ProMotE: an efficient algorithm for counting independent motifs in uncertain network topologies

Yuanfang Ren*, Aisharjya Sarkar and Tamer Kahveci

Abstract

Background: Identifying motifs in biological networks is essential in uncovering key functions served by these networks. Finding non-overlapping motif instances is however a computationally challenging task. The fact that biological interactions are uncertain events further complicates the problem, as it makes the existence of an embedding of a given motif an uncertain event as well.

Results: In this paper, we develop a novel method, ProMotE (Probabilistic Motif Embedding), to count non-overlapping embeddings of a given motif in probabilistic networks. We utilize a polynomial model to capture the uncertainty. We develop three strategies to scale our algorithm to large networks.

Conclusions: Our experiments demonstrate that our method scales to large networks in practical time with high accuracy where existing methods fail. Moreover, our experiments on cancer and degenerative disease networks show that our method helps in uncovering key functional characteristics of biological networks.

Keywords: Independent motif counting, Probabilistic networks, Polynomial

Background

Biological networks describe a system of interacting molecules. Through these interactions, these molecules carry out key functions such as regulation of transcription and transmission of signals [1]. Biological networks are often modeled as graphs, with nodes and edges representing interacting molecules (e.g., protein or gene) and the interactions between them respectively [2–4]. Studying biological networks has great potential to provide significant new insights into systems biology [5, 6].

Network motifs are patterns of local interconnections occurring significantly more in a given network than in a random network of the same size [7]. Identifying motifs is crucial to uncover important properties of biological networks. They have already been successfully used in many applications, such as understanding important genes that affect the spread of infectious diseases [8], revealing relationships across species [6, 9], and discovering processes which regulate transcription [10].

Network motif discovery is a computationally hard problem as it requires solving the well-known subgraph isomorphism problem, which is NP-complete [11]. The fact that biological interactions are often inherently stochastic events further complicates the problem [12]. An interaction may or may not happen with some probability. This uncertainty follows from the fact that biological processes governing these interactions, such as DNA replication process, inherently exhibit uncertainties. For example, DNA replication can initiate at different chromosome locations with various probabilities [13]. Besides the replication time variance, other epigenetic factors can also alter the expression levels of genes, which in turn affect the ability of proteins to interact [14].

Existing studies model the uncertainty of biological interactions using a probability value showing the confidence in its presence [12]. More specially, each edge in the network is associated with a probability value. Several databases, such as MINT [15] and STRING [16], already provide interaction confidence values. If a biological network has at least one uncertain interaction, we call it a
probabilistic network. Otherwise, it is a deterministic network. In the rest of the paper, we represent a probabilistic network using a graph denoted with $G = (V, E, P)$, where $V$ denotes the set of interacting molecules, $E$ denotes the interactions among them, and $P : E \to (0,1]$ is the function that assigns a probability value to each edge.

Several approaches have been developed to solve the network motif discovery problem (e.g., [17–19]). However, most of them focus on deterministic network topologies. The main reason behind this limitation is that a probabilistic network summarizes all deterministic networks generated by all possible subsets of interactions. Thus, a probabilistic network $G = (V, E, P)$ yields $2^{|E|}$ deterministic instances. The exponential growth of the number of deterministic instances makes it impossible to directly apply existing solutions to probabilistic networks. Relatively little research has been done on finding motifs in probabilistic networks. Tran et al. [20] proposed a method to derive a set of formulas for count estimation. This study however has not provided a general mathematical formulation for arbitrary motif topologies. It rather requires a unique mathematical formulation for each motif. Besides, it assumes that all interactions of the probabilistic network have the same probability. Thus, it fails to solve the generalized version of the problem where each interaction takes place with a possibly different probability. Todor et al. [21] developed a method to solve the generalized version of the problem. It computes the exact mean and variance of the number of motif instances. Both of above two methods count the maximum number of motif instances using $F_1$ measure, that is including all possible embeddings regardless of whether they overlap with each other or not.

There are two more restrictive frequency measures, $F_2$ and $F_3$, which avoid reuse of graph elements [19]. $F_2$ measure considers that two embeddings of a motif overlap if they share an edge. $F_3$ measure is more restrictive as it defines overlap as sharing of a node. These two measures count the maximum number of non-overlapping embeddings of a given motif. We explain the difference among three frequency measures on a hypothetical deterministic network $G^0$ (see Fig. 1a). Consider the motif pattern $M$ in Fig. 1b. $G^0$ yields six possible embeddings of $M$ denoted with the embedding set $\mathcal{H} = \{H_1, H_2, H_5, H_4, H_5, H_6\}$ (see Fig. 1c-h). Since $F_1$ measure counts all possible embeddings, the $F_1$ count is six. As embeddings $H_1$ and $H_6$ do not have common edges, the $F_2$ count is two. All pairs of embeddings in this set share nodes. As a result, the $F_3$ count is one.

$F_2$ and $F_3$ measures satisfy a fundamental characteristic, the downward closure property, which $F_1$ measure fails to have. This property is essential for constructing large motifs [22]. It ensures that the frequency of network motifs is monotonically decreasing with increasing size of motif patterns. For example, in the deterministic network $G^0$ (see Fig. 1a), given the triangle pattern (see Fig. 1i), there are two triangle embeddings in total (Fig. 1j). Consider a larger motif pattern, such as the pattern in Fig. 1b. The $F_1$ count however becomes six, which conflicts with the downward closure property. Besides, non-overlapping motifs are needed in navigation methods such as folding and unfolding of the network [23].

Fig. 1 An example to explain three frequency measures. a A hypothetical deterministic network $G^0$ with seven nodes and eight edges. b A motif pattern $M$ with four nodes and three edges. c–h Six possible embeddings of motif pattern $M$ in network $G^0$ denoted with the embedding set $\mathcal{H} = \{H_1, H_2, H_5, H_4, H_5, H_6\}$. i A triangle pattern. j An embedding set of triangle pattern
the importance of non-overlapping motifs into account, Sarkar et al. [24] developed a method to count the non-overlapping motifs in probabilistic networks using the $F_2$ measure. Their study builds a polynomial to model the distribution of the number of motif instances overlapping with a specific embedding of that motif. However, the exponential growth of the size of polynomial terms makes it not scalable to large networks.

Contributions. In this paper, we develop a scalable method, named ProMotE (Probabilistic Motif Embedding), to tackle the problem of counting independent motifs in a given probabilistic network. We formally define the problem in “Preliminaries and problem definition” section. We explain our method for the $F_2$ measure, yet the same algorithm can trivially be applied to the $F_3$ measure. This study has three major contributions over the existing literature: (1) The key bottleneck in counting motifs in probabilistic networks is computing the distribution of the number of overlapping embeddings of a given motif instance. We build a new method which allows us to avoid computing this distribution whenever possible. (2) Computing the distribution in (1) above necessitates constructing a polynomial. We devise two strategies, which compute bounds to the overlapping motif count distribution prior to constructing the entire polynomial. These bounds enable us to terminate the costly computation of the distribution whenever possible. (3) We develop a new strategy which allows multiplication of arbitrarily large polynomials using a limited amount of memory. Our experimental results demonstrate that our algorithm is orders of magnitude faster than existing methods. Our results on cancer and disease networks suggest that our method can help in uncovering key functional characteristics of the genes participating in those networks.

We organize the rest of the paper as follows. We present our algorithm in “Methods” section. We discuss our experimental results in “Results and discussion” section and conclude in “Conclusions” section.

Methods
In this section, we present our method, ProMotE. First, we formally define the independent motif counting problem in probabilistic networks (“Preliminaries and problem definition” section). We next summarize the method by Sarkar et al. [24] (“Overview of the existing solution” section). We then present the method developed in this paper. Our method introduces three strategies (Sections “Avoiding loss computation”, “Efficient polynomial collapse” and “Overcoming memory bottleneck”), which help us scale to large network size, for which existing methods fail.

Preliminaries and problem definition
In this section we present basic notation needed to define the problem considered in this paper. We denote the given probabilistic network and motif pattern with $G = (V, E, P)$ and $M$ respectively. For each edge $e_i$ in $G$, we denote the probability that $e_i$ is present and absent with $p_i$ and $q_i$ respectively (i.e., $p_i + q_i = 1$). We denote the set of all possible deterministic network topologies one can observe from $G$ with $D(G) = \{G^o = (V, E^o)| E^o \subseteq E\}$. We denote a specific deterministic network which inherits all nodes and edges from $G$ but assume that all of its edges exist with $G^o = (V, E)$. Figure 2 depicts a probabilistic network and its three possible deterministic networks (i.e., in total there are $2^8 = 256$ deterministic networks). We denote the probability of observing a specific deterministic network $G^o \in D(G)$ with

$$P(G^o|G) = \prod_{e_i \in E^o} p_i \prod_{e_j \notin E^o} q_j$$

Given a deterministic network $G^o = (V, E^o)$ and a motif pattern $M$, we represent the set of all its embeddings with $H(M|G^o)$. We construct the overlap graph for $H(M|G^o)$, denoted with $G^o$, by representing each embedding $H_k \in H(M|G^o)$ as a node and inserting an edge into two nodes if their corresponding embeddings share at least one edge. Thus, for a specific embedding $H_k$, the degree of its corresponding node in $G^o$ equals the number of embeddings overlapping with $H_k$. Figure 3 depicts the overlap graph of the embeddings found in deterministic network $G^o$ shown in Fig. 1. Consider a subset of embeddings $H^o \subseteq H(M|G^o)$. We define an indicator function $\xi()$ on $H^o$ as follows: $\xi(H^o) = 1$ if no two embeddings in $H^o$ share an edge, and $\xi(H^o) = 0$ otherwise.

