths originating from } S] \\ =&1 - \prod_{w \in N^-(u)} (1 - p_{u,w} \cdot \mathbb{1}[w \in S]) \\ =&1 - \prod_{w \in N^-(u) \cap S} (1 - p_{u,w}). \end{aligned} \tag{22}$$

Moreover, only non-seed nodes with incoming edges from seed nodes within S have the chance to be activated. Thus, equation (21) can be simplified as:

$$U(S) = \sum_{u \in \{V \setminus T\} \cap N^+(S)} (1 - \prod_{w \in N^-(u) \cap S} (1 - p_{u,w})). \tag{23}$$

Then we substitute Eq. (23) into the formula of marginal contribution for node v to S as follows:

$$\begin{aligned} &U(S \cup \{v\}) - U(S) \\ =& \sum_{u \in \{V \setminus T\} \cap N^+(S \cup \{v\})} (1 - \prod_{w \in N^-(u) \cap (S \cup \{v\})} (1 - p_{w,u})) \\ &- \sum_{u \in \{V \setminus T\} \cap N^+(S)} (1 - \prod_{w \in N^-(u) \cap S} (1 - p_{w,u})) \\ =& \sum_{u \in \{V \setminus T\} \cap N^+(\{v\})} (\prod_{w \in N^-(u) \cap S} (1 - p_{w,u}) - \prod_{w \in N^-(u) \cap (S \cup \{v\})} (1 - p_{w,u})) \\ &+ \sum_{u \in \{V \setminus T\} \cap N^+(S) \setminus N^+(\{v\})} (\prod_{w \in N^-(u) \cap S} (1 - p_{w,u}) - \prod_{w \in N^-(u) \cap (S \cup \{v\})} (1 - p_{w,u})) \end{aligned} \tag{24}$$

In the above Eq. (24), we divide the marginal contribution into two disjoint summation terms. In the first term, we consider $u \in \{V \setminus T\} \cap N^+(\{v\})$, which are the non-seed out-neighbor nodes of v, so we have

$$\prod_{w \in N^-(u) \cap (S \cup \{v\})} (1-p_{w,u}) = \left[\prod_{w \in N^-(u) \cap S} (1-p_{w,u}) \right] \cdot (1-p_{v,u}), \quad (25)$$

which follows the fact that adding v to the set of u's seed neighbors introduces the term $(1-p_{v,u})$ to the product.

In the second term, we consider $u \in \{V \setminus T\} \cap N^+(S) \setminus N^+(\{v\})$, which are non-seed nodes that are out-neighbor of seed nodes within S but are not v's out-neighbor, so we have

$$\prod_{w \in N^-(u) \cap (S \cup \{v\})} (1-p_{w,u}) = \prod_{w \in N^-(u) \cap S} (1-p_{w,u}), \quad (26)$$

Therefore, the two products are identical, causing the second term to cancel out.

Simplifying Eq. (24), we conclude that

$$U(S \cup \{v\}) - U(S) = \sum_{u \in \{V \setminus T\} \cap N^+(\{v\})} \left[p_{v,u} \cdot \prod_{w \in N^-(u) \cap S} (1-p_{w,u}) \right] \quad (27)$$

Note that $p_{u,v} \in (0,1]$ for all $(u,v) \in E$, all terms in the summation are non-negative, we have $U(S \cup \{v\}) - U(S) \geq 0$. Therefore, $U(S)$ is monotonically non-decreasing. □

Lemma 1. *For a seed node $t \in T$, under Single-step termination,*

$$Shap(t) = \sum_{k=0}^{|T|-1} \frac{k!\,(|T|-k-1)!}{|T|!} \cdot \left[\sum_{u \in \{V \setminus T\} \cap N^+(\{t\})} p_{t,u} \cdot \alpha_{u,t}(T \setminus \{t\}, k) \right] \quad (6)$$

$$\text{where} \quad \alpha_{u,t}(T \setminus \{t\}, k) = \sum_{\substack{S \subseteq T \setminus \{t\} \\ |S|=k}} \left(\prod_{w \in N^-(u) \cap S} (1-p_{w,u}) \right). \quad (7)$$

Proof. Let k denote the size of a subset S. By the definition of Shapley value in Definition 2:

$$\begin{aligned} & Shap(t) \\ &= \sum_{S \subseteq T \setminus \{t\}} \frac{|S|!\,(|T|-|S|-1)!}{|T|!} \cdot [U(S \cup \{t\}) - U(S)] \\ &= \sum_{k=0}^{|T|-1} \sum_{\substack{S \subseteq T \setminus t \\ |S|=k}} \frac{k!\,(|T|-k-1)!}{|T|!} \cdot [U(S \cup \{t\}) - U(S)] \quad (28) \\ &= \sum_{k=0}^{|T|-1} \frac{k!\,(|T|-k-1)!}{|T|!} \cdot \Big(\sum_{\substack{S \subseteq T \setminus t \\ |S|=k}} [U(S \cup \{t\}) - U(S)] \Big) \end{aligned}$$

Substituting the marginal contribution from Lemma 7 and switch the order of summation, we obtain:

$$\begin{aligned} & Shap(t) \\ &= \sum_{k=0}^{|T|-1} \frac{k!\,(|T|-k-1)!}{|T|!} \\ &\quad \cdot \left[\sum_{\substack{S \subseteq T \setminus t \\ |S|=k}} \sum_{u \in \{V \setminus T\} \cap N^+(\{t\})} [p_{t,u} \cdot \prod_{w \in N^-(u) \cap S} (1-p_{w,u})] \right] \quad (29) \\ &= \sum_{k=0}^{|T|-1} \frac{k!\,(|T|-k-1)!}{|T|!} \\ &\quad \cdot \left[\sum_{u \in \{V \setminus T\} \cap N^+(\{t\})} p_{t,u} \cdot \sum_{\substack{S \subseteq T \setminus t \\ |S|=k}} [\prod_{w \in N^-(u) \cap S} (1-p_{w,u})] \right] \end{aligned}$$

□

Lemma 2. *Given a seed node $t \in T$, one of its non-seed out-neighbor $u \in \{V \setminus T\} \cap N^+(\{t\})$ and an integer $k \in \{1, \ldots, |T|-1\}$, the following holds for any subset $L \subseteq T \setminus \{t\}$:*

$$\alpha_{u,t}(L,k) = \begin{cases} 1 & \text{if } k = 0 \\ (1-p_{n,u}) \cdot \alpha_{u,t}(L \setminus \{n\}, k-1) & \\ \quad + \alpha_{u,t}(L \setminus \{n\}, k) & \forall n \in L, \text{if } 1 \leq k \leq |L| \\ 0 & \text{otherwise,} \end{cases} \quad (8)$$

where $\alpha_{u,t}(L,k) = \sum_{\substack{S \subseteq L \\ |S|=k}} \left(\prod_{w \in N^-(u) \cap S} (1-p_{w,u}) \right)$.

Proof. We partition the subsets of L of size k into those that include n and those that do not.

For the sums, we have following observations:

- $k \in \{0, 1, \cdots, |L|\}$, limited by the size of L. If $k > |L|$, we set the value to 0 as it is meaningless.
- When $k = 0$, then subset $S = \emptyset$ for any L. Thus $\sum_{\substack{S \subseteq L \\ |S|=0}} [\prod_{w \in N^-(u) \cap S} (1-p_{w,u})] = 1$ for $\forall L \subseteq T \setminus \{t\}$.
- When $k = 1$ and $|L| = 1$, for $\forall v' \in T \setminus \{t\}$ and $L = \{v'\}$, $\sum_{\substack{S \subseteq \{v'\} \\ |S|=1}} [\prod_{w \in N^-(u) \cap S} (1-p_{w,u})] = \prod_{w \in N^-(u) \cap \{v'\}} (1-p_{w,u}) = (1-p_{v',u})$, which is the base case.
- When $|L| \geq 2$, for all $k = 1, \cdots, |L|$, the sum over possible subsets S can be decomposed as two parts: subsets S including n and subsets S not including n, where n denotes an arbitrary node in L.

Thus the decomposition is summarized as below. First, for $|L| = 1$ and $k = 1$, denote $L = \{v'\}$ without loss of generality. The summation can also be divided into two parts: a subset including the only node v' and an empty subset. Then the decomposition equation holds:

$$\begin{aligned} & \sum_{\substack{S \subseteq \{v'\} \\ |S|=1}} [\prod_{w \in N^-(u) \cap S} (1-p_{w,u})] \\ &= [\sum_{\substack{S \subseteq \emptyset \\ |S|=0}} 1 \cdot (1-p_{v',u})] + \sum_{\substack{S \subseteq \emptyset \\ |S|=1}} [\prod_{w \in N^-(u) \cap S} (1-p_{w,u})] \quad (30) \\ &= (1-p_{v',u}) \end{aligned}$$

Then for $|L| \geq 2$, the decomposition following the observation is:

$$\begin{aligned}
&\sum_{\substack{S\subseteq L\\|S|=k}} [\prod_{w\in N^-(u)\cap S}(1-p_{w,u})] \\
&= \sum_{\substack{S\subseteq L\setminus\{n\}\\|S|=k-1}} [\prod_{w\in N^-(u)\cap S}(1-p_{w,u})\cdot(1-p_{n,u})] \\
&+ \sum_{\substack{S\subseteq L\setminus\{n\}\\|S|=k}} [\prod_{w\in N^-(u)\cap S}(1-p_{w,u})]
\end{aligned} \tag{31}$$

Moving the term $1-p_{n,u}$ outside the summation, then we get the decomposition equation.

Note that when $k = |L|$, the latter sum is meaningless so it is set to 0. It means that if $k = |L|$, every subset S consist the arbitrary node n. □

Lemma 8. *Given any seed node $t \in T$, any of its non-seed out-neighbor $u \in \{V \setminus T\} \cap N^+(t)$, and any set $L \subseteq T \setminus \{t\}$ of size $|L|$, all values $\alpha_{u,t}(L,k)$ for $k \in \{0,\ldots,|L|\}$ can be computed in $O(|L|^2)$ time.*

Proof. Using Lemma 2, each $\alpha_{u,t}(L,k)$ can be decomposed recursively and the process can be represented as a recursion tree. Let $L = \{n_1, n_2, \ldots, n_{|L|}\}$ be an arbitrary ordering of the elements in L. We can compute the values $\alpha_{u,t}(L_i,k)$ for $L_i = \{n_1,\ldots,n_i\}$ and $k \in \{0,\ldots,i\}$ iteratively for $i = 0,\ldots,|L|$.

For a given i,$\alpha_{u,t}(L_i,k)$ for all $k \in \{0,\ldots,i\}$ can be computed using the values computed for L_{i-1} by Lemma 2:

$$\alpha_{u,t}(L_i,k) = \begin{cases} 1 & k = 0 \\ (1-p_{n_i,u})\cdot\alpha_{u,t}(L_{i-1},k-1) + \alpha_{u,t}(L_{i-1},k) & 1 \leq k \leq i \\ 0 & \text{otherwise} \end{cases} \tag{32}$$

This process builds up a recursive tree till $\alpha_{u,t}(L,k)$.

Moreover, since we compute $\alpha_{u,t}(L,k)$ for all possible k values,we can reuse the values of $\alpha_{u,t}(L_{i-1},k)$ for the computing of both $\alpha_{u,t}(L_i,k)$ and $\alpha_{u,t}(L_i,k+1)$. Combining recursion trees for each possible values of k together, we get a DP table with height $|L|$ and width $|L|$+1 in Figure 9. Because each node in the DP table computes in $O(1)$ time, the total time to compute all values up to $L_{|L|}$ is $O(|L|^2)$.

□

C APPENDIX FOR SECTION 4

Lemma 9. *Computing the Shapley value for any seed node in the 2-step termination is #P-hard.*

Proof. We prove this by a reduction from the problem of counting the number of satisfying assignments of a monotone 2-CNF formula, which is known to be #P-complete [49]. Let ϕ be a monotone 2-CNF formula with n variables $x_1,\ldots,x_n$ and m clauses $C_1,\ldots,C_m$. Each clause C_j is a disjunction of two unnegated variables. For each clause C_j, define $\text{Var}(C_j) = \{x_i \mid x_i \text{ appears in } C_j\}$. Denote the number of satisfying assignments of ϕ by $\#_\phi$. The proof proceeds in two main parts:

(1) **Graph Construction and Deriving the Shapley Value Expression (Appendix C.0.1).** For each pair (p,q) with $p \in \{1,\ldots,n\}$ and $q \in \{1,\ldots,m\}$, we build a graph instance $G_{(p,q)} = (V_{(p,q)}, E_{(p,q)})$ in the 2-step activation model. We fix one special seed node t_1 and derive its Shapley value in each graph instance, denoted by $Shap_{(p,q)}(t_1)$ (see Lemma 11).

(2) **Obtaining the number of satisfying assignments (Appendix C.0.2).** We show that if one can compute $Shap_{(p,q)}(t_1)$ for every pair (p,q) in polynomial time, then $\#_\phi$ can be obtained in polynomial time. The argument proceeds in following steps:
- **Step 1: Express $\#_\phi$ in terms of $\gamma_{k,c}$.** We begin by noting that there exists a bijection between subsets of variables and assignments to the variables in ϕ (see Observation 2), which allows us to define subcount variables $\gamma_{k,c}$ (the number of size-k subsets of variables that satisfy exactly c clauses). Then we can express $\#_\phi$ as the sum of $\gamma_{k,m}$ over all possible k (see Observation 3).
- **Step 2: Express $Shap_{(p,q)}(t_1)$ in terms of $\gamma_{k,c}$.** We relate the subsets of variable nodes in our construction to subsets of variables in ϕ (see Claim 1), showing that $Shap_{(p,q)}(t_1)$ can be rewritten using $\gamma_{k,c}$ (see Claim 2). So far, we connect the counting of satisfying assignments to the Shapley value of t_1 using $\gamma_{k,c}$.
- **Step 3: Form a matrix system and solve for $\gamma_{k,c}$** Collecting $Shap_{(p,q)}(t_1)$ for all pairs (p,q), we form a matrix system(Eq. (51)), prove that certain coefficient matrices are non-singular (Claim 3), and thus show that we can solve for all $\gamma_{c,k}$ in polynomial time once the Shapley values are known. It gives us the total number of satisfying assignments $\#_\phi$, completing the reduction.

