Elhesha and Kahveci *BMC Bioinformatics* (2016) 17:408
DOI 10.1186/s12859-016-1271-7

BMC Bioinformatics

RESEARCH ARTICLE Open Access

Identification of large disjoint motifs in biological networks

Rasha Elhesha* and Tamer Kahveci

Abstract

Background: Biological networks provide great potential to understand how cells function. Network motifs, frequent topological patterns, are key structures through which biological networks operate. Finding motifs in biological networks remains to be computationally challenging task as the size of the motif and the underlying network grow. Often, different copies of a given motif topology in a network share nodes or edges. Counting such overlapping copies introduces significant problems in motif identification.

Results: In this paper, we develop a scalable algorithm for finding network motifs. Unlike most of the existing studies, our algorithm counts independent copies of each motif topology. We introduce a set of small patterns and prove that we can construct any larger pattern by joining those patterns iteratively. By iteratively joining already identified motifs with those patterns, our algorithm avoids (i) constructing topologies which do not exist in the target network (ii) repeatedly counting the frequency of the motifs generated in subsequent iterations. Our experiments on real and synthetic networks demonstrate that our method is significantly faster and more accurate than the existing methods including SUBDUE and FSG.

Conclusions: We conclude that our method for finding network motifs is scalable and computationally feasible for large motif sizes and a broad range of networks with different sizes and densities. We proved that any motif with four or more edges can be constructed as a join of the small patterns.

Keywords: Biological networks, Motif discovery, Overlap graph, Subgraph isomorphism

Introduction

Biological networks describe how molecules interact to carry out various cellular functions. One common way to represent these networks is to use graphs, where the nodes and the edges represent the interacting molecules and the interactions between these molecules respectively [1]. Studying biological networks has great potential to help understand how cells function and how they respond to extra-cellular stimulants. Such studies have already been used successfully in many applications. Characterizing the variations in drug resistance of different cell lines [2], or identifying the pathways serving similar functions across different organisms [3, 4] are only few examples among many.

Motifs are frequent topological patterns in a given network [5]. Identifying motifs has been one of the key steps in understanding the functions served by biological networks such as gene regulatory or protein interaction networks [6–8]. Motifs can be used to uncover the basic structure and design principles of a network [9]. They are also often considered as the basic building blocks of a network [5] and one of the network local properties [10]. Thus, they can be used to classify networks [11] into functional sub-units. It is worth noting that motifs have been used in various applications like prediction of regulatory elements in genomic sequences [12].

Despite the fact that studying motifs is of utmost importance for network analysis, motifs identification remains to be a computationally hard problem [13]. The roots of the challenges behind motif discovery arise from several reasons. First, even when the motif topology is given, counting motif frequency (i.e. the number of occurrences of this motif), requires solving the subgraph isomorphism problem, which is NP-Complete [14]. Furthermore, when the motif topology is not known in advance, trying out all alternative topologies is infeasible as the number of

*Correspondence: relhesha@cise.ufl.edu
[1]CISE Department, University of Florida, 432 Newell Dr, 32611 Gainesville, Florida, USA

such topologies increases exponentially with the number of edges in the motif.

There are two ways for motif frequency formulation; (i) allow for different copies of the same motif to overlap (i.e., share nodes or edges) or (ii) count disjoint copies of the motif under consideration. Most of the existing methods in the literature on motif counting follow the first formulation. This formulation however has a fundamental drawback arising from the fact that it does not have *downward closure* property. Briefly, this means that the motif frequency does not decrease monotonically as the motif size increases. We discuss this drawback in detail in Sections "Summary of existing methods" along with why it makes it impossible to determine the largest sized motif in a given network. Several algorithms use the second formulation to compute the frequency of a given motif (e.g., [15]). Those algorithms, however, do not scale to large networks. Also, they are limited to small motifs as their time complexities grow exponentially with motif size. We elaborate on these methods in Section "Summary of existing methods" as well.

In this paper, we address the problem of finding motifs in a given network. More specifically, given a target network and a motif size (i.e., number of nodes in the motif), we aim to find the motifs of that size which have a frequency above a user specified threshold in that target network. Unlike most of the methods in the literature, we use the second formulation of motif counting described above, where no two copies of the same motif share an edge, to compute the frequency.

We develop a novel and scalable algorithm to solve the motif identification problem. The central idea of our method, which stands out among the existing literature, is to use a small set of patterns, called the *basic building patterns*. We prove that any motif with four or more edges can be constructed as a combination of these patterns. Following from this observation, our method first finds instances of these patterns. It then iteratively grows motifs by joining known motifs at that iteration with the instances of these patterns. Our algorithm develops efficient mechanisms to avoid a significant fraction of the costly isomorphism tests while growing new motifs. Counting non-overlapping instances of a given motif is a computationally challenging task that requires solving maximum independent set (MIS) problem which is known to be NP-complete [13]. We introduce a new and efficient strategy for this purpose. This strategy avoids enumerating the overlapping motif instances. It does this by algebraically computing the overlap count based on the neighbors of the motif nodes in the target network. Our experiments on both protein-protein interaction (PPI) and synthetic networks demonstrate that our method is significantly faster and more accurate than the existing methods. In addition, the increase in the running time of our algorithm is dramatically less than that of the competing methods as the motif size grows.