Consider a specific embedding $H_k$ in $G$. Because of the uncertain nature of the probabilistic network, each embedding exists with a probability value. As a result, the number of embeddings overlapping with $H_k$ is also uncertain. We represent it using a random variable $B_k$. To
calculate the distribution of $B_k$, we construct a bipartite graph denoted with $G_k = (V_1, V_2, \mathcal{E})$. $V_1$ and $V_2$ represent two node sets, and $\mathcal{E}$ represents the edges connecting nodes of $V_1$ with those of $V_2$. Each neighboring node of $H_k$ in the overlap graph corresponds to a node in $V_1$. Each edge in the edge set, which constitutes all those overlapping embeddings of $H_k$, corresponds to a node in $V_2$. Notice that this edge set excludes the edges of embedding $H_k$ itself. An edge exists between nodes $u \in V_1$ and $v \in V_2$ if the corresponding embedding of node $u$ has the edge denoted by $v$. Figure 4 shows the bipartite graph $G_4$ of embedding $H_4$ (see Fig. 1). $H_1, H_2, H_3, H_5$ and $H_6$ are neighbors of $H_4$ in the overlap graph $\mathcal{G}$ (see Fig. 3). Thus these embeddings are nodes in $V_1$ of $G_4$. Their edges include $e_1, e_2, e_3, e_4, e_5, e_6, e_7$ and $e_8$. As edges $e_3, e_5, e_6$ and $e_7$ are also edges of $H_4$, only $e_1, e_2, e_4$ and $e_8$ constitute $V_2$ of $G_4$.

To help better understand this paper, we introduce another two notations $x$-polynomial and collapse operator. Given a bipartite graph $G_k$, we compute a polynomial, called the $x$-polynomial as follows. For each node $v_i \in V_1$, it defines a unique variable $x_i$. For each node $v_j \in V_2$, the probability that $v_j$’s corresponding edge is present and absent is $p_j$ and $q_j$ ($q_j = 1 - p_j$) respectively. For each node $v_j \in V_2$, we construct a polynomial called edge polynomial $Z_j$ as

$$Z_j = p_j \prod_{(v_i, v_j) \in \mathcal{E}} x_i + q_j. \quad (1)$$

The first term of this edge polynomial consists of the product of the variables of those overlapping embeddings containing this edge. The second term only has the probability of the absence of this edge. We explain the concept of edge polynomial using the example of the bipartite graph in Fig. 4. In this example, the edge polynomial for edge $e_1$ is $Z_1 = p_1 x_1 + q_1$. Also the edge polynomial corresponding to $e_2$ is $Z_2 = p_2 x_1 x_2 x_3 + q_2$. The first term of this edge polynomial represents the case that when edge $e_2$ is present, it contributes to the existence of embeddings $H_1, H_2$ and $H_3$ with a probability $p_2$. The second term however represents the case that when edge $e_2$ is absent with probability $q_2$, none of those three embeddings exist. We compute the $x$-polynomial of $H_k$ denoted with $Z_{\mathcal{H}_k}$ as

$$Z_{\mathcal{H}_k} = \prod_{v_j \in V_2} Z_j. \quad (2)$$
The key characteristic of the x-polynomial in the above equation is that its terms model all possible deterministic network topologies for the edges denoted by $\mathcal{V}_2$. We write the $j$th term of the x-polynomial as $a_j \prod_{v \in \mathcal{V}_1} x_i^c$, where $a_j$ is the probability and $c_j$ is the exponent of the variable $x_i$. To compute this polynomial faster, we introduce a collapse operator for each variable $x_i$, denoted with $\phi_i()$, as follows. Let us denote the degree of $v_j \in \mathcal{V}_1$ with $\deg(v_j|G_k)$. For each node's unique variable $x_i$, we define an indicator function $\psi(c)$, where $\psi(c) = 1$ if $c = \deg(v_j|G_k)$, otherwise $\psi(c) = 0$. Using these notations, for the $j$th term of the x-polynomial, we compute collapse operator $\phi_i()$ as

$$
\phi_i \left( a_j \prod_{v \in \mathcal{V}_1} x_i^c \right) = [t \psi(c_{i_j}) + (1 - \psi(c_{i_j}))] a_j \prod_{v \in \mathcal{V}_1 - \{v_j\}} x_i^c.
$$

(3)

Notice that, the collapse operator $\phi_i$ only changes the variable $x_i$. It either replaces it with $t$ or completely removes it depending on the outcome of $\psi_i()$. When $\psi_i() = 1$ (i.e., $c_{i_j} = \deg(v_j|G_k)$), it means that all edges of embedding $H_r$ are present (e.g., $H_r$ exists). Thus, the variable $t$ replaces $x_i$, which means a motif is present. When $\psi_i() = 0$, it indicates that at least one edge of $H_r$ is absent. Thus, the entire $H_r$ is missing. For this reason, consider one of the terms resulting from the product of all degree polynomials in $\mathcal{Z}_{H_r}$, $q_1 p_2 p_4 q_8 x_1^2 x_2^2 x_3$. If we apply the collapse operator $\phi_1()$ to this term, the variable $x_1$ will be removed as $\psi_1() = 0$. Similarly, if we apply the collapse operator $\phi_2()$ to this term, the variable $x_2$ will be replaced with $t$ as $\psi_2() = 1$. After applying all collapse operators to this term, it becomes $q_1 p_2 p_4 q_8 t^3$ which indicates that when only edges $e_2$ and $e_4$ are present, there are three embeddings remaining. And this case happens with a probability $q_1 p_2 p_4 q_8$. We apply the collapse operator $\phi_r$ to the polynomial term as soon as it completes multiplication of the final edge polynomial of the variable $x_r$, which means that no other edge polynomial can increase the exponent of $x_r$.

Given these definitions, we formally define two different independent motif counting problems next.

**Definition 1 (Independent Motif Counting in Probabilistic Network I).** Given a probabilistic network $G = (V, E, P)$ and a motif pattern $M$, find a set of independent embeddings which yields the maximum expected number of occurrences in $G$, which is

$$
\arg\max_{\mathcal{H} \subseteq \mathcal{H}(G')} \left\{ \sum_{G' \in \mathcal{D}(G)} |\mathcal{H}(M|G') \cap \mathcal{H}'| \cdot \mathcal{P}(G'|G) \right\}.
$$

(4)

We explain the problem on a hypothetical probabilistic network $G$ (see Fig. 2). To better explain the problem, we also list some possible deterministic networks in Fig. 2. Notice that this probabilistic network has the same network topology as the deterministic network $G'$ in Fig. 1a. As a result, $G$ has six possible embeddings same with $G'$, which are $H_1, H_2, H_3, H_4, H_5$ and $H_6$ (see Fig. 1c-h). According to the problem definition, we seek to find a set of non-overlapping embeddings which contributes to the maximum expected number of motif count over all possible deterministic network topologies. For those six embeddings of $G$, we are able to construct five sets of independent embeddings, which are $\{H_1, H_6\}, \{H_2\}, \{H_3\}, \{H_4\}$ and $H_5$ (see Fig. 3 for the relationship between embeddings). For each set, we summarize the expected motif count over the set of all alternative deterministic network topologies based on Eq. 4. Table 1 lists the result.

Then, we choose the set with maximum motif count. Notice that, the resulting embedding set with the maximum expected motif count is not guaranteed to always have the largest motif frequency among all possible deterministic networks. For example, in deterministic network $G_1$, the set $\{H_1, H_6\}$ has the highest motif frequency; while in network $G_2$, it is the set $\{H_2\}$ that achieves the largest motif count. By requiring to select the set of embeddings with highest frequency in each possible deterministic network, we have our second independent motif counting problem. We formally define it next.

**Definition 2 (Independent Motif Counting in Probabilistic Network II).** Given a probabilistic network $G = (V, E, P)$ and a motif pattern $M$, compute the...