This establishes the #P-hardness of computing the Shapley value for the given 2-step model.

C.0.1 *Graph Construction and Shapley Value Expression.*

Construction. For each pair (p,q), where $p \in \{1,\ldots,n\}$ and $q \in \{1,\ldots,m\}$, construct the graph $G_{(p,q)} = (V_{(p,q)}, E_{(p,q)})$ as follows (see Figure 10 for an example):

- Nodes:
  (1) Variable nodes: For each variable x_i, create a seed node u_i. Denote the set of seed nodes corresponding to variables as $U = \{u_1,u_2,\ldots,u_n\}$.
  (2) Clause nodes: For each clause C_i, create a non-seed node v_i. Denote the set of non-seed nodes corresponding to clauses as $V = \{v_1,v_2,\ldots,v_m\}$.
  (3) Sink node: Add a non-seed node s.
  (4) Additional seed nodes: Add p additional seed nodes $t_1,\ldots,t_p$. Denote the set of new seed nodes as $T_p = \{t_1,\ldots,t_p\}$.

  Thus the set of nodes in $G_{(p,q)}$ is $V_{(p,q)} = U \cup V \cup T_p \cup \{s\}$. The set of seed nodes is $U \cup T_p$, and the set of non-seed nodes is $V \cup \{s\}$.
- Edges:
  (1) Variable-to-clause edges: For each clause $C_j = (x_i \vee x_k)$, create two edges (u_i,v_j) and (u_k,v_j) with probability $p(u_i,v_j) = p(u_k,v_j) = 1$. In other words, $(u_i,v_j) \in E_{(p,q)}$ if and only if $x_i \in var(C_j)$.
  (2) Clause-to-sink edges: For each clause C_j, add an edge from its corresponding node to the sink node, (v_j,s), with probability $p(v_j,s) = \frac{q}{q+1}$.
  (3) Seed-to-sink edges: For each additional seed node $t_i \in \{t_1,\ldots,t_p\}$, add an edge to the sink node, (t_i,s), with probability $p(t_i,s) = 1$.

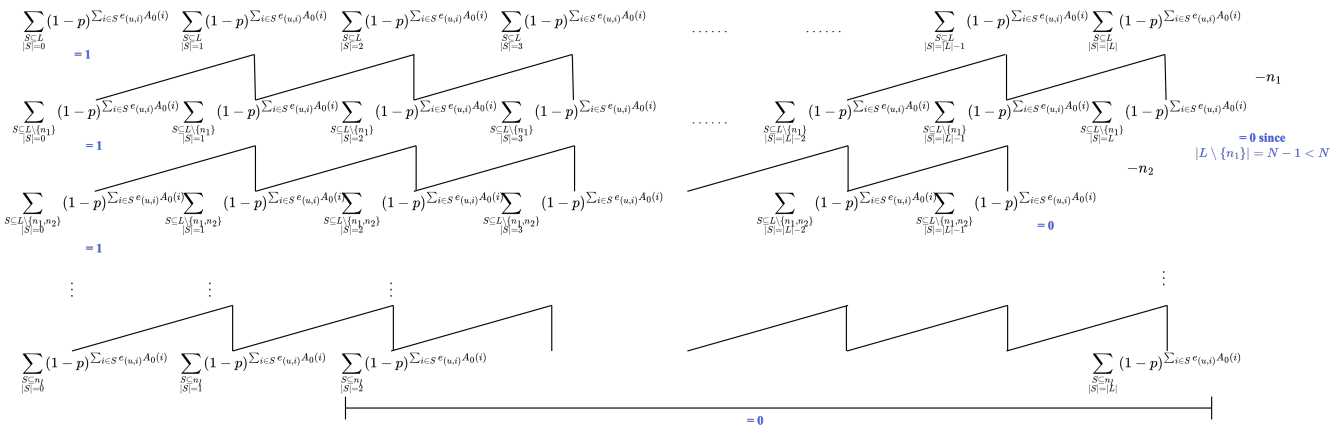


Figure 9: Recursion Tree for all sums

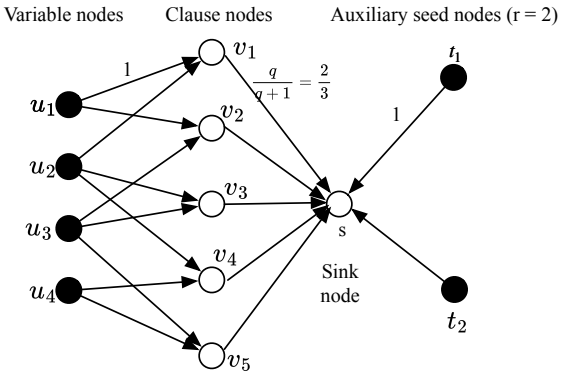


Figure 10: an example of instance graph $G_{(p,q)}$ for the proof of ?? where $p = 2, q = 2$. The monotone 2-CNF formula ϕ is $(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4)$.

For each pair of (p, q), the construction takes polynomial time, and we construct $m \cdot n$ such instances of network graphs in total.

Marginal contribution of t_1. We now derive the Shapley value of a fixed seed node t_1 in each instance $G_{(p,q)}$. Since the computation of Shapley value involves the marginal contributions of the seed node t_1 to the activation of the sink node s, we analyze the marginal contributions of t_1 first:

Lemma 10. *For each graph $G_{(p,q)}$ where $p \in \{1, \cdots, n\}$ and $q \in \{1, \cdots, m\}$, let $S \subseteq U \cup T_p \setminus \{t_1\}$ be any subset of nodes. The marginal contribution of t_1 to S is:*

$$U(S \cup t_1) - U(S) = \begin{cases} 1 & \text{if } S = \emptyset \\ (\frac{1}{q+1})^{\sum_{v_j \in V} \mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i, v_j) \in E_{(p,q)}]} & \text{if } S \subseteq U \text{ and } S \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

Proof. By the definition of the value function in Definition 1, the marginal contribution of t_1 to S is:

$$\begin{aligned} & U(S \cup t_1) - U(S) \\ = & \sum_{v \in V \cup \{s\}} \Pr[A(2, v) = 1 | \text{starting from } S \cup t_1] \\ & - \sum_{v \in V \cup \{s\}} \Pr[A(2, v) = 1 | \text{starting from } S] \end{aligned} \tag{34}$$

Since for each clause node $v \in V$, there is no path from t_1 to v:

$$\begin{aligned} & \sum_{v \in V} \Pr[A(2, v) = 1 | \text{starting from } S \cup t_1] \\ & - \sum_{v \in V} \Pr[A(2, v) = 1 | \text{starting from } S] \\ = & 0 \end{aligned} \tag{35}$$

Therefore, the marginal contribution of t_1 reduces to the difference in activation probabilities of the sink node s:

$$\begin{aligned} & U(S \cup t_1) - U(S) \\ = & \Pr[A(2, s) = 1 | \text{starting from } S \cup t_1] \\ & - \Pr[A(2, s) = 1 | \text{starting from } S] \end{aligned} \tag{36}$$

Then the computation of marginal contributions of t_1 can be divided into three cases for different subsets S:

(1) If S is an empty set, sink node s will be activated by t_1 along edge (t_1, s) with probability $p(t_1, s) = 1$, so the marginal contribution $U(t_1) - U(\emptyset) = 1 - 0 = 1$.
(2) If S is an non-empty set and $S \subseteq U$, i.e., contains only variable nodes, then the activation of s by S depends on a two-step process through clause nodes. The probability of s being activated at step 2 by S can be expressed as 1− the probability that s is not activated by any clause nodes in V:

$$\begin{aligned} & \Pr[A(2, s) = 1 | \text{starting from } S] \\ = & 1 - \prod_{v_j \in V} (1 - \frac{q}{q+1} \cdot \mathbb{1}[v_j \text{ is activated by } S]) \\ = & 1 - (\frac{1}{q+1})^{\sum_{v_j \in V} \mathbb{1}[v_j \text{ is activated by } S]} \end{aligned} \tag{37}$$

Since by the construction, each clause node $v_j \in V$ becomes active with probability 1 if and only if at least one of its neighbor

variable nodes is in S, the following equation holds for any subset $S \subseteq U$ and any clause node $v_j \in V$:

$$\mathbb{1}[v_j \text{ is activated by } S] = \mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i, v_j) \in E_{(p,q)}] \tag{38}$$

Plugging into (37), we have:

$$\begin{aligned} &\Pr[A(2,s) = 1 | \text{starting from } S] \\ &= 1 - (\frac{1}{q+1})^{\sum_{v_j \in V} \mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i,v_j) \in E_{(p,q)}]} \end{aligned} \tag{39}$$

Then if t_1 is added on top of S in that scenario, it directly activates s at step 1 with probability 1, regardless of the activation statuses of clause nodes.

Therefore, the marginal contribution of t_1 to S can be simplified as:

$$\begin{aligned} &U(S \cup t_1) - U(S) \\ &= 1 - \left(1 - (\frac{1}{q+1})^{\sum_{v_j \in V} \mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i,v_j) \in E_{(p,q)}]}\right) \\ &= (\frac{1}{q+1})^{\sum_{v_j \in V} \mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i,v_j) \in E_{(p,q)}]} \end{aligned} \tag{40}$$

(3) Otherwise, if S contains any seed nodes that is not variable nodes, i.e., S contains any seed node from $T_p \setminus \{t_1\}$, then the marginal contribution of t_1 is 0, since s will already be activated by S with probability 1.

In summary, we derive the Eq. (33) for any subset $S \subseteq U \cup T_p \setminus \{t_1\}$ which holds for each graph $G_{(p,q)}$. □

Shapley value of t_1. Now that we have derived the marginal contributions of t_1 for different subsets, we can compute its Shapley value. For each instance $G_{(p,q)}$, denote the Shapley value of t_1 as $Shap_{(p,q)}(t_1)$.

Lemma 11. *For each graph $G_{(p,q)}$, where $p \in \{1, \cdots, n\}$ and $q \in \{1, \cdots, m\}$, the Shapley value of t_1 is:*

$$\begin{aligned} Shap_{(p,q)}(t_1) &= \frac{1}{n+p} \\ &+ \sum_{k=1}^{n} \frac{k!\,(n+p-k-1)!}{(n+p)!} \cdot \sum_{\substack{S \subseteq U \\ |S|=k}} (\frac{1}{q+1})^{\sum_{v_j \in V} \mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i,v_j) \in E_{(p,q)}]} \end{aligned} \tag{41}$$

Proof. By the definition of Shapley value in Definition 2, we have:

$$\begin{aligned} &Shap_{(p,q)}(t_1) \\ &= \sum_{S \subseteq U \cup T_p \setminus \{t_1\}} \frac{|S|!\,(n+p-|S|-1)!}{(n+p)!} \cdot [U(S \cup t_1) - U(S)] \end{aligned} \tag{42}$$

First, we reorganize the sum by grouping subsets S by their size:

$$\begin{aligned} &Shap_{(p,q)}(t_1) \\ &= \sum_{k=0}^{n+p-1} \frac{k!\,(n+p-k-1)!}{(n+p)!} \cdot \sum_{\substack{S \subseteq U \cup T_p \\ |S|=k}} [U(S \cup t_1) - U(S)] \end{aligned} \tag{43}$$

Then we substitute the marginal contributions from Lemma 10 for different subsets S:

- When $k = 0$: The empty set contributes $\frac{0!(n+p-1)!}{(n+p)!} = \frac{1}{n+p}$
- When $1 \le k \le n$, we consider two subcases:
  - For subsets $S \subseteq U$ (containing only variable nodes), the marginal contribution is $(\frac{1}{q+1})^{\sum_{v_j \in V} \mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i,v_j) \in E_{(p,q)}]}$
  - Otherwise, the marginal contribution is 0.
- When $k > n$: Since there are only n variable nodes, all such sets contain at least one additional seednode from $T_p \setminus \{t_1\}$. Therefore, their marginal contribution is 0.

Collecting all the terms, we yield Eq. (41). □

C.0.2 *Obtaining the number of satisfying assignments.*

Step 1: Express $\#_\phi$ ***in terms of*** $\gamma_{k,c}$.

Observation 2. *For a monotone 2-CNF formula ϕ, there is a one-to-one correspondence between subsets of variables $\{x_1, \ldots, x_n\}$ and assignments to the variables. Specifically:*

(1) Every subset of variables uniquely maps to an assignment that setting variables in this subset to True and other variables to False.

(2) Conversely, any valid assignment to ϕ can be viewed as picking out those variables that are set to True to form a subset of variables.

By Observation 2, we say that a subset of variables *satisfies* c clauses of ϕ if the corresponding assignment (setting variables in this subset to True and other variables to False) makes exactly c clauses True.

Therefore, we define $\gamma_{k,c}$ as the number of size-k subsets of variables that satisfies exactly c clauses in ϕ, where $c \in \{0, 1, \ldots, m\}$ and $k \in \{0, 1, \ldots, n\}$. Note that since ϕ is a monotone 2-CNF formula, all literals are unnegated, thus all variables are set to False ($k = 0$) is equivalent to no clauses are satisfied ($c = 0$). Therefore, we have $\gamma_{0,0} = 1$, $\gamma_{k,0} = 0$ for all $k \ge 1$ and $\gamma_{0,c} = 0$ for all $c \ge 1$. Then we can express $\#_\phi$ using $\gamma_{k,c}$:

Observation 3. *The number of satisfying assignments $\#_\phi$ equals the sum of size-k subsets of variables that satisfy all m clauses over all possible subset sizes k:*

$$\#_\phi = \sum_{k=1}^{n} \gamma_{k,m} \tag{44}$$

Thus, if we can compute $\gamma_{k,c}$ for all $c \in \{1, \ldots, m\}$ and $k \in \{1, \ldots, n\}$, we can obtain $\#_\phi$ in polynomial time.