---

**Table 1** {$\{H_1, H_6\}$, {$H_2$}, {$H_3$}, {$H_4$} and {$H_5$} are the five possible independent embedding sets of the motif $M$ (Fig. 1) in network $G$ (Fig. 2). The table shows the number of embeddings occurring at each deterministic network for each independent embedding set and its expected value in $G$.

| Set | $G_1$ | $G_2$ | $G_3$ | ... | Expected motif count |
|-----|-------|-------|-------|-----|---------------------|
| {$H_1, H_6$} | 2 | 1 | 0 | ... | $2 \times \mathcal{P}(G_1|G) + 1 \times \mathcal{P}(G_2|G) + 0 \times \mathcal{P}(G_3|G) + \ldots$ |
| {$H_2$} | 1 | 1 | 1 | ... | $1 \times \mathcal{P}(G_1|G) + 1 \times \mathcal{P}(G_2|G) + 1 \times \mathcal{P}(G_3|G) + \ldots$ |
| {$H_3$} | 1 | 1 | 0 | ... | $1 \times \mathcal{P}(G_1|G) + 1 \times \mathcal{P}(G_2|G) + 0 \times \mathcal{P}(G_3|G) + \ldots$ |
| {$H_4$} | 1 | 1 | 0 | ... | $1 \times \mathcal{P}(G_1|G) + 1 \times \mathcal{P}(G_2|G) + 0 \times \mathcal{P}(G_3|G) + \ldots$ |
expected number of maximum independent occurrences of
$M$ in $G$, which is

$$
\sum_{G'' \in \mathcal{D}(\mathcal{G})} \arg\max_{H^n \subseteq \mathcal{H}(M|G'')} |H^n| \cdot \mathcal{P}(G''|G). \tag{5}
$$

Notice that in this problem, we are required to always select the largest independent embedding set in each possible deterministic network topology. We compute the expected number of independent motif by iterating over all possible deterministic networks and summing up the motif count. For example, in the example network (Fig. 2), the expected independent motif count is calculated by

$$
2 \cdot \mathcal{P}(G_1^1|G) + 1 \cdot \mathcal{P}(G_2^2|G) + 1 \cdot \mathcal{P}(G_3^3|G) + \ldots.
$$

The former definition of the independent motif counting problem above (Definition 1) seeks the genes, which are more likely to carry out the function characterized by the given motif across all possible deterministic topologies. The latter definition (Definition 2) does not care about the identity of the set of genes engaged in the process as the set of genes vary depending on the deterministic network topology observed. It instead counts the number of different ways we can observe the process separately for each topology even though that set may differ from one topology to another. In this paper, we focus on the first problem. The rationale is that we often do not know the specific deterministic topology realized at a given point in time. Furthermore, this topology can vary over time. Notice that this problem can be solved by enumerating all possible deterministic network topologies and independent embedding sets. However, it is infeasible to scale to large networks as the numbers of deterministic network topologies and independent embedding sets grow exponentially. In this paper, we develop a scalable method to tackle this problem by utilizing a polynomial model and three strategies. We discuss this polynomial model and three strategies next.

Overview of the existing solution

Here, we briefly describe the method by Sarkar et al. [24] for counting independent motif instances, as our method utilizes the same polynomial model in that study. Given a probabilistic graph $G = (V, E, \mathcal{P})$ and the specified motif pattern $M$, the algorithm works in three steps. First, it discovers all motif embeddings in the deterministic network $G'$ = $(V, E)$. It then builds an overlap graph for these embeddings. Next, it uses a heuristic strategy to count non-overlapping motif embeddings; it calculates a priority value for each node (we explain how to compute priority value below) and iteratively picks the node with the highest priority in the overlap graph. It includes the corresponding embedding to the result set, adds the probability that this embedding exists to the motif count and removes this node along with all of its neighbouring nodes from the overlap graph. It repeats this process until the graph is empty.

The key step of this method is calculating the priority value for each node in the overlap graph. The priority value of a node primarily depends on the number of neighbours of a node. In a probabilistic networks, both the existences of an embedding and its overlapping embeddings are uncertain as the edges which make up those embeddings are probabilistic. To accurately model this uncertainty, for each embedding $H_k$, it first calculates a gain value $a_k$, which equals to the probability that $H_k$ exists $(a_k = \prod_{e \in H_k} \mathcal{P}(e))$. Then it computes a loss value using the number of neighbours of $H_k$ which is represented with a random variable $B_k$. It then computes the loss value of $H_k$ as a function of $B_k$, denoted with $f(B_k)$. Finally, it determines the priority value, denoted with $\rho_k$, as a function of gain value and loss value. In this paper, we compute $\rho_k$ as $a_k/f(B_k)$.

Sarkar et al. compute the distribution of $B_k$ using a x-polynomial. To construct this x-polynomial, it first builds an undirected bipartite graph denoted with $G_k = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$. Then for each node $v_j \in \mathcal{V}_2$, it constructs an edge polynomial $Z_j$. After multiplying all edge polynomials and collapsing it, the x-polynomial takes the form

$$
Z_{H_k} = \sum_j p_{jk} t^j.
$$

The coefficients of the polynomial $Z_{H_k}$ is the true distribution of the random variable $B_k$ (i.e., \forall j, the coefficient of $t^j$ is the probability that $B_k = j$). For any further information, we refer the interested readers to [24].

Avoiding loss computation

Recall that, we calculate the distribution of $B_k$ for all nodes of the overlap graph only to select the one that yields the highest priority value $\rho_k()$ (see “Overview of the existing solution section”). Here, we develop a method to quickly compute an upper bound to $\rho_k$. This allows us to avoid computation of the distribution of $B_k$ for the node $v_i$ when the upper bound to $\rho_k$ is less than $\rho_i$ for any node $v_i$ considered prior to $v_i$. To explain this strategy, we first present our theory which establishes the foundation of the upper bound computation. We start by defining our notation.

Consider $G_k = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ of an embedding $H_k$. For a given subset $\mathcal{V}'_2 \subseteq \mathcal{V}_2$, let us denote the x-polynomial of $H_k$ after multiplying the edge polynomials of node set $\mathcal{V}'_2$ with $Z_{H_k,\mathcal{V}'_2}$. Below, we discuss our theory using a lemma, a theorem, and a corollary.

**Lemma 1** Consider the bipartite graph of motif embedding $H_k$ denoted with $G_k = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$. For all nodes $v_r \in \mathcal{V}_2 - \mathcal{V}'_2$, \forall r \in \{0, 1, 2, \ldots, |\mathcal{V}_1|\}$, we have

\[ \rho_k(v_r) \leq \rho_k(v_i) \]
\[ P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j}\right) \leq P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j \cup V_r}\right). \]

**Proof** We expand \( P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j}\right) \) as

\[ P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j}\right) = \sum_{t = \tau}^{\gamma(0)} \left( \sum_{j \in V_2^j} \sum_{i \in V_1} \left( \sum_{l \in V_2^j} \sum_{i \in V_1} x_i^l \right) \right). \]  

(7)

We first discuss how to compute the probability that exactly \( \tau \) neighboring embeddings of \( H_k \) exist. After multiplying edge polynomials and collapsing, \( Z_{H_k,V_2^j} \) takes the following form:

\[ Z_{H_k,V_2^j} = \phi_1\left( \phi_2 \left( \ldots \phi_{|V_1|} \left( \prod_{v \in V_2^j} \prod_{v \in V_2^j : v < v_2} Z_j \right) \right) \right), \]

\[ = \sum_j \delta_j \left( \sum_{l = \tau}^{\gamma(0)} \left( \sum_{x_i^l} \sum_{r \in V_1} \sum_{e_i^l} x_i^l \right) \right). \]

Here, \( \sum_{k \in V_2^j} \) which sums up all the coefficients of the polynomial terms containing \( t^l \), equals to the probability that exactly \( j \) neighboring embeddings of \( H_k \) exist after multiplying the edge polynomials of \( V_2 \). Next, we focus on one polynomial term from the above x-polynomial. Let us denote this polynomial term as \( A = \alpha t^{v_1} \prod_{v \in V_2^j} x_i^l \). Let

The polynomial term \( A \) expands into two polynomial terms denoted as \( B + C \), where \( B = \alpha t^{v_1} \prod_{v \in V_2^j} x_i^l \) and

\[ C = (1 - \alpha) t^{v_1} \prod_{v \in V_2^j} x_i^l. \]  

Two cases may happen after the collapsing of the polynomial terms \( B \) and \( C \).

**Case 1:** There is no collapse. The exponent of the variable \( t \) of polynomial terms \( B \) and \( C \) remains the same. Adding up the coefficients of term \( t^l \), we get

\[ \alpha \tau + (1 - \alpha) \tau = \alpha. \]

Thus, after multiplying another edge polynomial, the coefficient of term \( t^l \) remains the same. In other words, multiplying another edge polynomial has no effect on \( P(B_k \geq \tau) \). Mathematically,

\[ P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j}\right) = P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j \cup V_r}\right). \]

**Case 2:** There is collapse. In this case, the exponent of the variable \( t \) of polynomial term \( B \) will increase while it stays the same for polynomial term \( C \), since multiplying the second term of \( Z_r \) does not introduce any \( x \) variable. Let us denote the increment in the exponent of \( t \) (i.e., the number of \( x_i \) variables which collapse after multiplying \( Z_r \)) with \( j_0 \).

Now the polynomial terms \( B \) and \( C \) become

\[ P_{r,\alpha} t^{l + j_0} \prod_{v \in V_1} x_i^l \]  

and

\[ (1 - \alpha) t^{l + j_0} \prod_{v \in V_1} x_i^l \]  

respectively. How this multiplication affects \( P(B_k \geq \tau) \) depends on the relationship between \( j \) and \( \tau \). We have two cases:

**Case 2.a** When \( j < \tau \), polynomial term \( A \) does not contribute to \( P(B_k \geq \tau) \) before multiplying \( Z_r \). After multiplying \( Z_r \), polynomial term \( C \) also does not contribute to \( P(B_k \geq \tau) \). Whether polynomial term \( B \) contributes to \( P(B_k \geq \tau) \) depends on the relationship between \( j + j_0 \) and \( \tau \). If \( j + j_0 \geq \tau \), the probability that \( j + j_0 \) neighboring embeddings of \( H_k \) exist grows. Thus, based on the Eq. 7, \( P(B_k \geq \tau) \) increases by \( \alpha \tau (i.e., \) the coefficient of \( t^{j + j_0} \)). On the other hand, if \( j + j_0 < \tau \), polynomial term \( B \) has no effect on \( P(B_k \geq \tau) \). In conclusion, after multiplying one more edge polynomial, the value of \( P(B_k \geq \tau) \) either increases or remains the same. Mathematically,

\[ P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j} \right) \leq P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j \cup V_r}\right). \]

**Case 2b** When \( j \geq \tau \), the polynomial term \( A \) contributes to \( P(B_k \geq \tau) \). From Eq. 7, before multiplying \( Z_r \), the amount of contribution of polynomial term \( A \) to \( P(B_k \geq \tau) \) is \( \alpha \). After multiplying \( Z_r \), the amount of contribution is equal to the sum of the coefficients of the polynomial terms \( B \) and \( C \), where is \( \alpha \tau + (1 - \alpha) \tau = \alpha \). Thus, \( P(B_k \geq \tau) \) remains the same. Mathematically,

\[ P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j} \right) = P\left(B_k \geq \tau \middle| Z_{H_k,V_2^j \cup V_r}\right). \]

\[ \square \]

The above lemma leads to the following theorem:

**Theorem 1** Consider a motif embedding \( H_k \) and its corresponding bipartite graph \( G_k = (V_1, V_2, E) \). Also consider a subset \( V_2^j \subset V_2 \). Given a monotonic function \( \gamma() : \mathbb{R} \to \mathbb{R} \) such that \( \gamma(0) = 0 \) and for all \( x \), \( \gamma(x) \geq \gamma(0) \geq 0 \). For all \( v_r \in V_2 - V_2^j \), we have

\[ \sum_{j = 0}^{\gamma(0)} \gamma(j) P(B_k = j \middle| Z_{H_k,V_2}) \leq \sum_{j = 0}^{\gamma(0)} \gamma(j) P(B_k = j \middle| Z_{H_k,V_2^j \cup V_r}). \]

**Proof** From the monotonicity of \( \gamma() \) function, for all \( j \geq 1 \), we have

\[ \gamma(j) - \gamma(j - 1) \geq 0. \]

From Lemma 1, given \( V_2^j \) and \( V_r \), for all \( j \geq 0 \), we have

\[ P(B_k = j \middle| Z_{H_k,V_2}) \leq P(B_k = j \middle| Z_{H_k,V_2^j \cup V_r}). \]

For all \( j \geq 1 \), by multiplying both sides of the inequality with \( (\gamma(j) - \gamma(j - 1)) P(B_k = j \middle| Z_{H_k,V_2}) \), we get

\[ (\gamma(j) - \gamma(j - 1)) P(B_k = j \middle| Z_{H_k,V_2}) \leq (\gamma(j) - \gamma(j - 1)) P(B_k = j \middle| Z_{H_k,V_2^j \cup V_r}). \]
Thus, summing up this inequality \( \forall j \leq |V_1| \), we get
\[
\sum_{j=1}^{|V_1|} (\gamma(j) - \gamma(j-1))P(B_k \geq j|Z_{H_k,V_2'}) \leq \sum_{j=1}^{|V_1|} (\gamma(j) - \gamma(j-1))P(B_k \geq j|Z_{H_k,V_2'\cup V_r}) .
\] (8)

We rewrite the left side of this inequality as
\[
\sum_{j=1}^{|V_1|} (\gamma(j) - \gamma(j-1))P(B_k \geq j|Z_{H_k,V_2'}) = \sum_{j=1}^{|V_1|} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'}) - \sum_{j=1}^{|V_1|} \gamma(j-1)P(B_k \geq j|Z_{H_k,V_2'})
\]
\[
= \sum_{j=1}^{|V_1|} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'}) - \sum_{j=0}^{|V_1|-1} \gamma(j)P(B_k \geq j+1|Z_{H_k,V_2'})
\]
\[
= \sum_{j=0}^{|V_1|-1} \gamma(j)+\gamma(|V_1|)|P(B_k \geq j|Z_{H_k,V_2'}) - \sum_{j=0}^{|V_1|-1} \gamma(j)P(B_k \geq j+1|Z_{H_k,V_2'})
\]
\[
+ \gamma(|V_1|)P(B_k \geq |V_1||Z_{H_k,V_2'}) - \gamma(0)P(B_k \geq 1|Z_{H_k,V_2'})
\] (9)

Given \( P(B_k = j) = P(B_k \geq j) - P(B_k \geq j+1) \) and \( \gamma(0) = 0 \), we rewrite Eq. 9 as
\[
\sum_{j=1}^{|V_1|} (\gamma(j) - \gamma(j-1))P(B_k \geq j|Z_{H_k,V_2'}) = \sum_{j=1}^{|V_1|-1} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'}) + \gamma(|V_1|)P(B_k = |V_1||Z_{H_k,V_2'})
\]
\[
= \sum_{j=1}^{|V_1|} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'})
\]

Similarly, we rewrite the right side of Inequality (8) as
\[
\sum_{j=1}^{|V_1|} (\gamma(j) - \gamma(j-1))P(B_k \geq j|Z_{H_k,V_2'\cup V_r}) = \sum_{j=1}^{|V_1|} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'\cup V_r}) .
\]

Using the above equations, We rewrite the Inequality (8) as
\[
\sum_{j=1}^{|V_1|} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'}) \leq \sum_{j=1}^{|V_1|} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'\cup V_r}) .
\]

As \( \gamma(0) = 0 \), using the above inequality, we get
\[
\sum_{j=0}^{|V_1|} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'}) \leq \sum_{j=0}^{|V_1|} \gamma(j)P(B_k \geq j|Z_{H_k,V_2'\cup V_r}) .
\]

This theorem gives us a general form of \( f(B_k) \) function which is monotonically increasing. For example, the expected value of \( B_k, \text{Exp}(B_k) \) falls into that category. Corollary below proves it:

**Corollary 1** Given \( V_2' \) and \( v_r \in V_2 - V_2' \), the expected number of neighboring embeddings of \( H_k \) monotonically increases with growing edge polynomial set:
\[
\text{Exp}(B_k|Z_{H_k,V_2'}) \leq \text{Exp}(B_k|Z_{H_k,V_2'\cup V_r}) .
\]

**Proof** The expected value of \( B_k \) can be computed as
\[
\text{Exp}(B_k) = \sum_{j=0}^{|V_1|} jP(B_k = j).
\]

We have \( \gamma(j) = j \) which is a monotonical function. Thus, from Theorem 1, we have
\[
\text{Exp}(B_k|Z_{H_k,V_2'}) \leq \text{Exp}(B_k|Z_{H_k,V_2'\cup V_r}) .
\]

Using Theorem 1, we develop our method for avoiding the costly computation of the distribution of \( B_k \) for each embedding \( H_k \) of the given motif in the target network. Our method works for all monotonic loss functions (e.g., \( f(B_k) = \text{Exp}(B_k) \)). Assume that, \( \exists \gamma > 1, \forall 1 \leq i < k, \) we already computed the values \( a_i, \text{distribution of } B_i, f(B_i), \) and thus \( \rho_i \). Let us denote the largest observed priority value so far with \( \rho^* = \max_{1 \leq i < k} \{\rho_i\} \). We explain next how we use this information to avoid computation of the distribution of \( B_k \) whenever possible. Let us denote the bipartite graph of \( H_k \) with \( G_k = (V_1,V_2,E) \). Our algorithm iteratively multiplies the edge polynomials for all the nodes in \( V_2' \) one by one and collapses the resulting polynomial. Let us denote the set of nodes in \( V_2' \) with \( V_2' = v_1,v_2,\ldots,v_{|V_2|} \). Without losing generality, let us assume that we multiply the edge polynomials in the order \( v_1,v_2,\ldots,v_{|V_2'|}, \forall j, 1 \leq j \leq |V_2'| \), after multiplying the first \( j \) polynomials, we get an intermediate probability distribution \( B'_k \). Using that distribution \( B'_k \), we compute an intermediate priority value for \( H_k \) and denote it with \( \rho'_k \). Recall that Theorem 1 states that \( \forall j, \rho'_k \leq \rho'^{-1}_k \) (i.e., the priority value monotonically decreases if loss value monotonically increases). Following from this theorem, we terminate computation of \( B_k \) as soon as \( \rho'_k \) becomes less than the best priority value observed, \( \rho^* \). This eliminates costly polynomial multiplication for \( H_k \).