Step 2: Express $Shap_{(p,q)}(t_1)$ ***in terms of*** $\gamma_{k,c}$.

Claim 1. *Given a subset of variable nodes $S \subseteq U$, for any clause node $v_j \in V$:*

$$\mathbb{1}[\exists u_i \in S \text{ s.t. } (u_i, v_j) \in E_{(p,q)}] = \mathbb{1}[\exists x_i \in \mathrm{Var}(S) \text{ s.t. } x_i \in \mathrm{Var}(C_j)] \tag{45}$$

where $\mathrm{Var}(S)$ *denotes the set of variables corresponding to variable nodes in S for each subset $S \subseteq U$.*

Proof. By the construction of each $G_{(p,q)}$, an edge (u_i, v_j) exists if and only if variable x_i appears in clause C_j in ϕ. Therefore, $\exists u_i \in S$ such that $(u_i, v_j) \in E_{(p,q)}$ if and only if $\exists x_i \in \mathrm{Var}(S)$ such that $x_i \in \mathrm{Var}(C_j)$. □

Using Claim 1, the Shapley value of t_1 in each $G_{(p,q)}$ can be reduced to Eq. (46), in terms of $\gamma_{k,c}$:

Claim 2. *The Shapley value of t_1 in each instance $G_{(p,q)}$ can be expressed as:*

$$Shap_{(p,q)}(t_1) = \frac{1}{n+p} + \sum_{k=1}^{n}\sum_{c=1}^{m} \frac{k!\,(n+p-k-1)!}{(n+p)!} \cdot (\frac{1}{q+1})^c \cdot \gamma_{k,c} \tag{46}$$

Proof. By Claim 1, the inner sum of $Shap_{(p,q)}(t_1)$ in each instance $G_{(p,q)}$ can be rewrite as:

$$\sum_{\substack{S\subseteq U\\ |S|=k}} (\frac{1}{q+1})^{\sum_j \mathbb{1}[\exists u_i\in S \text{ s.t. } (u_i,v_j)\in E_{(p,q)}]} = \sum_{\substack{\text{Var}(S)\subseteq X\\ |\text{Var}(S)|=k}} (\frac{1}{q+1})^{\sum_j \mathbb{1}[\exists x_i\in \text{Var}(S) \text{ s.t. } x_i\in \text{Var}(C_j)]} \tag{47}$$

where X denotes the set of variables $\{x_1, \ldots, x_n\}$ in ϕ.

Since a clause C_j in ϕ is satisfied if and only if at least one of its variables is set to True. Therefore, having $\exists x_i \in \text{Var}(S)$ s.t. $x_i \in \text{Var}(C_j)$ precisely captures when clause C_j is satisfied by the assignment where variables in Var(S) are set to True and remaining variables are set to False. Thus:

$$\begin{aligned} &\sum_j \mathbb{1}[\exists x_i \in \text{Var}(S) \text{ s.t. } x_i \in \text{Var}(C_j)] \\ &= \sum_j \mathbb{1}[C_j \text{ is satisfied by Var}(S)] \\ &= |\{C_j \in \phi : C_j \text{ is satisfied by Var}(S)\}| \end{aligned} \tag{48}$$

which counts the number of clauses satisfied by the variables in Var(S).

For any fixed size $k \in [1, n]$, we can partition these set of variables Var(S) with $|\text{Var}(S)| = k$ according to the number of clauses they satisfy. Note that since at least one variable is set to True ($k \geq 1$), at least one clause is satisfied. Then we can simplify the inner sum as:

$$\begin{aligned} &\sum_{\substack{\text{Var}(S)\subseteq X\\ |\text{Var}(S)|=k}} (\frac{1}{q+1})^{|\{C_j\in\phi : C_j \text{ is satisfied by Var}(S)\}|} \\ &= \sum_{c=1}^{m} \sum_{\substack{\text{Var}(S)\subseteq X, |\text{Var}(S)|=k\\ |\{C_j\in\phi : C_j \text{ is satisfied by Var}(S)\}|=c}} (\frac{1}{q+1})^c \\ &= \sum_{c=1}^{m} \gamma_{k,c} (\frac{1}{q+1})^c \end{aligned} \tag{49}$$

Substituting Eq. (49) back into Eq. (41) from Lemma 11, we can rewrite the Shapley value expression as:

$$Shap_{(p,q)}(t_1) = \frac{1}{n+p} + \sum_{k=1}^{n} \frac{k!\,(n+p-k-1)!}{(n+p)!} \cdot \sum_{c=1}^{m} \gamma_{k,c} (\frac{1}{q+1})^c \tag{50}$$

Rearranging the terms, we obtain Eq. (46). □

Step 3: Form a matrix system and solve for $\gamma_{k,c}$. For each pair of parameters (p, q), we obtain the $Shap_{(p,q)}(t_1)$ of the form Eq. (46). We collect all $m \cdot n$ equations into a matrix system:

$$A \times \Gamma \times B = D \tag{51}$$

where:

- A is an $n \times n$ matrix with entries $a_{p,k} = \frac{k!(n+p-k-1)!}{(n+p)!}$ for $k, p \in [1, n]$:

$$A = \begin{pmatrix} \frac{1!(n-1)!}{(n+1)!} & \frac{2!(n-2)!}{(n+1)!} & \cdots & \frac{n!0!}{(n+1)!} \\ \frac{1!(n)!}{(n+2)!} & \frac{2!(n-1)!}{(n+2)!} & \cdots & \frac{n!1!}{(n+2)!} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1!(2n-2)!}{(2n)!} & \frac{2!(2n-3)!}{(2n)!} & \cdots & \frac{n!(n-1)!}{(2n)!} \end{pmatrix} \tag{52}$$

- B is an $m \times m$ matrix with entries $b_{c,q} = (\frac{1}{q+1})^c$ for $c, q \in [1, m]$:

$$B = \begin{pmatrix} (\frac{1}{2})^1 & (\frac{1}{3})^1 & \cdots & (\frac{1}{m+1})^1 \\ \vdots & \vdots & \ddots & \vdots \\ (\frac{1}{2})^m & (\frac{1}{3})^m & \cdots & (\frac{1}{m+1})^m \end{pmatrix} \tag{53}$$

- D is an $n \times m$ matrix with entries $d_{p,q} = Shap_{(p,q)}(t_1) - \frac{1}{n+p}$ for $p \in [1, n]$ and $q \in [1, m]$:

$$D = \begin{pmatrix} Shap_{(1,1)}(t_1) - \frac{1}{n+1} & \cdots & Shap_{(1,m)}(t_1) - \frac{1}{n+1} \\ \vdots & \ddots & \vdots \\ Shap_{(n,1)}(t_1) - \frac{1}{2n} & \cdots & Shap_{(n,m)}(t_1) - \frac{1}{2n} \end{pmatrix} \tag{54}$$

- Γ is an $n \times m$ matrix with each entry is $\gamma_{k,c}$ for $k \in [1, n]$ and $c \in [1, m]$:

$$\Gamma = \begin{pmatrix} \gamma_{1,1} & \gamma_{1,2} & \cdots & \gamma_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{n,1} & \gamma_{n,2} & \cdots & \gamma_{n,m} \end{pmatrix} \tag{55}$$

Now, each $Shap_{(p,q)}(t_1)$ can be expressed as:

$$\begin{aligned} &Shap_{(p,q)}(t_1) - \frac{1}{n+p} \\ &= d_{p,q} \\ &= \sum_{k=1}^{n}\sum_{c=1}^{m} a_{p,k} \cdot \gamma_{k,c} \cdot b_{c,q} \end{aligned} \tag{56}$$

A and B are known coefficient matrices, D is known if there exists an oracle for computing the Shapley value, and Γ is the matrix of unknown variables $\gamma_{k,c}$. Then we will show that A and B are non-singular, and thus we can solve for Γ in polynomial time.

Claim 3. *Both matrices A (Eq. (52)) and B (Eq. (53)) are non-singular.*

Proof. For the coefficient matrix A in Eq. (52), we multiply each row $i \in [1, n]$ by $(n + i)!$; divide each column $j \in [1, n]$ by $j!$; and reverse the order of columns. Through these transformations, we get the following matrix denoted as A':

$$A' = \begin{pmatrix} 0! & 1! & \cdots & (n-1)! \\ \vdots & \vdots & \ddots & \vdots \\ (n-1)! & n! & \cdots & (2n-2)! \end{pmatrix} \tag{57}$$

The entries of matrix A' is $a'_{i,j} = (i - 1 + j - 1)!$, and then its determinant is $\det(A') = \prod_{i=0}^{n-1} i!\, i! \neq 0$, thus A' is non-singular [2]. Because these algebraic manipulations preserve non-singularity, A is also non-singular.

For the coefficient matrix B in Eq. (53), we take the transpose of B:

$$B^\top = \begin{pmatrix} (\frac{1}{2})^1 & (\frac{1}{2})^2 & \cdots & (\frac{1}{2})^m \\ \vdots & \vdots & \ddots & \vdots \\ (\frac{1}{m+1})^1 & (\frac{1}{m+1})^2 & \cdots & (\frac{1}{m+1})^m \end{pmatrix} \tag{58}$$

Divide each row by $(\frac{1}{i+1})$, we get a Vandermonde matrix denoted as $(B^\top)'$:

$$B^{\top\prime} = \begin{pmatrix} 1 & \frac{1}{2} & \cdots & (\frac{1}{2})^{m-1} \\ 1 & \frac{1}{3} & \cdots & (\frac{1}{3})^{m-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \frac{1}{m+1} & \cdots & (\frac{1}{m+1})^{m-1} \end{pmatrix} \tag{59}$$

Matrix $(B^\top)'$ is a Vandermonde matrix, so the determinant is $\det((B^\top)') = \prod_{1\le i<j\le m}(\frac{1}{j+1} - \frac{1}{i+1}) \neq 0$ and it is a non-singular matrix. Thus $\det(B) = \det(B^\top) = \det((B^\top)') \cdot \prod_{i=1}^m \frac{1}{i+1} \neq 0$, implying that B is non-singular. □

Finally, we show that we can solve the matrix system Eq. (51) in polynomial time and thus compute the total number of satisfying assignments $\#_\phi$:

Claim 4. *If we can compute $Shap_{(p,q)}(t_1)$ for each $G_{(p,q)}$ in polynomial time, we can compute the number of satisfying assignments $\#_\phi$ in polynomial time.*

Proof. By Claim 3, we can solve all $\gamma_{k,c}$ by:

$$\Gamma = A^{-1} \times D \times B^{-1} \tag{60}$$

If we can compute $Shap_{(p,q)}(t_1)$ for each $G_{(p,q)}$ in polynomial time, we can compute all $\gamma_{k,c}$ in polynomial time through Eq. (60). Then we can sum $\gamma_{k,m}$ over all k to compute $\#_\phi$ in polynomial time. □

D APPENDIX FOR SECTION 5

D.1 Appendix for Monte-Carlo Permutation Estimator

Proposition 1. *Given $(G(V,E),T,p)$,* ApproxPermuteMC *returns $\widehat{Shap(t)}$ for all $t \in T$ such that: (1) $\mathbb{E}[\widehat{Shap(t)}] = Shap(t)$, $\forall t \in T$; (2) for any $\epsilon > 0$ and $\delta \in (0,1)$, with probability at least $1-\delta$, $|\widehat{Shap(t)} - Shap(t)| \le \epsilon$ holds for all $t \in T$; (3) the running time is $O\left(\frac{|V|^4}{\epsilon^4}\ln(\frac{|T|}{\delta} + \frac{|V|^2|T|^2}{\epsilon^2\delta}) \cdot |T| \cdot |E|\right)$.*

The proof consists of the following three parts.

D.1.1 Unbiasedness of the Estimator. We first show that the estimator of the value function $\hat{U}$ is unbiased. Then, based on the unbiasedness of $\hat{U}$, and the uniformity of the permutation sampling, we prove the unbiasedness of the Shapley value estimator $\widehat{Shap(t)}$.

Lemma 12 (Unbiasedness of value function estimation). *For any seed set $S \subseteq T$, the estimator $\hat{U}(S)$ is unbiased, i.e.,*

$$\mathbb{E}[\hat{U}(S)] = U(S) \tag{61}$$

Proof. For a fixed seed set S, let $\mathbb{I}_{i,v}(S)$ be the indicator random variable for whether node $v \in V\backslash T$ is activated in the i-th simulation starting from S. Then the estimator $\hat{U}(S)$ is:

$$\hat{U}(S) = \frac{1}{m}\sum_{i=1}^m \sum_{v\in V\backslash T} \mathbb{I}_{i,v}(S) \tag{62}$$

Taking the expectation of $\hat{U}(S)$, we have:

$$\mathbb{E}[\hat{U}(S)] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^m \sum_{v\in V\backslash T} \mathbb{I}_{i,v}(S)\right] \tag{63}$$

$$= \frac{1}{m}\sum_{i=1}^m \sum_{v\in V\backslash T} \mathbb{E}[\mathbb{I}_{i,v}(S)] \tag{64}$$

$$= \frac{1}{m}\sum_{i=1}^m \sum_{v\in V\backslash T} \Pr(\mathbb{I}_{i,v}(S) = 1) \tag{65}$$

$$= \sum_{v\in V\backslash T} \Pr(v \text{ is activated } |S) \tag{66}$$

$$= U(S) \tag{67}$$

Eq. (64) follows from linearity of expectation. Eq. (65) uses the fact that the expectation of an indicator random variable equals its probability of being 1.

Eq. (66) holds because all simulations are identically distributed, the probability that v is activated given seed set S is the same in all simulations, i.e., $\Pr(\mathbb{I}_{i,v}(S) = 1) = \Pr(\mathbb{I}_{j,v}(S) = 1)$ for any i, j. Denote $\Pr(v \text{ is activated } |S)$ as the probability that v is activated given seed set S in a single simulation. Then we have:

$$\frac{1}{m}\sum_{i=1}^m \Pr(\mathbb{I}_{i,v}(S) = 1) = \frac{1}{m} \cdot m \cdot \Pr(v \text{ is activated } |S) \tag{68}$$

which allows us to factor out $\frac{1}{m}\sum_{i=1}^m$.

Finally, Eq. (67) follows from Definition 1 that $U(S)$ is the expected number of activated nodes by S. So we have shown the unbiasedness of $\hat{U}(S)$ for any S. □

Then we prove the unbiasedness of the Shapley value estimator:

Lemma 13 (Unbiasedness of Shapley value estimator). *For any seed $t \in T$, the estimator $\widehat{Shap(t)}$ is unbiased, i.e.,*

$$\mathbb{E}[\widehat{Shap(t)}] = Shap(t) \tag{69}$$

Proof. By taking expectation on **??** and using the linearity of expectation, we have:

$$\begin{aligned} \mathbb{E}[\widehat{Shap(t)}] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n \hat{\Delta}_t^{\pi_i}\right] \\ &= \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\hat{\Delta}_t^{\pi_i}] \end{aligned} \tag{70}$$

Then for any single permutation π_i we have:

$$\mathbb{E}[\hat{\Delta}_t^{\pi_i}] = \mathbb{E}[\hat{U}(S_{\pi_i,t} \cup \{t\}) - \hat{U}(S_{\pi_i,t})] \tag{71}$$

Note that for any single permutation π_i, randomness comes from two sources: (1) the random sampling of the permutation π_i and (2) the random Monte Carlo simulations used to estimate U. Thus, using the law of total expectation:

$$\mathbb{E}[\hat{\Delta}_t^{\pi_i}] = \mathbb{E}_{\pi_i\sim\Pi(T)}\left[\mathbb{E}_{\hat{U}}\left[\hat{U}(S_{\pi_i,t} \cup \{t\}) - \hat{U}(S_{\pi_i,t})|\pi_i\right]\right] \tag{72}$$

By Lemma 12, $\hat{U}$ is an unbiased estimator of U for any fixed subset of seed nodes, i.e., $\mathbb{E}[\hat{U}(S)] = \mathbb{E}_{\hat{U}}[\hat{U}(S)] = U(S)$. Moreover,

since each permutation π_i is sampled uniformly at random from $\Pi(T)$, we have:

$$\begin{aligned}\mathbb{E}[\hat{\Delta}_t^{\pi_i}] &= \mathbb{E}_{\pi_i\sim\Pi(T)}\left[\mathbb{E}_{\hat{U}}\left[\hat{U}(S_{\pi_i,t}\cup\{t\}) - \hat{U}(S_{\pi_i,t})|\pi_i\right]\right] \\ &= \mathbb{E}_{\pi_i\sim\Pi(T)}\left[U(S_{\pi_i,t}\cup\{t\}) - U(S_{\pi_i,t})\right] \\ &= \frac{1}{|T|!}\sum_{\pi\in\Pi(T)}[U(S_{\pi,t}\cup\{t\}) - U(S_{\pi,t})]\end{aligned} \tag{73}$$

Therefore:

$$\begin{aligned}\mathbb{E}[\widehat{Shap(t)}] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\Delta}_t^{\pi_i}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{|T|!}\sum_{\pi\in\Pi(T)}[U(S_{\pi,t}\cup\{t\}) - U(S_{\pi,t})]\right) \\ &= \frac{1}{|T|!}\sum_{\pi\in\Pi(T)}[U(S_{\pi,t}\cup\{t\}) - U(S_{\pi,t})] \\ &= Shap(t)\end{aligned} \tag{74}$$

□

D.1.2 Approximation Bound. We will use the Hoeffding's inequality in the proof of Lemma 14.

Fact 1 (Hoeffding's inequality). *Given independent random variables $X_1, X_2, \cdots, X_n$ with bounds $a_i \le X_i \le b_i$, for all $t > 0$ it satisfies:*

$$\Pr[|\sum_{i=1}^{n}X_i - \mathbb{E}[\sum_{i=1}^{n}X_i]|\ge t] \le 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right) \tag{75}$$

Lemma 14 (Approximation error bound of Shapley value estimator). *For any $\epsilon > 0$ and $\delta > 0$, the following holds for all $t \in T$:*

$$\Pr\left(|\widehat{Shap(t)} - Shap(t)|\ge \epsilon\right) \le \delta. \tag{76}$$

if the number of permutations satisfies

$$n \ge \frac{8|V|^2}{\epsilon^2}\ln\left(\frac{4|T|}{\delta}\right) \tag{77}$$

and the number of Monte Carlo simulations satisfies

$$m \ge \frac{8|V|^2}{\epsilon^2}\ln(\frac{4n|T|}{\delta}) \tag{78}$$

Proof. The total error in estimating $Shap(t)$ can be decomposed as two terms as follows: (1) the error from estimating marginal contributions $E_1(t)$ and (2) the error from sampling a finite number of permutations $E_2(t)$.

$$\begin{aligned}|\widehat{Shap(t)} - Shap(t)| &= \left|\frac{1}{n}\sum_{i=1}^{n}\hat{\Delta}_t^{\pi_i} - Shap(t)\right| \\ &= \left|\frac{1}{n}\sum_{i=1}^{n}\hat{\Delta}_t^{\pi_i} - \frac{1}{n}\sum_{i=1}^{n}\Delta_t^{\pi_i} + \frac{1}{n}\sum_{i=1}^{n}\Delta_t^{\pi_i} - Shap(t)\right| \\ &\le \underbrace{\left|\frac{1}{n}\sum_{i=1}^{n}(\hat{\Delta}_t^{\pi_i} - \Delta_t^{\pi_i})\right|}_{E_1(t)} + \underbrace{\left|\frac{1}{n}\sum_{i=1}^{n}\Delta_t^{\pi_i} - Shap(t)\right|}_{E_2(t)}\end{aligned} \tag{79}$$

We bound the probability of $E_1(t)$ and $E_2(t)$ separately using Hoeffding's inequality and then take the union bound.

Error in marginal contribution estimation. For each marginal contribution $\Delta_t^{\pi_i}$, the estimation error is:

$$\begin{aligned}&|\hat{\Delta}_t^{\pi_i} - \Delta_t^{\pi_i}| = |\hat{U}(S_{\pi_i,t}\cup\{t\}) - \hat{U}(S_{\pi_i,t}) - (U(S_{\pi_i,t}\cup\{t\}) - U(S_{\pi_i,t}))| \\ &\le |\hat{U}(S_{\pi_i,t}\cup\{t\}) - U(S_{\pi_i,t}\cup\{t\})| + |\hat{U}(S_{\pi_i,t}) - U(S_{\pi_i,t})|\end{aligned} \tag{80}$$

Since both $\hat{U}(S_{\pi_i,t}\cup\{t\})$ and $\hat{U}(S_{\pi_i,t})$ are estimated via Monte Carlo simulations with $|V|$ as the upper bound, applying Hoeffding's inequality for each $\hat{\Delta}_t^{\pi_i}$ and set ϵ_Δ as the maximum error in estimating $\Delta_t^{\pi_i}$:

$$\begin{aligned}\Pr\left(\left|\hat{\Delta}_t^{\pi_i} - \Delta_t^{\pi_i}\right| \ge \epsilon_\Delta\right) &\le 2\exp\left(-\frac{2m\epsilon_\Delta^2}{(2|V|)^2}\right) \\ &= 2\exp\left(-\frac{m\epsilon_\Delta^2}{2|V|^2}\right)\end{aligned} \tag{81}$$

Then we apply this result to bound the probability of $E_1(t)$ exceeding ϵ_Δ:

$$\begin{aligned}\Pr(E_1(t)\ge\epsilon_\Delta) &= \Pr\left(\left|\frac{1}{n}\sum_{i=1}^{n}(\hat{\Delta}_t^{\pi_i} - \Delta_t^{\pi_i})\right| \ge \epsilon_\Delta\right) \\ &\le \Pr\left(\exists i : |\hat{\Delta}_t^{\pi_i} - \Delta_t^{\pi_i}|\ge\epsilon_\Delta\right)\end{aligned} \tag{82}$$

Because we estimated the marginal contribution of a given seed node for all n permutations, we have:

$$\begin{aligned}\Pr(E_1(t)\ge\epsilon_\Delta) &\le \Pr\left(\exists i : |\hat{\Delta}_t^{\pi_i} - \Delta_t^{\pi_i}|\ge\epsilon_\Delta\right) \\ &\le n\cdot\Pr\left(\left|\hat{\Delta}_t^{\pi_i} - \Delta_t^{\pi_i}\right|\ge\epsilon_\Delta\right) \\ &\le n\cdot 2\exp\left(-\frac{m\epsilon_\Delta^2}{2|V|^2}\right)\end{aligned} \tag{83}$$

Applying the union bound over all seeds:

$$\Pr(\exists t\in T : E_1(t)\ge\epsilon_\Delta) \le |T|\cdot n\cdot 2\exp\left(-\frac{m\epsilon_\Delta^2}{2|V|^2}\right) \tag{84}$$

Then we set $\epsilon_\Delta = \frac{\epsilon}{2}$ and replace m using Eq. (78):

$$\begin{aligned}&\Pr(\exists t\in T : E_1(t)\ge\frac{\epsilon}{2}) \\ &\le |T|\cdot n\cdot 2\exp\left(-\frac{\epsilon^2}{8|V|^2}\cdot\frac{8|V|^2}{\epsilon^2}\ln(\frac{4n|T|}{\delta})\right) \\ &= \frac{\delta}{2}\end{aligned} \tag{85}$$

Therefore, the probability that $\forall t\in T$, $E_1(t)$ at most $\frac{\epsilon}{2}$ is at least $1-\frac{\delta}{2}$.

Error from sampling permutations. Then we consider $E_2(t)$, the error from sampling permutations. Both $U(S_{\pi_i,t}\cup\{t\})$ and $U(S_{\pi_i,t})$ are bounded between 0 and $|V|$, so we can apply Hoeffding's inequality as before and set ϵ_{perm} as the maximum error for $E_2(t)$:

$$\Pr\left(E_2(t) \geq \epsilon_{\text{perm}}\right) = \Pr\left(\left|\frac{1}{n}\sum_{i=1}^{n}\Delta_t^{\pi_i} - Shap(t)\right| \geq \epsilon_{\text{perm}}\right)$$
$$\leq 2\exp\left(-\frac{2n\epsilon_{\text{perm}}^2}{(2|V|)^2}\right) = 2\exp\left(-\frac{n\epsilon_{\text{perm}}^2}{2|V|^2}\right) \tag{86}$$

Similarly, applying the union bound over all seeds:

$$\Pr\left(\exists t \in T : E_2(t) \geq \epsilon_{\text{perm}}\right) \leq |T|\cdot 2\exp\left(-\frac{n\epsilon_{\text{perm}}^2}{2|V|^2}\right) \tag{87}$$

We set $\epsilon_{\text{perm}} = \frac{\epsilon}{2}$ and replace n using Eq. (77):

$$\begin{aligned}&\Pr\left(\exists t \in T : E_2(t) \geq \frac{\epsilon}{2}\right)\\ &\leq |T|\cdot 2\exp\left(-\frac{\epsilon^2}{8|V|^2}\cdot\frac{8|V|^2}{\epsilon^2}\ln(\frac{4|T|}{\delta})\right)\\ &= \frac{\delta}{2}\end{aligned} \tag{88}$$

Hence, the probability that $\forall t \in T$, $E_2(t)$ at most $\frac{\epsilon}{2}$ is at least $1 - \frac{\delta}{2}$.

Combining the errors. Combine the probability of $E_1(t)$ and $E_2(t)$ exceeding their respective errors:

$$\begin{aligned}&\Pr\left(\exists t \in T : \left|\hat{Shap}(t) - Shap(t)\right| \geq \epsilon\right)\\ &\leq \Pr\left(\exists t \in T : E_1(t) + E_2(t) \geq \epsilon\right)\\ &\leq \Pr\left(\exists t \in T : E_1(t) \geq \frac{\epsilon}{2}\right) + \Pr\left(\exists t \in T : E_2(t) \geq \frac{\epsilon}{2}\right)\\ &\leq \frac{\delta}{2} + \frac{\delta}{2} = \delta.\end{aligned} \tag{89}$$

This shows that with probability at least $1 - \delta$, the total error for all seeds is bounded by ϵ, completing the proof of the approximation guarantee.

□

D.1.3 Runtime Complexity. Last, we prove the polynomial runtime of ApproxPermuteMC:

Lemma 15. ApproxPermuteMC *runs in polynomial time in* $|V|$, $|T|$, $1/\epsilon$, *and* $\ln(1/\delta)$.

Proof. For each of the n permutations and each seed node in T, we perform m Monte Carlo simulations. So the total number of simulations is $O(n \cdot |T| \cdot m)$. Since each simulation runs a BFS traversal that takes $O(|E|)$ time, the runtime per simulation is $O(|E|)$. therefore, the total runtime is $O(n \cdot |T| \cdot m \cdot |E|)$

As analyzed in Lemma 14, the sampling sizes m and n are polynomial:

$$m \geq \frac{8|V|^2}{\epsilon^2}\ln(\frac{4n|T|}{\delta}) = O\left(\frac{|V|^2}{\epsilon^2}\ln(\frac{|V|^2|T|^2}{\epsilon^2\delta})\right) \tag{90}$$

$$n \geq \frac{8|V|^2}{\epsilon^2}\ln(\frac{4|T|}{\delta}) = O\left(\frac{|V|^2}{\epsilon^2}\ln(\frac{|T|}{\delta})\right) \tag{91}$$

As $|E| = O(|V|^2)$ and $|T| \leq |V|$, substituting the values of n and m, the total runtime is polynomial in $|V|$, $\frac{1}{\epsilon}$, and $\ln(\frac{1}{\delta})$. □

D.2 Appendix for Single-Realization Permutation Estimator

Proposition 8. ApproxPermuteDirect *achieves an* additive *FPRAS, satisifying:*

(1) (Unbiasedness) $\mathbb{E}[\widehat{Shap(t)}] = Shap(t)$.