When to stop the calculation of \( B_k \) largely depends on the best priority value \( \rho^* \) observed so far. The larger \( \rho^* \) is, the sooner we terminate the computation of \( B_k \). Thus, the ideal ordering places embeddings with larger priority values should earlier. The dilemma here is that we do not
know the priority values of the embeddings at this stage. Therefore, we use a proxy value of each embedding \( H_k \), denoted with \( Q_k \), which is trivial to compute, \( a_k \) (i.e., the gain value of \( H_k \)) divided by the number of overlapping embeddings with \( H_k \). We rank embeddings in descending order of their \( Q_k \) values. The rationale behind using the value \( Q_k \) is as follows. Recall that the priority value of \( H_k \) is determined by the gain value \( a_k \) and the loss value, which largely depends on the distribution of its neighboring nodes. \( Q_k \) conjectures that the larger the degree of the corresponding node of \( H_k \) is, the larger its loss value is. Thus, \( Q_k \) is inversely proportional to the number of embeddings, which conflict with \( H_k \).

Efficient polynomial collapsation

Collapsation plays an important role in calculating the distribution of \( B_k \) of the embedding \( H_k \) efficiently. The sooner we collapse the polynomial terms, the earlier we compute an upper bound to \( \rho_k \). Here, we introduce two orthogonal strategies to ensure early collapsation during the construction of the x-polynomial. We describe our strategies on the bipartite graph \( G_k = (V_1, V_2, E) \) of \( H_k \). Our first strategy focuses on \( V_1 \). The second one focuses on \( V_2 \).

Optimization on \( V_1 \)

In order to collapse an x-polynomial term, which contains variable \( x_i \), we need to multiply all the edge polynomials containing \( x_i \) (see Eq. 3). The degree of the node \( v_i \in V_1 \), \( \text{deg}(v_i|G_k) \) is equal to the number of edge polynomials that the variable \( x_i \) needs to be collapsed. Consider a node \( v_i \in V_1 \) with \( \text{deg}(v_i|G_k) = 1 \). Suppose that \( \exists v_j \in V_2, (v_i, v_j) \in E \). We collapse the variable \( x_i \) as soon as the edge polynomial \( Z_j \) has been multiplied into the x-polynomial. In this case, the collapse operator \( \phi \) will replace the variable \( x_i \) in x-polynomial with \( t \). Following from this observation, our first strategy works as follows. Consider a node \( v_j \in V_2 \). Let us denote the set of all nodes \( v_i \in V_1 \) for which \( \text{deg}(v_i|G_k) = 1 \) and \( (v_i, v_j) \in E \) with \( V_{1,j} \). We rewrite Eq. 1 (see Section “Preliminaries and problem definition”) as

\[
Z_j = p t^{|V_{1,j}|} \prod_{v_j \in V_{1,j} \setminus V_{1,j}^{-1}} x_i + q_j.
\]

The above equation means that before we do any polynomial multiplication, we first apply the collapse operator \( \phi_j \) (\( \forall i, \) such that \( v_i \in V_{1,j} \)) to \( Z_j \). This preemptive collapsation prevents the exponential growth in the number of collapsation operations for those \( v_i \) satisfying the conditions above. For example, in the example bipartite graph (see Fig. 4), for edge \( e_8 \), its original edge polynomial \( Z_8 = p_8 x_6 + q_8 \) can be rewritten as \( Z_8 = p_8 t + q_8 \) to avoid applying the collapse operation \( \phi \) in any further polynomial multiplication.

Optimization on \( V_2 \)

The order in which edge polynomials are multiplied has a great effect on the cost of polynomial multiplication. Recall from “Preliminaries and problem definition section” that each variable \( x_i \) collapses only after the multiplication of the final edge polynomial for \( x_i \) has been completed. Following from this observation, our second strategy conjectures that increasing the number of collapsing variables after the product of a given number of edge polynomials reduces both the running time and the amount of memory needed to store the x-polynomial. We explain this on the bipartite graph in Fig. 4. In our example, to simplify our notation, we will only consider the optimization on \( V_2 \) and ignore the impact of the optimization on \( V_1 \) described above. We have four edge polynomials in total. If we multiply four edge polynomials in the order of \( Z_1, Z_2, Z_4, Z_8 \), no collapsation takes place until we multiply \( Z_4 \). This is because \( Z_4 \) is the final edge polynomial for \( x_1, x_2, x_3 \) and \( x_5 \). As a result, this ordering requires in total 32 collapses (i.e., number of polynomial terms is \( 2^3 = 8 \), and we collapse each of \( x_1, x_2, x_3 \) and \( x_5 \) resulting in \( 8 + 8 + 8 + 8 \) operations). Adding the collapsation cost of the last edge polynomial, \( 2^4 = 16 \), we achieve 32 + 16 = 48 operations in total. Now, let us analyse the cost of the same product when we multiply the edge polynomials in the order of \( (Z_4, Z_2, Z_8, Z_1) \). In this ordering, \( Z_4 \) is the final edge polynomial for \( x_5 \). Thus, we need another \( 2^1 = 2 \) operations for variable \( x_5 \). The following edge polynomial \( Z_2 \) is the final edge polynomial for variables \( x_2 \) and \( x_3 \). Therefore, once we multiply \( Z_2 \), we can collapse variables \( x_2 \) and \( x_3 \). Thus, for variables \( x_2 \) and \( x_3 \), only 8 collapses are needed (i.e., there are 4 polynomial terms and 2 collapse operators). Variables \( x_4 \) and \( x_1 \) collapses after the product of \( Z_8 \) and \( Z_1 \) leading to eight and sixteen more collapse operations respectively. In total, this ordering yields only 34 (i.e., \( 2 + 8 + 8 + 16 \)) collapses. By reordering the edge polynomials, we reduce not only the time for collapsation, but also the memory space for storing the variables. Furthermore, as we explain below, an effective ordering has potential to avoid the loss computation without losing the accuracy of the result. Next, we formally define the problem of ordering of the edge polynomials.

Definition 3 ORDERING OF THE EDGE POLYNOMIALS.

Assume a bipartite graph \( G_k = (V_1, V_2, E) \) and a specified loss value denoted by \( \epsilon \) are given. Each node \( v_i \in V_1 \) has a unique variable \( x_i \) and a collapse operator \( \phi_i \). Each node \( v_j \in V_2 \) has an edge polynomial \( Z_j \). For each collapse operator \( \phi \), let us denote the number of the polynomial terms it has been applied to with \( N_\phi \). Let us denote a permutation of the integers in the \( \{1 : |V_2|\} \) interval with \( \pi = [\pi_1, \pi_2, \ldots, \pi_{|V_2|}] \). Our problem is to find the ordering \( \pi \), for which \( \exists r \), such that after multiplying the first \( r \) polynomials in the
order of $\pi$, we have $f(B_\pi) \geq \epsilon$ and $r|V_1|2^r + \sum_{i=1}^{|V_1|} N_{\phi_i}$ is minimized.

Notice that in the definition above, we aim to minimize the number of edge polynomials (i.e., variable $r$). Also if there are multiple orders with the same number of edge polynomials, we prefer the one that requires the least collapse operations (i.e., $\sum_{i=1}^{|V_1|} N_{\phi_i}$).

One straightforward method to solve this problem is to calculate the collapse cost for all possible orderings of the edge polynomials and choose the one with the smallest cost. This, however, is infeasible as there are $|V_2|!$ alternative orderings. Here, we develop a greedy iterative algorithm to quickly estimate an ordering. Briefly, at each iteration, our algorithm chooses the edge which contributes to the collapse of most variables. We explain our algorithm using the bipartite graph shown in Fig. 4.

Our algorithm maintains two matrices. We denote these two matrices at the $i$th iteration with $W_i$ and $D_i$. At each stage, we update the $W_i$ matrix and generate a $D_i$ matrix based on $W_i$. We will explain these two matrices in detail later. We choose a suitable edge polynomial using the $D_i$ matrix, and repeat this process until the last edge polynomial.