(2) (Approximation bound) For any $\epsilon > 0$ *and* $\delta \in (0,1)$, *with probability at least* $1 - \delta$, $\Pr\left[\ |\widehat{Shap(t)} - Shap(t)| \leq \epsilon\right] \geq 1 - \delta$ *is satisfied for all* $t \in T$ *if:* $n_\pi = O\left(\frac{|V|^2}{\epsilon^2}\ln\frac{|T|}{\delta}\right)$.

(3) (Running time) The algorithm runs in polynomial time in the input size, $\frac{1}{\epsilon}$ *and* $\ln(\frac{1}{\delta})$, *with the total runtime of* $O(n_\pi \cdot T \cdot |E|)$.

D.2.1 Unbiasedness of the Estimator. To prove the unbiasedness of the estimator, we need to show that the expected value of the estimator is equal to the true Shapley value.

Claim 5. *The estimator* $\widehat{Shap(t)}$ *is unbiased, i.e.,*

$$\mathbb{E}[\widehat{Shap(t)}] = Shap(t) \tag{92}$$

Proof. We need to consider two sources of randomness: (1) the randomness in estimating marginal contributions using realization graph g, and (2) the randomness in sampling permutations. Fixed an permutation π_i, the expected value of $\hat{\Delta}_t^{\pi_i}$ over the randomness of realization graph g is:

$$\begin{aligned}\mathbb{E}_g[\hat{\Delta}_t^{\pi_i}|\pi_i] &= \mathbb{E}[u_g(S_{\pi_i,t} \cup \{t\}) - u_g(S_{\pi_i,t})]\\ &= \mathbb{E}[u_g(S_{\pi_i,t} \cup \{t\})] - \mathbb{E}[u_g(S_{\pi_i,t})]\\ &= U(S_{\pi_i,t} \cup \{t\}) - U(S_{\pi_i,t})\\ &= \Delta_t^{\pi_i}\end{aligned} \tag{93}$$

The first equality is by definition. The second equality follows from linearity of expectation. The third equality holds because for any seed set S, $\mathbb{E}[u_g(S)] = U(S)$, which can be proven as follows:

For any seed set S, each node $v \in V \setminus T$, let $\mathbb{I}_{g,v}(S)$ be the indicator random variable that equals 1 if v is reachable from S in g, and 0 otherwise. Then:

$$\begin{aligned}\mathbb{E}[u_g(S)] &= \mathbb{E}[\sum_{v\in V\setminus T}\mathbb{I}_{g,v}(S)]\\ &= \sum_{v\in V\setminus T}\mathbb{E}[\mathbb{I}_{g,v}(S)]\\ &= \sum_{v\in V\setminus T}\Pr(v \text{ is activated by } S)\\ &= U(S)\end{aligned} \tag{94}$$

The first equality follows from the definition of $u_g(S)$. The second equality uses linearity of expectation. The third equality holds because $\mathbb{E}[\mathbb{I}_{g,v}(S)]$ is the probability that v is reachable from S in a random live-edge graph, which equals the probability that v is activated by S in the original graph under the IC model. The final equality follows from the definition of $U(S)$ as the expected number of nodes activated by seed set S.

Next, we average over the randomness of the permutations. We apply the law of total expectation:

$$\begin{aligned}\mathbb{E}[\widehat{Shap(t)}] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{\Delta}_t^{\pi_i}\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\hat{\Delta}_t^{\pi_i}] \\ &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\pi_i}\left[\mathbb{E}_g[\hat{\Delta}_t^{\pi_i}|\pi_i]\right]\end{aligned} \tag{95}$$

From our previous proof, we know that for any fixed permutation π_i:

$$\mathbb{E}_g[\hat{\Delta}(\pi_i, t)|\pi_i] = \Delta_t^{\pi_i} \tag{96}$$

Therefore,

$$\begin{aligned}\mathbb{E}[\widehat{Shap(t)}] &= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\pi_i}[\Delta_t^{\pi_i}] \\ &= \mathbb{E}_{\pi}[\Delta_t^{\pi}] \\ &= \frac{1}{|T|!}\sum_{\pi\in\Pi(T)}[U(S_{\pi,t}\cup\{t\}) - U(S_{\pi,t})] \\ &= Shap(t)\end{aligned} \tag{97}$$

The first equality uses linearity of expectation. The second equality holds because each π_i is sampled uniformly at random from $\Pi(T)$. The third equality expands the expectation over all possible permutations. The final equality follows from the definition of Shapley value. □

D.2.2 Additive Approximation Bounds. We now prove both additive approximation bounds for our estimator.

Theorem D.1 (Additive Approximation). *For any $\epsilon > 0$ and $\delta \in (0,1)$, with probability at least $1-\delta$:*

$$|\widehat{Shap(t)} - Shap(t)| \leq \epsilon \tag{98}$$

when $n \geq O(\frac{|V|^2}{2\epsilon^2}\ln(\frac{2|T|}{\delta}))$.

Proof. Since each $\hat{\Delta}(\pi_i, t)$ is bounded in $[0, |V|]$ and independently sampled, for each seed node t, we can apply Hoeffding's inequality for each

$$\begin{aligned}\Pr(|\widehat{Shap(t)} - Shap(t)| \geq \epsilon) &= \Pr\left(\left|\frac{1}{n}\sum_{i=1}^{n}\hat{\Delta}(\pi_i,t) - \mathbb{E}[\hat{\Delta}(\pi_i,t)]\right| \geq \epsilon\right) \\ &\leq 2\exp\left(-\frac{2n\epsilon^2}{|V|^2}\right)\end{aligned} \tag{99}$$

Applying the union bound over all seed nodes, we get:

$$\begin{aligned}&\Pr(\exists t \in T : |\widehat{Shap(t)} - Shap(t)| \geq \epsilon) \\ &\leq \sum_{t\in T}\Pr(|\widehat{Shap(t)} - Shap(t)| \geq \epsilon) \\ &\leq |T|\cdot 2\exp\left(-\frac{2n\epsilon^2}{|V|^2}\right)\end{aligned} \tag{100}$$

Replace n with $O(\frac{|V|^2}{2\epsilon^2}\ln(\frac{2|T|}{\delta}))$ in the above inequality, we get:

$$\Pr(|\widehat{Shap(t)} - Shap(t)| \geq \epsilon) \leq |T|\cdot 2\exp\left(-\frac{2\epsilon^2}{|V|^2}\cdot\frac{|V|^2}{2\epsilon^2}\ln(\frac{2|T|}{\delta})\right) \leq \delta \tag{101}$$

□

E APPENDIX FOR SECTION 5.3

Lemma 16. *For any $\varepsilon > 0$, $\ell > 0$, and $k \in [|T|]$. If the number of RR sets θ satisfies:*

$$\theta \geq \frac{n'(\ell\ln n' + \ln|T| + \ln 4)\left(2+\frac{2}{3}\varepsilon\right)}{\varepsilon^2 Shap^{(k)}} \tag{102}$$

where $Shap^{(k)}$ is the k-th largest Shapley value among all $\{Shap(t)\}_{t\in T}$. Then, the following holds with probability at least $1-\frac{1}{2n'^{\ell}}$:

(1) For all $t \in T$ with $Shap(t) > Shap^{(k)}$,

$$\left|\widehat{Shap(t)} - Shap(t)\right| \leq \varepsilon Shap(t) \tag{103}$$

(2) For all $t \in T$ with $Shap(t) \leq Shap^{(k)}$,

$$\left|\widehat{Shap(t)} - Shap(t)\right| \leq \varepsilon Shap^{(k)} \tag{104}$$

E.1 Proof of Lemma 16

Before we present the proof of Lemma 16, we present the following Chernoff bounds [12] that will be used in our analysis:

Fact 2 (Chernoff Bounds). *Let Y be the sum of t i.i.d. random variables with mean μ and within the range $[0,1]$. For any $\delta > 0$, the following upper tail bound holds:*

$$\Pr\{Y - t\mu \geq \delta\cdot t\mu\} \leq \exp\left(-\frac{\delta^2}{2+\frac{2}{3}\delta}t\mu\right). \tag{105}$$

For any $0 < \delta < 1$, the lower tail bound is given by:

$$\Pr\{Y - t\mu \leq -\delta\cdot t\mu\} \leq \exp\left(-\frac{\delta^2}{2}t\mu\right). \tag{106}$$

Lemma 16. *For any $\varepsilon > 0$, $\ell > 0$, and $k \in [|T|]$. If the number of RR sets θ satisfies:*

$$\theta \geq \frac{n'(\ell\ln n' + \ln|T| + \ln 4)\left(2+\frac{2}{3}\varepsilon\right)}{\varepsilon^2 Shap^{(k)}} \tag{102}$$

where $Shap^{(k)}$ is the k-th largest Shapley value among all $\{Shap(t)\}_{t\in T}$. Then, the following holds with probability at least $1-\frac{1}{2n'^{\ell}}$:

(1) For all $t \in T$ with $Shap(t) > Shap^{(k)}$,

$$\left|\widehat{Shap(t)} - Shap(t)\right| \leq \varepsilon Shap(t) \tag{103}$$

(2) For all $t \in T$ with $Shap(t) \leq Shap^{(k)}$,

$$\left|\widehat{Shap(t)} - Shap(t)\right| \leq \varepsilon Shap^{(k)} \tag{104}$$

Proof. First, for every $t \in T$ with $Shap(t) > Shap^{(k)}$:

$$\Pr[|\widehat{Shap}(t) - Shap(t)| \geq \varepsilon \cdot Shap(t)] \tag{107}$$

$$= Pr[|\frac{n'}{\theta}\sum_{j=1}^{\theta} X_{\boldsymbol{R}_j}(t) - Shap(t)| \geq \varepsilon \cdot Shap(t)] \tag{108}$$

$$= Pr[|\sum_{j=1}^{\theta} X_{\boldsymbol{R}_j}(t) - \theta \cdot \frac{Shap(t)}{n'}| \geq \varepsilon \cdot \theta \cdot \frac{Shap(t)}{n'}] \tag{109}$$

$$= Pr[|\sum_{j=1}^{\theta} X_{\boldsymbol{R}_j}(t) - \theta \cdot \mathbb{E}[X_{\boldsymbol{R}_j}(t)]| \geq \varepsilon \cdot \theta \cdot \mathbb{E}[X_{\boldsymbol{R}_j}(t)]] \tag{110}$$

$$\leq 2\exp(-\frac{\varepsilon^2}{2+\frac{2}{3}\varepsilon} \cdot \frac{\theta}{n'} \cdot Shap(t)) \quad \text{(By the Chernoff Bound)} \tag{111}$$

$$\leq 2\exp(-\frac{\varepsilon^2}{2+\frac{2}{3}\varepsilon} \cdot \frac{Shap(t)}{n'} \cdot \frac{n'\,(\ell \ln n' + \ln|T| + \ln 4)\left(2+\frac{2}{3}\varepsilon\right)}{\varepsilon^2 Shap^{(k)}}) \tag{112}$$

(By Eq. (102))

$$= 2\exp(-\frac{Shap(t)}{Shap^{(k)}} \cdot (\ell \ln n' + \ln|T| + \ln 4)) \tag{113}$$

$$\leq 2\exp(-\ell \ln n' - \ln|T| - \ln 4) \quad (\text{By } Shap(t) > Shap^{(k)}) \tag{114}$$

$$= \frac{1}{2|T|n'^{\ell}} \tag{115}$$

Then, for every $t \in T$ with $Shap(t) \leq Shap^{(k)}$, we have:

$$\Pr[|\hat{Shap}(t) - Shap(t)| \geq \varepsilon \cdot Shap^{(k)}] \tag{116}$$

$$= Pr[|\sum_{j=1}^{\theta} X_{\boldsymbol{R}_j}(t) - \theta \cdot \frac{Shap(t)}{n'}| \geq (\varepsilon \cdot \frac{Shap^{(k)}}{Shap(t)}) \cdot (\theta \cdot \frac{Shap(t)}{n'})] \tag{117}$$

$$\leq 2\exp(-\frac{(\varepsilon \cdot \frac{Shap^{(k)}}{Shap(t)})^2}{2+\frac{2}{3}(\varepsilon \cdot \frac{Shap^{(k)}}{Shap(t)})} \cdot \frac{\theta}{n'} \cdot Shap(t) \tag{118}$$

(By the Chernoff Bound)

$$= 2\exp(-\frac{\varepsilon^2 (Shap^{(k)})^2}{n'(2Shap(t) + \frac{2}{3}\varepsilon Shap^{(k)})} \cdot \theta) \tag{119}$$

$$\leq 2\exp(-\frac{\varepsilon^2 Shap^{(k)}}{n'(2+\frac{2}{3}\varepsilon)} \cdot \theta) \quad (Shap(t) \leq Shap^{(k)}) \tag{120}$$

$$\leq 2\exp(-\frac{\varepsilon^2 Shap^{(k)}}{n'(2+\frac{2}{3}\varepsilon)} \cdot \frac{n'\,(\ell \ln n' + \ln|T| + \ln 4)\left(2+\frac{2}{3}\varepsilon\right)}{\varepsilon^2 Shap^{(k)}}) \tag{121}$$

(By Eq. (102))

$$\leq 2\exp(-\ell \ln n' - \ln|T| - \ln 4) \tag{122}$$

$$= \frac{1}{2|T|n'^{\ell}} \tag{123}$$

Last, we apply the union bound to the two cases to obtain the final result that for all $t \in T$:

$$\Pr[\text{Approximation guarantee fails for some } t \in T] \leq |T| \cdot \frac{1}{2|T|n'^{\ell}} = \frac{1}{2n'^{\ell}} \tag{124}$$

□

E.2 Estimating a Lower Bound for $Shap^{(k)}$ Using a Martingale Approach

E.2.1 Martingale Definition and Properties.