We first explain the two matrices above. The first matrix denoted by $W_i$ maintains the relationship between the $x$ variables and edge polynomials. Let us denote the $r$th row and $s$th column of $W_i$ with $W_i[r,s]$. Also, let us denote the bipartite graph of $H_k$ at the $i$th iteration with $G^i_k$. If the $x$-polynomial $Z_\nu$ contains the variable $x_r$, we set $W_i[r,s] = 1/\deg (\nu, G^i_k)$. Otherwise, we set $W_i[r,s] = 0$. Conceptually, this number indicates the contribution of the edge polynomial $Z_\nu$ to collapse variable $x_r$. For instance, $W_i[r,s] = 1$ implies that $Z_\nu$ is the final edge polynomial of $x_r$. Figure 5 presents matrix $W_1$ corresponding to the bipartite graph in Fig. 4. Here, $Z_2$ and $Z_4$ contain $x_2$. As a result $W_1[2,2] = W_1[2,3] = 1/2$.

We construct matrix $D_i$ from $W_i$. This matrix counts different levels of contributions of each edge polynomial. It has $|V_2|$ rows and $\max_{u \in V_1} \{\deg(u|G^i_k)\}$ columns. We set the entry $D_i[r,s]$ to the product of the number of entries in the $r$th column of $W_i$ having the value $1/s$ and the edge probability in $Z_r$. For example, in the matrix $W_1$ in Fig. 5, $Z_2$ (i.e., column 2) contributes two variables at value $1/2$ and one variable $1/3$. As a result, we set the second row of $D_1$ to $[0, 2p_2, p_2]$.

At the $i$th iteration, using the matrix $D_i$, we choose the next edge polynomial for multiplication as follows. We start by looking at the first column of $D_i$, and choose the row (i.e., edge polynomial) with the largest value. If there are multiple such rows, we use the second column of $D_i$ among those polynomials. We repeat this process until we find such a row with the largest value. If there are still more than one rows after reaching the last column of $D_i$, we randomly choose the edge polynomial corresponding to one of them. For example, in Fig. 5, both $Z_4$ and $Z_5$ has values in the first column. We choose one of them depending on their edge probability. If they have the same edge probability (i.e., $p_4 = p_5$), we look at their values in the second column, where $Z_4$ should be selected.

At the $i$th iteration, assume that we pick the edge polynomial $Z_r$. We update $G^i_k$ for the next iteration by removing the node $\nu$ from $G^i_k$ along with all of its incident edges.

**Overcoming memory bottleneck**

The number of terms of the $x$-polynomial can grow exponentially, particularly when collapse does not take place. This quickly leads to memory bottleneck especially for dense overlap graphs. In this section, we present a recursive strategy to overcome this bottleneck.

The main idea behind this strategy is as follows. Given a new edge polynomial, we multiply it with only a subset of the current $x$-polynomial terms, while deferring others. After completing the multiplication of all edge polynomials, we multiply the deferred polynomial terms using last-in, first-out policy. Figure 6 depicts this idea. Here, the shaded bar represents the subset of terms of the current $x$-polynomial to be multiplied with the next edge polynomial (e.g., $Z_{i+1}$). Let us denote the number of terms in this subset with $N_1$ and the number of deferred terms

| $W_1$ | $Z_1$ | $Z_2$ | $Z_4$ | $Z_8$ |
|-------|-------|-------|-------|-------|
| $x_1$ | $1/3$ | $1/3$ | $1/3$ | $0$   |
| $x_2$ | $0$   | $1/2$ | $1/2$ | $0$   |
| $x_3$ | $0$   | $1/2$ | $1/2$ | $0$   |
| $x_5$ | $0$   | $0$   | $1$   | $0$   |
| $x_6$ | $0$   | $0$   | $0$   | $1$   |

| $D_1$ | $1$ | $1/2$ | $1/3$ |
|-------|-----|-------|-------|
| $Z_1$ | $0$ | $0$   | $p_1$ |
| $Z_2$ | $0$ | $2p_2$| $p_2$ |
| $Z_4$ | $p_4$| $2p_4$| $p_4$ |
| $Z_8$ | $p_8$| $0$   | $0$   |

**Fig. 5** The two matrices $W_1$ and $D_1$ we maintained to order the edge polynomials.
with $N_2$ prior to multiplying with $Z_{i+1}$. After multiplication with $Z_{i+1}$, the number of terms become $2N_1 + N_2$. Repeating this process, after multiplying $j$ edge polynomials, we have only $(j+1)N_1 + N_2$ terms. Notice that this is a dramatic improvement over the original algorithm which creates $2(N_1 + N_2)$ terms. The number of deferred terms is governed by the amount of available memory. That is we choose $N_1$ as large as possible while the maximum term count $(j+1)N_1 + N_2$ remains less than available memory.

Results and discussion

In this section, we experimentally evaluate the performance of ProMotE on synthetically generated (“Evaluation on synthetic networks section”) and real networks (“Evaluation on cancer networks” to “Evaluation on neurodegenerative disease networks section”). We run our experiments on a Linux server which has AMD Opteron 24 core processors (up to 1.4GHZ) and 32GB memory.

Evaluation on synthetic networks

Running time evaluation

The key computational challenge we aim to tackle in this paper is to scale to large network sizes while existing methods fail. Here, we evaluate how well we achieved this goal. We compare the running time of ProMotE to that of Sarkar et al. [24] as this is the most recent and most efficient algorithm in the literature to the best of our knowledge. As these two methods have the similar expected motif count, we do not report their motif count. We test both methods on synthetic networks. We generate random networks with growing number of nodes using three random network models, namely Erdős Rényi (ER) [25], Watts Strogatz (WS) [26] and Barabási-Albert (BA) [27] models. We synthetically assign a probability value generated from the uniform distribution in the interval (0,1) to each edge of networks. We set the average node degree to two. For each number of nodes, we repeat the experiments for 20 random networks. We measure the total running time for four motif patterns (see Fig. 7). These motifs are also used in the study by Sarkar et al. [24]. To avoid the potential outlier, for the 20 networks of each network size, we exclude the two networks with largest running time, and another two with the smallest running time. For the remaining networks, we report the average, minimum and maximum values. For each method, we report the running time on networks if the algorithm completes in less than 10,000 seconds. Figure 8 presents the results.

We observe that our method is several orders of magnitude faster than Sarkar et al. for all parameter settings. As the network size increases, the gap between the running times of two methods grows. This indicates that our optimization strategies are greatly helpful on large networks. The advantage of ProMotE stands out the most for the networks generated using ER and BA models. For instance, on networks generated using ER model (see Fig. 8a), ProMotE successfully scales to massive network sizes (i.e., thousands of nodes). On the other hand, Sarkar et al. fails to run beyond 100 nodes. This implies that ProMotE is not only practical but it is also essential to study real networks, as the size of real networks is often large. Furthermore, we observe that the topological
characteristics of the network greatly affect the cost of counting independent motifs. For the same network size, we observe a dramatic increase in running time as we move from WS model to ER model and then to BA model. ER model generates networks with a small average shortest path length along with a small clustering coefficient. WS model however builds networks with both short average paths and strong clustering coefficients. Thus, networks generated by ER model are more likely to have isolated nodes, which in turn make the remaining connected components much denser than those in networks generated by WS model. Dense networks however result in more computation. As a result, ProMotE runs much faster on networks of WS model. For the BA model, it constructs the so called scale-free networks characterized by a highly heterogeneous degree distribution, which follows power law. Thus, there exist more hub nodes in networks of BA model, which result in more overlapping motif embeddings. As a result, the overlap graphs of networks of BA model are much more complicated, which in turn introduce a large quantity of computation. Recall that the BA model generates scale free networks. Thus it is expected to generate networks that resemble real data better than the other two models. This indicates that counting independent motifs in real networks is a challenging problem, and ProMotE is needed to study them in practical time.

Comparison against the literature
In the literature, current approaches to the probabilistic networks often transform probabilistic networks to deterministic networks first, and then apply methods for deterministic networks. These approaches include ignoring probability values [28, 29], considering edges with probability values above a given threshold [30], and sampling the probabilistic network by doing a Bernoulli trial with probability $p_i$ for each edge $e_i$ [31]. For simplicity, we denote these three approaches with binary, threshold and sampling method respectively. For each of these method, after transforming probabilistic networks to deterministic networks, we apply the method in the literature [19] to find the motif count, which heuristically selects motifs with the least number of overlapping embeddings. As these methods are not specifically devised to solve the problem in this paper and they finally work on deterministic networks, the performance of ProMotE in terms of running time is significantly worse than these methods. As a result, we compare our method against these methods only in terms of expected motif count.

Recall that the threshold method maintains the set of edges with probability value above a given threshold value and removes the remaining edges. Thus, the outcome of this method depends on the threshold value. Here, we first evaluate the performance of the threshold method for varying values of threshold. We run our experiment on synthetic networks with WS model of various network sizes, 1000, 5000 and 10000. In particular, for each network size, we repeat the experiments for 20 random networks. We vary the threshold value from 0 to 0.7 at increments of 0.1. For each threshold value setting, we run experiment on all generated networks and report the average motif count. Figure 9 plots the results.