Definition 5 (Martingale). *A sequence of random variables $\{Y_i(t)\}_{i\geq 1}$ is a martingale if and only if that for all $i \geq 1$, (1) $\mathbb{E}[|Y_i|] < \infty$; (2) $\mathbb{E}[Y_i(t) \mid Y_1(t), Y_2(t), \cdots, Y_{i-1}(t)] = Y_{i-1}(t)$.*

Lemma 17. *Let θ' be the number of RR sets generated in Phase 1, and let $\boldsymbol{R}_1^{(1)}, \boldsymbol{R}_2^{(1)}, \ldots, \boldsymbol{R}_{\theta'}^{(1)}$ be these RR sets. For every $t \in T$ and every $i \geq 1$,*

$$\mathbb{E}\left[X_{\boldsymbol{R}_i^{(1)}}(t) \mid X_{\boldsymbol{R}_2^{(1)}}(t), \cdots, X_{\boldsymbol{R}_{i-1}^{(1)}}(t)\right] = \frac{Shap(t)}{n'}, \tag{125}$$

Define the sequence $\{Y_i(t)\}_{i\geq 1}$ as the following:

$$\boldsymbol{Y}_i(t) = \sum_{j=1}^{i}\left(X_{\boldsymbol{R}_j^{(1)}}(t) - \frac{Shap(t)}{n'}\right). \tag{126}$$

Then for every $t \in T$, $\{Y_i(t), i \geq 1\}$ is a martingale.

Proof. Since the generation process of each RR set independent, each $X_{\boldsymbol{R}_i^{(1)}}(t)$ is independent of the previous RR sets. Then for every $t \in T$ and $i \geq 1$, we have:

$$\mathbb{E}\left[X_{\boldsymbol{R}_i^{(1)}}(t) \mid X_{\boldsymbol{R}_2^{(1)}}(t), \cdots, X_{\boldsymbol{R}_{i-1}^{(1)}}(t)\right] = \frac{Shap(t)}{n'} \tag{127}$$

By the definition of $\boldsymbol{Y}_i(t)$, we know that first, the value range of $\boldsymbol{Y}_i(t)$ is $[-i, i]$. Second,

$$\boldsymbol{Y}_i(t) = \boldsymbol{Y}_{i-1}(t) + \left(X_{\boldsymbol{R}_i^{(1)}}(t) - \frac{Shap(t)}{n'}\right) \tag{128}$$

By Eq. (127), we have:

$$\begin{aligned}
&\mathbb{E}\left[\boldsymbol{Y}_i(t) \mid \boldsymbol{Y}_{i-1}(t), \cdots, \boldsymbol{Y}_1(t)\right] \\
&= \boldsymbol{Y}_{i-1}(t) + \mathbb{E}\left[X_{\boldsymbol{R}_i^{(1)}}(t) \mid X_{\boldsymbol{R}_2^{(1)}}(t), \cdots, X_{\boldsymbol{R}_{i-1}^{(1)}}(t)\right] - \frac{Shap(t)}{n'} \\
&= \boldsymbol{Y}_{i-1}(t) + \frac{Shap(t)}{n'} - \frac{Shap(t)}{n'} \\
&= \boldsymbol{Y}_{i-1}(t)
\end{aligned} \tag{129}$$

Therefore, we prove that $\{Y_i(t), i \geq 1\}$ is a martingale. □

E.2.2 Martingale Concentration Inequality.

We utilize the following martingale concentration inequality as [10].

In our context, since $X_{\boldsymbol{R}_i^{(1)}}(t) \in [0,1]$ and $\frac{Shap(t)}{n'} \in [0,1]$, the martingale sequence satisfies $|X_{\boldsymbol{R}_i^{(1)}}(t) - \frac{Shap(t)}{n'}| \leq 1$. So we can apply the following tail bounds:

Fact 3 (Martingale Tail Bounds). *Let $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_t$ be a sequence of random variables such that (1) the value range is $[0,1]$ for each X_i; (2) for some $\mu \in [0,1]$, $\mathbb{E}[\boldsymbol{X}_i \mid \boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{i-1}] = \mu$ for every $i \in [t]$. Let $\boldsymbol{Y} = \sum_{i=1}^{t} \boldsymbol{X}_i$. For any $\delta > 0$, we have:*

$$\Pr\{\boldsymbol{Y} - t\mu \geq \delta \cdot t\mu\} \leq \exp\left(-\frac{\delta^2}{2+\frac{2}{3}\delta} t\mu\right) \tag{130}$$

For any $0 < \delta < 1$, we have

$$\Pr\{Y - t\mu \le -\delta \cdot t\mu\} \le \exp\left(-\frac{\delta^2}{2} t\mu\right) \tag{131}$$

E.2.3 Analysis of the Estimation Algorithm. The following lemma shows that the estimator $\boldsymbol{est}_i^{(k)}$ is a good estimate of $Shap^{(k)}$ for both case that x_i is greater than or less than $Shap^{(k)}$.

*

Proof. Let $\boldsymbol{R}_1^{(1)}, \boldsymbol{R}_2^{(1)}, \ldots, \boldsymbol{R}_{\theta_i}^{(1)}$ be the θ_i generated RR sets by the end of the i-th iteration of the for-loop of Phase 1. By Lemma 17, we can apply the martingale tail bound of Fact 29 on the sequence $\{X_{\boldsymbol{R}_1^{(1)}}(t), X_{\boldsymbol{R}_2^{(1)}}(t), \cdots, X_{\boldsymbol{R}_i^{(1)}}(t)\}$ for each $t \in T$.

Denote $\boldsymbol{est}_{t,i}$ as the value of $\boldsymbol{est}_t$ in the i-th iteration of the same for-loop. Then we have $\boldsymbol{est}_{t,i} = \sum_{j=1}^{\theta_i} X_{\boldsymbol{R}_j^{(1)}}(t)$. Denote $\widehat{Shap}_i(t) = \frac{n' \cdot \boldsymbol{est}_{t,i}}{\theta_i}$ as the estimator of $Shap(t)$ at the end of the i-th iteration.

Case 1: Consider $x_i > Shap^{(k)}$. First, for every $t \in T$ such that $Shap(t) \le Shap^{(k)}$, we have:

$$\begin{aligned}
&\Pr\left\{\frac{n' \cdot \boldsymbol{est}_{t,i}}{\theta_i} \ge (1+\varepsilon') \cdot x_i\right\} \\
&= \Pr\left\{\boldsymbol{est}_{t,i} \ge (1+\varepsilon')\frac{\theta_i \cdot x_i}{n'}\right\} \\
&= \Pr\left\{\boldsymbol{est}_{t,i} - \theta_i \cdot \frac{x_i}{n'} \ge \varepsilon' \frac{\theta_i \cdot x_i}{n'}\right\} \\
&\le \Pr\left\{\boldsymbol{est}_{t,i} - \theta_i \cdot \frac{Shap(t)}{n'} \ge (\varepsilon' \cdot \frac{x_i}{Shap(t)})\frac{\theta_i \cdot Shap(t)}{n'}\right\} \quad (\text{By } x_i > Shap(t)) \\
&\le \exp\left(-\frac{\left(\varepsilon' \cdot \frac{x_i}{Shap(t)}\right)^2}{2 + \frac{2}{3}\left(\varepsilon' \cdot \frac{x_i}{Shap(t)}\right)} \cdot \theta_i \cdot \frac{Shap(t)}{n'}\right) \quad (\text{By Fact 3}) \\
&= \exp\left(-\frac{\varepsilon'^2 \cdot x_i^2}{2Shap(t) + \frac{2}{3}\varepsilon' x_i} \cdot \frac{\theta_i}{n'}\right) \\
&\le \exp\left(-\frac{\varepsilon'^2 \cdot x_i}{2 + \frac{2}{3}\varepsilon'} \cdot \frac{\theta_i}{n'}\right) \quad (\text{By } x_i > Shap^{(k)} \ge Shap(t)) \\
&\le \frac{1}{2|T| \cdot n'^{\ell} \log_2 n'} \quad (\text{By ??})
\end{aligned} \tag{132}$$

Note that $\boldsymbol{est}_i^{(k)}$ is the k-th largest value among $\{\boldsymbol{est}_{t,i}\}_{t \in T}$. There are at most $|T|-k$ nodes t with $Shap(t) < Shap^{(k)}$. This implies that there is at least one node t with $Shap(t) \ge Shap^{(k)}$ and $\boldsymbol{est}_{t,i} \le \boldsymbol{est}_i^{(k)}$. More precisely, there are at most k such nodes t with $Shap(t)$ ranked at or above $Shap^{(k)}$. Therefore, we have:

$$\begin{aligned}
&\Pr\left\{\frac{n' \cdot \boldsymbol{est}_i^{(k)}}{\theta_i} \le (1+\varepsilon') \cdot x_i\right\} \\
&\le \Pr\left\{\exists t \in T, Shap(t) \le Shap^{(k)}, \frac{n' \cdot \boldsymbol{est}_{t,i}}{\theta_i} \ge (1+\varepsilon') \cdot x_i\right\} \\
&\le \frac{k}{2|T|n'^{\ell} \log_2 n'}
\end{aligned} \tag{133}$$

Case 2: Consider $x_i \le Shap^{(k)}$. First, for every $t \in T$ such that $Shap(t) \le Shap^{(k)}$, we have:

$$\begin{aligned}
&\Pr\left\{\frac{n' \cdot \boldsymbol{est}_{t,i}}{\theta_i} \ge (1+\varepsilon') \cdot Shap^{(k)}\right\} \\
&= \Pr\left\{\boldsymbol{est}_{t,i} - \theta_i \cdot \frac{Shap^{(k)}}{n'} \le \varepsilon' \frac{\theta_i \cdot Shap^{(k)}}{n'}\right\} \\
&\le \Pr\left\{\boldsymbol{est}_{t,i} - \theta_i \cdot \frac{Shap(t)}{n'} \le (\varepsilon' \cdot \frac{Shap^{(k)}}{Shap(t)})\frac{\theta_i \cdot Shap(t)}{n'}\right\} \\
&\quad (\text{By } Shap(t) \le Shap^{(k)}) \\
&\le \exp\left(-\frac{\left(\varepsilon' \cdot \frac{Shap^{(k)}}{Shap(t)}\right)^2}{2 + \frac{2}{3}\left(\varepsilon' \cdot \frac{Shap^{(k)}}{Shap(t)}\right)} \cdot \theta_i \cdot \frac{Shap(t)}{n'}\right) \quad (\text{By Fact 3}) \\
&= \exp\left(-\frac{\varepsilon'^2 \cdot (Shap^{(k)})^2}{2Shap(t) + \frac{2}{3}\varepsilon' Shap^{(k)}} \cdot \frac{\theta_i}{n'}\right) \\
&\le \exp\left(-\frac{\varepsilon'^2 \cdot Shap^{(k)}}{2 + \frac{2}{3}\varepsilon'} \cdot \frac{\theta_i}{n'}\right) \quad (\text{By } Shap(t) \le Shap^{(k)}) \\
&\le exp\left(-\frac{\varepsilon'^2 \cdot x_i}{2 + \frac{2}{3}\varepsilon'} \cdot \frac{\theta_i}{n'}\right) \quad (\text{By } x_i \le Shap^{(k)}) \\
&\le \frac{1}{2|T| \cdot n'^{\ell} \log_2 n'} \quad (\text{By ??})
\end{aligned} \tag{134}$$

Similarly, by taking the union bound, we have

$$\begin{aligned}
&\Pr\left\{\frac{n' \cdot \boldsymbol{est}_i^{(k)}}{\theta_i} \ge (1+\varepsilon') \cdot Shap^{(k)}\right\} \\
&\le \Pr\left\{\exists t \in T, Shap(t) \le Shap^{(k)}, \frac{n' \cdot \boldsymbol{est}_{t,i}}{\theta_i} \ge (1+\varepsilon') \cdot Shap^{(k)}\right\} \\
&\le \frac{1}{2n'^{\ell} \log_2 n'}
\end{aligned} \tag{135}$$

□

E.2.4 Establishing the Lower Bound $\boldsymbol{LB}$. In Algorithm 4, we computed $\boldsymbol{LB}$ when the condition: $n' \cdot \frac{\boldsymbol{est}^{(k)}}{\theta_i} \ge (1+\varepsilon') \cdot x_i$ is satisfied:

$$\boldsymbol{LB} = n' \cdot \frac{\boldsymbol{est}^{(k)}}{\theta_i(1+\varepsilon')} \tag{136}$$

*

Proof. Let $\boldsymbol{LB}_i = n' \cdot \frac{\boldsymbol{est}_i^{(k)}}{\theta_i \cdot (1+\varepsilon')}$.

case 1: If $Shap^{(k)} \ge x_{\lfloor \log_2 n' \rfloor - 1}$, let i^* be the smallest index such that $x_{i^*} \le Shap^{(k)}$. Thus for all iterations $i < i^*$, we have $x_i > Shap^{(k)}$. Applying **??** (1) on each $i \le i^*$:

$$\Pr\left\{\frac{n' \cdot \boldsymbol{est}_i^{(k)}}{\theta_i} < (1+\varepsilon') \cdot x_i\right\} \ge 1 - \frac{1}{2n'^{\ell} \log_2 n'} \tag{137}$$

Taking the union bound over all iterations $i < i^*$, we have:

$$\Pr\left\{\frac{n' \cdot \boldsymbol{est}_i^{(k)}}{\theta_i} < (1+\varepsilon') \cdot x_i \text{ for all } i < i^*\right\} \geq 1 - \frac{i-1}{2n'^{\ell}\log_2 n'} \tag{138}$$

This means that with probability at least $1 - \frac{i-1}{2n'^{\ell}\log_2 n'}$, the algorithm will not break before the i^*-th iteration. Therefore, $\boldsymbol{LB} = \boldsymbol{LB}_i$ for some $i \geq i^*$ or $\boldsymbol{LB} = 1$.

Then for every $i \geq i^*$, we have $x_i \leq Shap^{(k)}$. Applying **??** (Case 2) on each $i \geq i^*$:

$$\begin{aligned}
&\Pr\left\{\frac{n' \cdot \boldsymbol{est}_i^{(k)}}{\theta_i} < (1+\varepsilon') \cdot Shap^{(k)}\right\} \\
&= \Pr\left\{\boldsymbol{LB}_i < Shap^{(k)}\right\} \\
&\geq 1 - \frac{1}{2n'^{\ell}\log_2 n'}
\end{aligned} \tag{139}$$

Taking the union bound again, we have:

$$\Pr\left\{\boldsymbol{LB} < Shap^{(k)}\right\} \geq 1 - \frac{1}{2n'^{\ell}} \tag{140}$$

Case 2: If $Shap^{(k)} < x_{\lfloor \log_2 n' \rfloor - 1}$, we use the similar argument as the above and we can show that with probability at least $1 - \frac{1}{2n'^{\ell}}$, the for-loop would not break at any iteration and thus $\boldsymbol{LB} = 1$ as the initail value. Since $Shap^{(k)} \geq 1$, we still have $\boldsymbol{LB} \leq Shap^{(k)}$. □

E.2.5 Final Approximation Guarantee. *

Proof. By Lemma **??**, with probability at least $1 - \frac{1}{2n'^{\ell}}$, we have $\boldsymbol{LB} \leq Shap^{(k)}$. From the approximation guarantee established in Lemma 16, with θ set appropriately using $\boldsymbol{LB}$, we have that, with probability at least $1 - \frac{1}{2n'^{\ell}}$, the estimates $\widehat{Shap}(t)$ satisfy the desired bounds **??**.

Then, we use $\boldsymbol{LB}$ in place of $Shap^{(k)}$ to determine the number of RR sets θ in Phase 2 of the algorithm, as per Equation (102). By the union bound, we know that with probability at least $1 - \frac{1}{n'^{\ell}}$, the estimates $\widehat{Shap}(t)$ satisfy **??**.

□

E.3 Time Complexity

In this section, we analyze the time complexity of our algorithm. We establish upper bounds on the expected running time by analyzing the number of RR sets generated in both Phase 1 and Phase 2, and the computational cost associated with generating and processing each RR set.

A random variable $\boldsymbol{\tau}$ is a *stopping time* for martingale $\{\boldsymbol{Y}_i, i \geq 1\}$ if τ takes positive integer values, and the event $\boldsymbol{\tau} = i$ depends only on the values of $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_i$.

Fact 4 (Martingale Stopping Theorem). *Suppose that $\{\boldsymbol{Y}_i, i \geq 1\}$ is a martingale and $\boldsymbol{\tau}$ is a stopping time for $\{\boldsymbol{Y}_i, i \geq 1\}$. If $\tau \leq c$ for some constant c independent of $\{\boldsymbol{Y}_i, i \geq 1\}$, then $\mathbb{E}[\boldsymbol{Y}_\tau] = \mathbb{E}[\boldsymbol{Y}_1]$.*

Given a fixed subset $R \subseteq V$, let the width of R, denoted $\omega(R)$, be the total in-degrees of nodes in R. The time complexity to generate the random *RR* set $\boldsymbol{R}$ is $\Theta(\omega(\boldsymbol{R}) + 1)$. We leave the constant 1 in the above formula because $\omega(\boldsymbol{R})$ could be less than 1 or even $o(1)$ when $m < n$, while $\Theta(1)$ time is needed just to select a random root. The expected time complexity to generate a random RR set is $\Theta(\mathbb{E}[\omega(\boldsymbol{R})] + 1)$.

Let $EPT = \mathbb{E}[\omega(\boldsymbol{R})]$ be the expected width of a random *RR* set. Let $\boldsymbol{\theta'}$ be the random variable denoting the number of RR sets generated in Phase 1.

We first establish Lemma 18 that relates the expected running time of our algorithm to the expected number of RR sets generated and the expected time to generate an RR set. Next, as the time complexity of generating a single RR set is related to the expected width EPT of a random RR set, we relate EPT of a random RR set to the influence spread of a single node in Lemma 19. Then we derive the bounds on the expected number of RR sets generated in both Phase 1 ($\boldsymbol{\theta'}$) and Phase 2 ($\boldsymbol{\theta}$) in Lemma 21. Combining the results from above lemmas, we obtain the overall expected running time of our algorithm in Theorem 1.

We first establish a lemma that relates the expected running time of our algorithm to the expected number of RR sets generated and the expected time to generate an RR set as [10]. The lemma and proof is the same as [10].

Lemma 18. *The expected running time of Algorithm 4 is:*

$$\Theta\left(\left(\mathbb{E}[\boldsymbol{\theta'}] + \mathbb{E}[\boldsymbol{\theta}]\right) \cdot (EPT + 1)\right) \tag{141}$$

where:

- $\boldsymbol{\theta'}$ *is the number of RR sets generated in Phase 1.*
- $\boldsymbol{\theta}$ *is the number of RR sets generated in Phase 2.*
- $EPT = \mathbb{E}[\omega(\boldsymbol{R})]$ *is the expected width of a random RR set* $\boldsymbol{R}$*, with* $\omega(\boldsymbol{R})$ *being the total in-degree of nodes in* $\boldsymbol{R}$*.*

Proof. The proof is the same as [10] and contains the following 4 steps:

(1) Time Complexity of Generating an RR Set

For each RR set $\boldsymbol{R}$, the time to generate $\boldsymbol{R}$ is $\Theta(\omega(\boldsymbol{R}) + 1)$, as the time to select a random root node and perform BFS to find the RR set. Note that the constant term $\Theta(1)$ is not absorbed by $\omega(\boldsymbol{R})$ since $\omega(\boldsymbol{R})$ because the width of the RR set could be less than 1.

(2) Analysis of Phase 1

Let $\boldsymbol{R}_1^{(1)}, \boldsymbol{R}_2^{(1)}, \ldots, \boldsymbol{R}_{\boldsymbol{\theta'}}^{(1)}$ be the RR sets generated in Phase 1. For each RR set $\boldsymbol{R}_j^{(1)}$, we need to update $\boldsymbol{est}_t$ for each $t \in T$ that appears in $\boldsymbol{R}_j^{(1)}$, which takes $\Theta(|\boldsymbol{R}_j^{(1)}|)$ time. Since $T \subseteq V$, and the size of $\boldsymbol{R}_j^{(1)}$ is at most $\omega(\boldsymbol{R}_j^{(1)}) + 1$ (because the induced subgraph is weakly connected), the algorithm takes $\Theta(\omega(\boldsymbol{R}_j^{(1)}) + 1 + |\boldsymbol{R}_j^{(1)}|) = \Theta(\omega(\boldsymbol{R}_j^{(1)}) + 1)$ for each RR set. Therefore, summing up for all $\boldsymbol{\theta'}$ RR sets, the total expected running time of Phase 1 is:

$$\Theta\left(\sum_{j=1}^{\boldsymbol{\theta'}} \left(\omega(\boldsymbol{R}_j^{(1)}) + 1\right)\right) \tag{142}$$

Define $\boldsymbol{W}_i = \sum_{j=1}^{i} \left(\omega(\boldsymbol{R}_j^{(1)}) - EPT\right)$ for $i \geq 1$. By an argument similar to that in Lemma 17, $\{\boldsymbol{W}_i, i \geq 1\}$ is a martingale. The stopping time $\boldsymbol{\theta'}$ is upper bounded by a constant $\theta_{\lfloor \log_2 n' \rfloor - 1}$, and depends only on the RR sets already generated. Therefore, by Fact 4:

$$\mathbb{E}[\boldsymbol{W}_{\boldsymbol{\theta'}}] = \mathbb{E}[\boldsymbol{W}_1] = 0 \tag{143}$$

Thus:

$$\mathbb{E}\left[\sum_{j=1}^{\boldsymbol{\theta}'}\omega(\boldsymbol{R}_j^{(1)})\right]-\mathbb{E}[\boldsymbol{\theta}']\cdot EPT=0 \tag{144}$$

Rearranging:

$$\mathbb{E}\left[\sum_{j=1}^{\boldsymbol{\theta}'}\omega(\boldsymbol{R}_j^{(1)})\right]=\mathbb{E}[\boldsymbol{\theta}']\cdot EPT \tag{145}$$

Therefore, the expected running time of Phase 1 is:

$$\Theta\left(\mathbb{E}[\boldsymbol{\theta}']\cdot(EPT+1)\right). \tag{146}$$

(3) Analysis of Phase 2

Similarly, in Phase 2, we generate $\boldsymbol{\theta}$ RR sets independently. The expected running time of Phase 2 is:

$$\Theta\left(\mathbb{E}[\boldsymbol{\theta}]\cdot(EPT+1)\right) \tag{147}$$

(4) Total Expected Running Time

Combining both phases, the total expected running time is:

$$\Theta\left(\left(\mathbb{E}[\boldsymbol{\theta}']+\mathbb{E}[\boldsymbol{\theta}]\right)\cdot(EPT+1)\right) \tag{148}$$

□

E.3.1 Expected Width of Random RR Sets. Next, we relate the expected width EPT of a random RR set to the value function of a single node. The argument and analysis is the same as [10] except that we replace n with n'.

Lemma 19. *Let $\tilde{v}$ be a random node drawn from $V\setminus T$ with probability proportional to the in-degree of $\tilde{v}$ in G'. Let $\boldsymbol{R}$ be a random RR set generated in G'. Then:*

$$EPT=\mathbb{E}_{\boldsymbol{R}}[\omega(\boldsymbol{R})]=\frac{m'}{n'}\cdot\mathbb{E}_{\tilde{\boldsymbol{v}}}[U(\tilde{v})], \tag{149}$$

where:

- *m' is the number of edges in G'.*
- *$n'=|V\setminus T|$.*
- *$U(\tilde{v})$ is the expected value function of $\tilde{v}$.*

Proof. For a fixed set $R\subseteq V$, let $p(R)$ be the probability that a randomly selected edge in G' points to a node in R. Since R has $\omega(R)$ edges pointing to nodes in R, and the total number of edges in G' is m', we have that $p(R)=\frac{\omega(R)}{m'}$.

Denote d_v as the in-degree of a node v. We have:

$$\begin{aligned}p(R)&=\sum_{(u,v)\in E'}\frac{1}{m'}\cdot\mathbb{I}\{v\in R\}\\&=\sum_{v\in V}\frac{d_v}{m'}\cdot\mathbb{I}\{v\in R\}=\mathbb{E}_{\tilde{\boldsymbol{v}}}[\mathbb{I}\{\tilde{\boldsymbol{v}}\in R\}]\end{aligned} \tag{150}$$

Then for a random RR set $\boldsymbol{R}$, we have:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{R}}[\omega(\boldsymbol{R})]&=m'\cdot\mathbb{E}_{\boldsymbol{R}}[p(\boldsymbol{R})]\\&=m'\cdot\mathbb{E}_{\boldsymbol{R}}\left[\mathbb{E}_{\tilde{\boldsymbol{v}}}[\mathbb{I}\{\tilde{\boldsymbol{v}}\in\boldsymbol{R}\}]\right]\\&=m'\cdot\mathbb{E}_{\hat{\boldsymbol{v}}}\left[\mathbb{E}_{\boldsymbol{R}}[\mathbb{I}\{\tilde{\boldsymbol{v}}\in\boldsymbol{R}\}]\right]\\&=m'\cdot\mathbb{E}_{\hat{\boldsymbol{v}}}\left[\Pr(\tilde{\boldsymbol{v}}\in\boldsymbol{R})\right]\end{aligned} \tag{151}$$

Recall that $U(S)=n'\cdot\Pr[S\cap\boldsymbol{R}\neq\emptyset]$. Therefore, when the subset S is a single node $\tilde{v}$, we have $\Pr(\tilde{\boldsymbol{v}}\in\boldsymbol{R})=\frac{U(\tilde{\boldsymbol{v}})}{n'}$. Therefore, we have:

$$EPT=\mathbb{E}_{\boldsymbol{R}}[\omega(\boldsymbol{R})]=m'\cdot\mathbb{E}_{\hat{\boldsymbol{v}}}[\frac{U(\tilde{\boldsymbol{v}})}{n'}]=\frac{m'}{n'}\cdot\mathbb{E}_{\hat{\boldsymbol{v}}}[U(\tilde{\boldsymbol{v}})] \tag{152}$$

□

E.3.2 Bounding $\mathbb{E}[\boldsymbol{\theta}']$ and $\mathbb{E}[\boldsymbol{\theta}]$. We now derive bounds on $\mathbb{E}[\boldsymbol{\theta}']$ and $\mathbb{E}[\boldsymbol{\theta}]$, the expected numbers of RR sets generated in Phases 1 and 2, respectively. First, we prove the following two inequalities.