We observe that the motif count of the threshold method first grows with the increasing threshold value. It then falls sharply. It obtains the peak value when the threshold is 0.2. This is possibly because too small/large threshold leads to the retaining/removing most of the edges of the network. Either case may make the threshold method underutilize the information available in the interaction probabilities. As a result, a suitable threshold value is necessary for the threshold method. However, the varying distribution of edge probabilities of different probabilistic networks makes it difficult to set a fixed threshold value for the threshold method. In the rest of our experiments, we fix the threshold value of the threshold method to 0.2 as it obtains the best value on the average across a broad spectrum of parameter settings.

Next, we compare our method with binary, threshold and sampling methods. We test these methods on synthetic networks with different network models of various
network sizes. To ensure the reliability of our results, for each parameter, we conduct experiments on 20 different networks and report the average. Specifically, for sampling method, as each running on the same network may have the different value, to get a reliable result, we run 10 times on each network and report the average. Figure 10 reports the result. Our results demonstrate that our method has the highest motif count for all network sizes with various network models. Moreover, the gap between ProMotE and other methods increases with the growing number of nodes. This is very promising since we expect to have more number of nodes in real networks. For instance, on the network with WS network model of size 10,000, ProMotE has 13.2 to 95.4% more motif counts than other methods. The threshold method achieves the second best motif count. That said, it is worth noting that we give a positive bias towards the threshold method since we fix the threshold to the value which maximized its motif count in our previous experiments. Sampling method has overall worst performance across different network sizes. The reason behind is that finding a set of independent embeddings that yields the maximum expected number of occurrences in a probabilistic network is a nonlinear function. Although, the sampling approach can provide provable confidence intervals for estimating linear functions such as sum and average, it fails to do that for nonlinear functions such as counting independent motifs in probabilistic networks. Due to the nonlinear nature of our problem, a sampling approach is expected to produce inaccurate results. Furthermore, we observe that the expected motif count of all methods grows with the increasing network size. This is because the network with larger size is expected to have more motifs.

In summary, our method is very efficient and accurate on counting independent motif in probabilistic networks.

Evaluation on cancer networks
In order to observe the performance of our method on real networks, we apply our algorithm on six cancer datasets from the MSigDB database [32] for Homo Sapiens. We extract genes from the C2:curated gene sets of MSigDB. We then feed each cancer gene set into the STRING database [16] to generate its interaction network. We use the gene co-expression values present in the STRING dataset for these networks to compute their interaction
Table 2: Real networks from cancer dataset used in our experiments; number of nodes and edges, average node degree and clustering coefficient

| Cancer  | Nodes | Edges | Avg. degree | C. coefficient |
|---------|-------|-------|-------------|----------------|
| Thyroid | 39    | 43    | 2.21        | 0.773          |
| Bladder | 49    | 51    | 2.08        | 0.572          |
| Endometrial | 54 | 61    | 2.26        | 0.676          |
| Lung    | 64    | 66    | 2.06        | 0.732          |
| Colorectal | 72  | 83    | 2.31        | 0.798          |
| Pancreatic | 80 | 87    | 2.17        | 0.791          |

probabilities. Specifically, for a pair of nodes, the interaction probability between them is computed as the Pearson’s correlation coefficient. In the literature, there are also other ways to compute interaction probabilities. For example, Sharan et al. [33] addressed this problem by utilizing features like the volume of evidence present for the interaction, gene expression correlation, and network topology to learn the edge probabilities. Gabr et al. [34] used end-to-end signal reachability probabilities between pairs of genes to guide the computation of the edge probabilities. All these studies use transcriptional values in their computations in slightly different ways. Table 2 presents the dataset details for each cancer type. We measure independent motif counts for the four basic patterns listed in Fig. 7.

Figure 11a presents the running time of ProMotE. In addition, we also plot the average degree of each network to display the effect of running time on network size. Our results demonstrate that ProMotE successfully identifies independent motifs in practical time (1.5 secs to less than 2.2 h) for all networks. It is worth noting that Sarkar et al.’s method does not scale to these network sizes.

Figure 11b shows the number of non-overlapping instances of the four basic patterns present in each of the cancer networks. We observe that the pattern M1 is the most abundant topology in all networks. This is expected as M1 is a subgraph of M2, M3, and M4. The other three motifs have similar counts. We also observe that the motif count does not necessarily grow with the network size. For instance Thyroid cancer has the least number of nodes and edges, yet it contains more non-overlapping motif instances than almost all the other networks in our dataset. This implies that the topology of the network governs the motif distribution. Recall that the running time of ProMotE tends to grow with network size on the cancer networks (see Fig. 11a). This implies that the running time of ProMotE does not strongly depend on the independent motif count as well.

Evaluation on neurodegenerative disease networks

Next, we evaluate ProMotE on three neurodegenerative disease networks; Alzheimer’s, Huntington’s and Parkinson’s disease, hereafter referred to as AD, HD and PD respectively. We obtain this dataset from the MSigDB database similar to the cancer networks (see “Evaluation on cancer networks section”). Table 3 lists the dataset details. One major difference between this and the cancer dataset is that the neurodegenerative networks are both larger in size and average degree. It is worth noting that the AD network is a subgraph of that of HD network. Also, all three networks share substantial amount of edges (details later in this section). We focus on motif M2 (the loop pattern) in this experiment.

Table 4 shows the results. We observed that AD and HD networks yield the same number of embeddings and $F_2$ count. Totally, 37% to 56% of the genes participate in at least one motif embedding. PD network has the largest motif count and fraction of genes in motif embeddings. Such large motif count indicates that these networks are organized largely as a combination of small loops. Finally, our method completed counting motifs in slightly over half an hour per network. This demonstrates that our method scales up to very large network sizes and densities.
Table 3 Real networks from neurodegenerative dataset used in our experiment, the disease name, numbers of nodes and edges, average node degree and clustering coefficient

| Disease | Nodes | Edges | Avg. degree | C. coefficient |
|---------|-------|-------|-------------|----------------|
| AD      | 162   | 737   | 9.1         | 0.876          |
| HD      | 177   | 756   | 8.54        | 0.845          |
| PD      | 125   | 751   | 12          | 0.827          |

AD, HD and PD correspond to Alzheimer’s, Huntington’s and Parkinson’s disease, respectively.

Next, we take a closer look at the distribution of the shared edges, which appear in a motif embedding, across different diseases. Figure 12 presents the distribution. We observe that the edges present in AD and HD are exactly the same. PD network however deviates from the other two by about 6%. This implies that all three diseases possibly are governed by very similar processes, with PD having slight variations.

Next, we focus on the statistical significance of the biological processes and molecular functions of the genes that comprise the result set for the three neurodegenerative diseases for M2 pattern. To do this, we perform gene ontology analysis using PANTHER [35]. For each network, we use all the genes appearing in an embedding of M2. We filter the Gene Ontology (GO) terms with a p-value above 0.025. We then search each of the remaining GO term in PubMed with the disease corresponding to that network (i.e., AD, HD, or PD). Table 5 shows the results for which at least one reference publication exists.

Our results demonstrate that ProMotE identifies disease specific functional properties. For instance, mitochondria often referred to as the power house of the cell is responsible for many important cellular functions especially neuronal viability. Therefore, aberrations in mitochondrial processes have potential to lead to neuronal disorder. Four distinct pathways carry out mental processing for brain metabolism, two of them being tricarboxylic acid cycle (TCA) and electron transport chain (the other two are glycolysis, pentose shunt). These pathways are affected by direct modification of the enzymes or alterations at the gene expression level. TCA is responsible for producing reducing equivalents in the form of reduced nicotinamide adenine dinucleotide (NADH) and reduced flavin adenine dinucleotide (FADH2). Altered activities of the TCA cycle enzymes result in imbalance which is associated with AD-related changes in metabolism.

Another important function is the production of adenosine triphosphate (ATP) through the combined effects of TCA and the respiratory chain, also known as electron transport chain. The respiratory chain requires two electron carriers: ubiquinone/coenzyme Q and cytochrome c. Also, it consists of five protein complexes: NADH dehydrogenase-ubiquinone oxidoreductase (complex I), succinate dehydrogenase-ubiquinone oxidoreductase (complex II), ubiquinone-cytochrome c oxidoreductase (complex III), cytochrome c oxidase (complex IV), and ATP synthase (complex V). Production of ATP involves two coordinated mechanisms: electrons received from energy substrates such as NADH are transported through the mitochondrial complexes towards molecular oxygen, producing water; at the same time electrochemical gradient is generated by driving protons across the mitochondrial inner membrane by I, III, and IV protein complexes. Finally, ATP is produced by the accumulation of these protons into the matrix with the help of complex V (ATP synthase). Altered mitochondrial respiration, especially at the level of complex I thus associates with PD.