Lemma 20. *For each $i=1,2,\ldots,\lfloor\log_2 n'\rfloor-1$, if $Shap^{(k)}\geq(1+\varepsilon')^2\cdot x_i$, then the following holds with probability at least $1-\frac{k}{2|T|n'^{\ell}\log_2 n'}$:*

$$\frac{n'\cdot\boldsymbol{est}_i^{(k)}}{\theta_i}>\frac{Shap^{(k)}}{1+\varepsilon'} \tag{153}$$

$$\frac{n'\cdot\boldsymbol{est}_i^{(k)}}{\theta_i}>(1+\varepsilon')\cdot x_i \tag{154}$$

Proof. Let $\boldsymbol{R}_1^{(1)},\boldsymbol{R}_2^{(1)},\ldots,\boldsymbol{R}_{\theta_i}^{(1)}$ be the θ_i generated RR sets by the end of the i-th iteration of the for-loop in Phase 1. Recall that $\boldsymbol{est}_{t,i}=\sum_{j=1}^{\theta_i}X_{\boldsymbol{R}_j^1}(t)$.

If $Shap^{(k)}\geq(1+\varepsilon')^2\cdot x_i$, then we can apply Fact 3 and have:

$$\begin{aligned}&\Pr\left[n'\cdot\frac{\boldsymbol{est}_{t,i}}{\theta_i}\leq\frac{Shap(t)}{1+\varepsilon'}\right]\\&=\Pr\left[\boldsymbol{est}_{t,i}-\theta_i\cdot\frac{Shap(t)}{n'}\leq-\frac{\varepsilon'}{1+\varepsilon'}\cdot\theta_i\cdot\frac{Shap(t)}{n'}\right]\\&\leq\exp\left[-\frac{\varepsilon'^2}{2(1+\varepsilon')^2}\cdot\theta_i\cdot\frac{Shap(t)}{n'}\right]\quad\text{(By Fact 3)}\\&\leq\exp\left[-\frac{\varepsilon'^2\cdot x_i}{2n'}\cdot\theta_i\right]\quad\left(\text{By }x_i\leq\frac{Shap^{(k)}}{(1+e')^2}\leq\frac{Shap(t)}{(1+\varepsilon')^2}\right)\\&\leq\frac{1}{2|T|n'^{\ell}\log_2 n'}\end{aligned} \tag{155}$$

Then we take the union bound to obtain Eq. (153). Note that there is at least one node t with $Shap(t)\geq Shap^{(k)}$ and $\boldsymbol{est}_{t,i}\leq\boldsymbol{est}_i^{(k)}$. More precisely, there are at most k such nodes t with $Shap(t)$ ranked at or above $Shap^{(k)}$. Therefore, we have:

$$\begin{aligned}&\Pr\left[n'\cdot\frac{\boldsymbol{est}_i^{(k)}}{\theta_i}\leq\frac{Shap^{(k)}}{1+\varepsilon'}\right]\\&\leq\Pr\left[\exists t\in T,Shap(t)\geq Shap^{(k)},n'\cdot\frac{\boldsymbol{est}_{t,i}}{\theta_i}\leq\frac{Shap(t)}{(1+\varepsilon')}\right]\\&\leq k\Pr\left[n'\cdot\frac{\boldsymbol{est}_{t,i}}{\theta_i}\leq\frac{Shap(t)}{(1+\varepsilon')}\right]\\&\leq\frac{k}{2|T|n'^{\ell}\log_2 n'}\end{aligned} \tag{156}$$

Last, by applying $Shap^{(k)}\geq(1+\varepsilon')^2\cdot x_i$ to Eq. (153), we can also obtain Eq. (154). □

Then we show the expected number of RR sets generated in both Phase 1 and Phase 2.

Lemma 21. *Under our algorithm, the expected number of RR sets generated satisfies:*

$$\mathbb{E}[\boldsymbol{\theta}'] = O\left(\frac{n\ell \log n}{Shap^{(k)}\varepsilon^2}\right) \tag{157}$$

$$\mathbb{E}[\boldsymbol{\theta}] = O\left(\frac{n\ell \log n}{Shap^{(k)}\varepsilon^2}\right) \tag{158}$$

provided that $\ell \geq \dfrac{\log_2 k - \log_2 \log_2 n' + \log_2 n' - \log_2 |T|}{\log_2 n'}$.

Proof. We first prove for $\boldsymbol{\theta}'$. We consider two cases seperately and then combine them: $Shap^{(k)} < (1+\varepsilon')^2 \cdot x_{\lfloor \log_2 n' \rfloor - 1}$ and $Shap^{(k)} \geq (1+\varepsilon')^2 \cdot x_{\lfloor \log_2 n' \rfloor - 1}$.

Case 1: $Shap^{(k)} < (1+\varepsilon')^2 \cdot x_{\lfloor \log_2 n' \rfloor - 1}$. Recall that θ_i in Phase 1 is:

$$\theta_i = \left\lceil \frac{n'(2+\frac{2}{3}\varepsilon')}{\varepsilon'^2 \cdot x_i} \left(\ell \ln n' + \ln|T| + \ln(\log_2 n') + \ln 2\right) \right\rceil \tag{159}$$

Since $x_{\lfloor \log_2 n' \rfloor - 1} = \frac{n'}{2^{\lfloor \log_2 n' \rfloor - 1}} \leq 2$, we have $Shap^{(k)} < 4(1+\varepsilon')^2$. Thus in the worst case we can bound $\boldsymbol{\theta}'$ based on $Shap^{(k)}$ as:

$$\begin{aligned}
\boldsymbol{\theta}' &= \theta_{\lfloor \log_2 n' \rfloor - 1} && (160)\\
&\leq \left\lceil \frac{n'\left(\ell \ln n' + \ln|T| + \ln\log_2 n' + \ln 2\right)\left(2+\frac{2}{3}\varepsilon'\right)}{\varepsilon'^2} \right\rceil && (161)\\
&\leq \left\lceil \frac{n'\left(\ell \ln n' + \ln|T| + \ln\log_2 n' + \ln 2\right)\left(2+\frac{2}{3}\varepsilon'\right)\cdot 4(1+\varepsilon')^2}{\varepsilon'^2 \cdot Shap^{(k)}} \right\rceil && (162)\\
&= O\left(\frac{n'(\ell \log n' + \log|T|)}{Shap^{(k)}\varepsilon^2}\right) && (163)\\
&= O\left(\frac{n\ell \log n}{Shap^{(k)}\varepsilon^2}\right) && (164)
\end{aligned}$$

From Eq. (162) to Eq. (163) follows the fact that $\varepsilon' = \sqrt{2}\cdot\varepsilon$ and ε is a sufficiently small positive parameter, so $\left(2+\frac{2}{3}\varepsilon'\right)\cdot 4(1+\varepsilon')^2$ remains a constant. Moreover, we can relax the bound from Eq. (163) to Eq. (164) in terms of the size of the network n.

Similarly, for $\boldsymbol{\theta}$, since $\boldsymbol{LB} \geq 1$, we have:

$$\begin{aligned}
\boldsymbol{\theta} &\leq \left\lceil \frac{n'\left(\ell \ln n' + \ln|T| + \ln 4\right)\left(2+\frac{2}{3}\varepsilon'\right)}{\varepsilon^2} \right\rceil\\
&\leq \left\lceil \frac{n'\left(\ell \ln n' + \ln|T| + \ln 4\right)\left(2+\frac{2}{3}\varepsilon'\right)4(1+\varepsilon')^2}{\varepsilon^2 \cdot Shap^{(k)}} \right\rceil && (165)\\
&= O\left(\frac{n'(\ell \log n' + \log|T|)}{Shap^{(k)}\varepsilon^2}\right)\\
&= O\left(\frac{n\ell \log n}{Shap^{(k)}\varepsilon^2}\right)
\end{aligned}$$

Case 2: $Shap^{(k)} \geq (1+\varepsilon')^2 \cdot x_{\lfloor \log_2 n' \rfloor - 1}$

Let i^* be the smallest index such that $(1+\varepsilon')^2 \cdot x_{i^*} \leq Shap^{(k)}$. Therefore, $Shap^{(k)} < (1+\varepsilon')^2 \cdot x_{i^*-1}$. Here we denote $x_0 = n'$.

By the Eq. (154) in Lemma 20, we know that with probability at least $1 - \frac{k}{2|T|n'^{\ell}\log_2 n'}$, Phase 1 will stop at the i^*-th iteration. Similarly, we can bound $\boldsymbol{\theta}'$ as:

$$\begin{aligned}
\boldsymbol{\theta}' = \theta_{i^*} &= \left\lceil \frac{n'\left(\ell \ln n' + \ln|T| + \ln\log_2 n' + \ln 2\right)\left(2+\frac{2}{3}\varepsilon'\right)}{\varepsilon'^2 x_{i^*}} \right\rceil\\
&\leq \left\lceil \frac{n'\left(\ell \ln n' + \ln|T| + \ln\log_2 n' + \ln 2\right)\left(2+\frac{2}{3}\varepsilon'\right)(1+\varepsilon')^2}{\varepsilon'^2 Shap^{(k)}} \right\rceil\\
&= O\left(\frac{n'(\ell \log n' + \log|T|)}{Shap^{(k)}\varepsilon^2}\right)\\
&= O\left(\frac{n\ell \log n}{Shap^{(k)}\varepsilon^2}\right)
\end{aligned} \tag{166}$$

By the Eq. (153) in Lemma 20, when Phase 1 stops at the i^*-th iteration, the $\boldsymbol{LB}$ satisfies:

$$\boldsymbol{LB} = \frac{n' \cdot \boldsymbol{est}_{i^*}^{(k)}}{\theta_{i^*}\cdot(1+\varepsilon')} \geq \frac{Shap^{(k)}}{(1+\varepsilon')^2} \tag{167}$$

Then we can bound $\boldsymbol{\theta}$ as:

$$\begin{aligned}
\boldsymbol{\theta} &\leq \left\lceil \frac{n'\left(\ell \ln n' + \ln|T| + \ln 4\right)\left(2+\frac{2}{3}\varepsilon\right)(1+\varepsilon')^2}{\varepsilon^2 Shap^{(k)}} \right\rceil\\
&= O\left(\frac{n'(\ell \log n' + \log|T|)}{Shap^{(k)}\varepsilon^2}\right) && (168)\\
&= O\left(\frac{n\ell \log n}{Shap^{(k)}\varepsilon^2}\right)
\end{aligned}$$

Moreover, Phase 1 will not stop at the i^*-th iteration with probability at most $\frac{k}{2|T|n'^{\ell}\log_2 n'}$. In the worst case, it continues to iteration $\lfloor \log_2 n' \rfloor - 1$, and $\boldsymbol{\theta}' = O\left(\frac{n'(\ell \log n' + \log|T|)}{\varepsilon^2}\right) = O\left(\frac{n\ell \log n}{\varepsilon^2}\right)$. Combine with Eq. (166), because the fact that $Shap^{(k)} \leq n'$, and the condition that $\ell \geq \dfrac{\log_2 k - \log_2 \log_2 n' + \log_2 n' - \log_2 |T|}{\log_2 n'}$, we

have:

$$\begin{aligned}\mathbb{E}[\boldsymbol{\theta}'] &= O\left(\frac{n\ell\log n}{Shap^{(k)}\varepsilon^2}\right) + \frac{k}{2|T|n'^{\ell}\log_2 n'}\cdot O\left(\frac{n\ell\log n}{\varepsilon^2}\right)\\ &= O\left(\frac{n\ell\log n}{Shap^{(k)}\varepsilon^2}\right)\end{aligned} \tag{169}$$

Similarly, the worst case in Phase 2 is that $\boldsymbol{LB} = 1$, and we have: $\boldsymbol{\theta} = O\left(\frac{n'(\ell\log n'+\log|T|)}{\varepsilon^2}\right) = O\left(\frac{n\ell\log n}{\varepsilon^2}\right)$. Combine with Eq. (168), we have:

$$\begin{aligned}\mathbb{E}[\boldsymbol{\theta}] &= O\left(\frac{n\ell\log n}{Shap^{(k)}\varepsilon^2}\right) + \frac{k}{2|T|n'^{\ell}\log_2 n'}\cdot O\left(\frac{n\ell\log n}{Shap^{(k)}\varepsilon^2}\right)\\ &= O\left(\frac{n\ell\log n}{Shap^{(k)}\varepsilon^2}\right)\end{aligned} \tag{170}$$

Note that we set $\varepsilon' = \sqrt{2}\cdot\varepsilon$ as suggested in [47]. Moreover, the condition on ℓ ensures the probability that Phase 1 will not stop at the i^*-th iteration is at most $\frac{1}{2n'}$.

□

E.3.3 Final Time Complexity. Combining the results from Lemmas 18, 19, and 21, we obtain the overall expected running time of our algorithm.

Theorem 1. *The expected running time of our algorithm is:*

$$O\left(\frac{n\ell\log n}{Shap^{(k)}\varepsilon^2}\cdot\frac{m'}{n'}\cdot\mathbb{E}_{\tilde{v}}[U(\tilde{v})]\right) \tag{171}$$

Under the condition that

$$\ell \geq \frac{\log_2 k - \log_2\log_2 n' + \log_2 n' - \log_2|T|}{\log_2 n'}$$

Proof. From Lemma 18, the expected running time is:

$$\Theta\left(\left(\mathbb{E}[\boldsymbol{\theta}'] + \mathbb{E}[\boldsymbol{\theta}]\right)\cdot(EPT+1)\right) \tag{172}$$

Substituting $\mathbb{E}[\boldsymbol{\theta}']$ and $\mathbb{E}[\boldsymbol{\theta}]$ from 21, we have:

$$\mathbb{E}[\boldsymbol{\theta}'] + \mathbb{E}[\boldsymbol{\theta}] = O\left(\frac{n\ell\log n}{Shap^{(k)}\varepsilon^2}\right) \tag{173}$$

From Lemma 19, we have:

$$\text{EPT} = \frac{m'}{n'}\cdot\mathbb{E}_{\tilde{v}}[U(\tilde{v})] \tag{174}$$

Combining these, the expected running time is:

$$O\left(\frac{n\ell\log n}{Shap^{(k)}\varepsilon^2}\cdot\frac{m'}{n'}\cdot\mathbb{E}_{\tilde{v}}[U(\tilde{v})]\right) \tag{175}$$

This completes the proof. □