Conclusions
In this paper, we developed ProMotE, an efficient method to count non-overlapping motif instances in probabilistic networks. This method uses a polynomial model to capture the dependencies between overlapping embeddings. We proposed three strategies to avoid computation of loss value, to expedite collapsation of polynomial terms, and to overcome the memory bottleneck faced when applied to large networks. Our experiments on both synthetic and
Table 5  Gene ontology analysis of the genes appearing in the independent embeddings of the triangle pattern in the three disease networks with publications, and the diseases associated with the ontology terms

| Category          | GO ID   | Function name                          | p-value   | Reference          | Disease |
|-------------------|---------|----------------------------------------|-----------|--------------------|---------|
| Biological        | 0006120 | mitochondrial electron transport, NADH to ubiquinone | 2.65E-58  | Perier and Vila [36] | PD      |
|                   | 0042776 | mitochondrial ATP synthesis coupled proton transport | 1.43E-20  | VanDuyne et al. [37] | AD, PD  |
| Process           | 0006122 | mitochondrial electron transport, ubiquinol to cytochrome c | 4.66E-20  | Fiskum et al. [38] | PD      |
|                   | 0006099 | tricarboxylic acid cycle                | 5.13E-19  | Kim et al. [39]    | AD      |
|                   | 0003954 | NADH dehydrogenase activity             | 2.43E-02  | Shi et al. [40]    | AD      |
|                   | 0004129 | cytochrome-c oxidase activity           | 9.19E-53  | Zubenko et al. [41] | AD      |
| Molecular         | 0008121 | ubiquinol-cytochrome-c reductase activity | 1.4E-18   | Cardoso et al. [42] | AD      |
| Function          | 0000104 | succinate dehydrogenase activity        | 9.53E-14  | Liang et al. [43]  | AD      |
|                   | 0048038 | quinone binding                         | 6.42E-06  | Fattoretti et al. [44] | AD, HD |
|                   | 0051537 | 2 iron, 2 sulfur cluster binding        | 8.33E-04  | Wang et al. [45]   | AD, PD  |

real networks demonstrate that our method scales to large networks and identifies the key functional characteristics of cancer and disease phenotypes.

Abbreviations
AD: Alzheimer’s disease; ATP: Adenosine triphosphate; BA: Barabási-Albert; ER: Erdős Rényi; FADH2: Reduced flavin adenine dinucleotide; HD: Huntington’s disease; NADH: Reduced nicotinamide adenine dinucleotide; ProMoE: Probabilistic motif embedding; PD: Parkinson’s disease; TCA: Tricarboxylic acid cycle; WS: Watts Strogatz

Acknowledgements
We would like to thank the reviewers for their insightful comments and suggestions.

Funding
Publication of this article was funded by NSF under grant DBI-1262451. Neither funding body played any role in the design of this study and collection, analysis, and interpretation of data or in writing the manuscript.

Availability of data and materials
The datasets generated and/or analysed during this study along with the code for our models and instructions for their use are available under open licenses at https://github.com/KahveciLab/probabilisticIndependentMotif.

Authors’ contributions
Developed the method: YR, AS and TK. Conceived and Designed the experiments: YR, AS and TK. Performed the data analysis: YR, AS and TK. Performed the experiments and interpreted the results: YR, AS and TK. Contributed to the writing of the manuscript: YR, AS and TK. All authors read, provided comment and approved the final manuscript.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Publisher’s Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 27 November 2017 Accepted: 6 June 2018
Published online: 26 June 2018

References
1. Bray D, et al. Protein molecules as computational elements in living cells. Nature 1995;376(6653): 307–12.
2. Fajjole M, Rotondo G, Daviet L, Bergametti F, Inchauspé G, Titoias P, Transy C, Legrain P. A genomic approach of the hepatitis c virus generates a protein interaction map. Gene. 2000;242(1):369–79.
3. Uetz P, Giot L, Cagney G, Mansfield TA, Judson RS, Knight JR, Lockshon D, Narayan V, Srinivasan M, Pochart P. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. Nature. 2000;403(6770):623−7.
4. Girvan M, Newman ME. Community structure in social and biological networks. Proc Natl Acad Sci. 2002;99(12):7821−6.
5. Albert R, Jeong H, Barabási A-L. Error and attack tolerance of complex networks. Nature. 2000;406(6794):378–82.
6. Green ML, Karp PD. A bayesian method for identifying missing enzymes in predicted metabolic pathway databases. BMC Bioinformatics. 2004;5(1):1.
7. Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, Alon U. Network motifs: simple building blocks of complex networks. Science. 2002;298(5594):824−7.
8. Wang P, Lu J, Yu X. Identification of important nodes in directed biological networks: A network motif approach. PLoS ONE. 2014;9(8):106132.
9. Hu Y, Flockhart I, Vinayagam A, Bergwitz C, Berger B, Perrimon N, Moehr SE. An integrative approach to ortholog prediction for disease-focused and other functional studies. BMC Bioinformatics. 2011;12(1):1.
10. Shen-Orr SS, Milo R, Mangan S, Alon U. Network motifs in the transcriptional regulation network of escherichia coli. Nat Genet. 2002;31(1):64−68.
11. Karey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: WH Freeman; 1979.
12. Bader JS, Chaudhuri A, Rothberg JM, Chant J. Gaining confidence in high-throughput protein interaction networks. Nat Biotechnol. 2002;20(1):78−85.
13. Ryba T, Hiratani I, Sasaki T, Battaglia D, Kulik M, Zhang J, Dalton S, Gilbert DM. Replication timing: a fingerprint for cell identity and pluripotency. PLoS Comput Biol. 2011;7(10):1002225.
14. Schübler D, Scalzo D, Kooperberg C, van Steensel B, Delrow J, Groudine M. Genome-wide dna replication profile for drosophila melanogaster: a link between transcription and replication timing. Nat Genet. 2002;32(3):438−42.
15. Colclough D, Franceschini A, Kuhn M, Simonovic M, Roth A, Mingez P, Doerk S, Stark M, Muller J, Bork P. The STRING database in 2011:
functional interaction networks of proteins, globally integrated and scored. Nucleic Acids Res. 2010;39(suppl_1):D561–D568.

17. Inokuchi A, Washio T, Motoda H. Complete mining of frequent patterns from graphs: Mining graph data. Mach Learn. 2003;50(3):321–54.

18. Kuramochi M, Karypis G. Frequent subgraph discovery. In: Data Mining, 2001. ICDM 2001. Proceedings IEEE International Conference On. New Jersey: IEEE; 2001. p. 313–320.

19. Schreiber F, Schwöbbermeyer H. Frequency concepts and pattern detection for the analysis of motifs in networks. In: Transactions on Computational Systems Biology. Heidelberg: Springer; 2005. p. 89–104.

20. Tran NH, Choi KP, Zhang L. Counting motifs in the human interactome. Nat Commun. 2013;4:2241.

21. Todor A, Dobra A, Kahveci T. Counting motifs in probabilistic biological networks. In: ACM Conference on Bioinformatics, Computational Biology and Health Informatics. New York: ACM; 2015. p. 116–125.

22. Kuramochi M, Karypis G. Finding frequent patterns in a large sparse graph. Data Min Knowl Disc. 2005;11(3):243–71.

23. Klukas C, Koschützki D, Schreiber F. Graph pattern analysis with patterngravisto. J Graph Algorithm Appl. 2005;9(1):19–29.

24. Sarkar A, Ren Y, Elhesha R, Kahveci T. Counting independent motifs in probabilistic networks. In: ACM Conference on Bioinformatics, Computational Biology and Health Informatics. New York: ACM; 2016. p. 231–240.

25. Erdös P, Rényi A. On random graphs. Publ Math Debr. 1959;6:290–7.

26. Watts DJ, Strogatz SH. Collective dynamics of ‘small-world’ networks. Nature. 1998;393(6684):440–2.

27. Barabási A-L, Albert R. Emergence of scaling in random networks. Science. 1999;286(5439):509–12.

28. Marcotte EM, Pellegrini M, Thompson MJ, Yeates TO, Eisenberg D. A combined algorithm for genome-wide prediction of protein function. Nature. 1999;402(6757):83–86.

29. Schwikowski B, Uetz P, Fields S. A network of protein–protein interactions in yeast. Nat Biotechnol. 2000;18(12):1257–61.

30. Poitet T, Citwill AR, Cazelles K, Gravel D, Fortin M-J, Strouf DB. The structure of probabilistic networks. Methods Ecol Evol. 2015;7(3):303–12.

31. Huang H, Zhang LV, Roth FP, Bader JS. Probabilistic paths for protein complex inference. In: Systems Biology and Computational Proteomics. Heidelberg: Springer; 2007. p. 14–28.

32. Liberzon A, Subramanian A, Pinchback R, Thorvaldsdóttir H, Tamayo P, Mesirov JP. Molecular signatures database (msigdb) 3.0. Bioinformatics. 2011;27(12):1739–40.

33. Van Duyn N, Settivari R, LeVora J, Zhou S, Unrine J, Nass R. The metal transporters smf-3/dmt-1 mediates aluminum-induced dopamine neuron degeneration. J Neurochem. 2013;124(1):147–57.

34. Cardoso SM, Proença MT, Santos S, Santana I, Oliveira CR. Cytochrome c oxidase is decreased in Alzheimer’s disease platelets. Neurobiol Aging. 2004;25(1):105–10.