The rest of this paper is organized as follows. We present the key definitions needed to discuss our method and the related literature in Section "Background". We describe our motif discovery algorithm in Section "Methods". We experimentally evaluate our method and compare it to the existing algorithms in Section "Results and discussion". We end with a brief conclusion in Section "Conclusions".

Background

In this section, we provide the definitions and the terminology needed to describe our method (Section "Definitions and notation"). We then summarize the key literature tackling similar problems to the one considered in this paper (Section "Summary of existing methods").

Definitions and notation

We represent a given biological network using a graph denoted with $G = (V, E)$. Here, the set of nodes V denotes the set of interacting molecules, and the set of edges E denotes the interactions among them. In the rest of this paper, we use the term graph to denote a biological network. Here, we focus on undirected graphs. Figure 1a represents a graph that contains seven nodes and eight edges.

We say that a graph is *connected* if there is a path between all pairs of its nodes. We say that a graph $S = (V_S, E_S)$ is a *subgraph* of G if $V_S \subseteq V$ and $E_S \subseteq E$. In the rest of this paper, we only consider connected

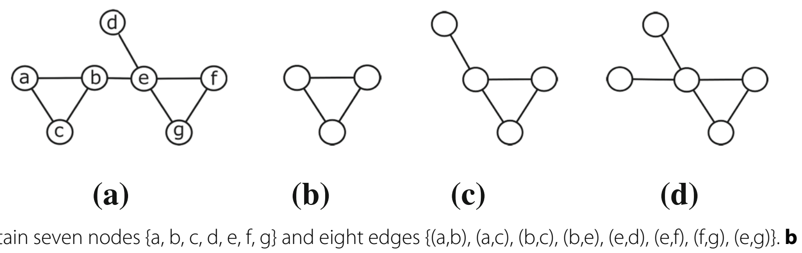


Fig. 1 **a** A graph *G* that contain seven nodes {a, b, c, d, e, f, g} and eight edges {(a,b), (a,c), (b,c), (b,e), (e,d), (e,f), (f,g), (e,g)}. **b** A pattern with two embeddings in *G*, {(a,b), (a,c), (b,c)} and {(e,f), (f,g), (e,g)}. **c** A pattern with three embeddings in *G*, {(a,b), (a,c), (b,c), (b,e)}, {(e,f), (f,g), (e,g), (e,d)}, and {(e,f), (f,g), (e,g), (b,e)}. **d** A pattern that has one copy in *G*, {(b,e), (e,d), (e,f), (f,g), (e,g)}

subgraphs. Thus, to simplify our terminology, we use the term subgraph instead of connected subgraph. Notice that a subgraph of a given graph can be uniquely determined by the set of edges E_S of that subgraph as all of its nodes are connected.

We say that two subgraphs $S_1 = (V_{S_1}, E_{S_1})$ and $S_2 = (V_{S_2}, E_{S_2})$ of G are *identical* if they have the same set of edges. A less constrained association between two subgraphs is *isomorphism*. Two subgraphs S_1 and S_2 are isomorphic if the following condition holds: There exists a bijection $f : V_{S_1} \rightarrow V_{S_2}$ such that $\forall (u, v) \in E_{S_1}, \iff (f(u), f(v)) \in E_{S_2}$.

We say that two subgraphs S_1 and S_2 *overlap* if they share at least one edge (i.e., $E_{S_1} \cap E_{S_2} \neq \emptyset$). In Fig. 1a, consider the four subgraphs S_1, S_2, S_3, and S_4 defined by the set of edges {(a,b), (a,c), (b,c), (b,e)}, {(e,f), (f,g), (e,g), (e,d)}, {(e,f), (f,g), (e,g), (b,e)} , and {(b,e), (d,e), (e,f), (e,g)} respectively. S_1 and S_2 are disjoint as they do not share any edges. S_1 and S_3 overlap as they share the edge (b,e). Similarly S_2 and S_3 overlap. All three subgraphs S_1, S_2, and S_3 are isomorphic as they have the same topology. S_1 and S_4 are non-isomorphic as they do not satisfy the bijection function defined above.

Notice that isomorphism is a transitive relation. Thus, for a given subgraph S of G, the set of all subgraphs of G which are isomorphic to S defines an equivalence class. We represent the subgraphs in each equivalence class with a graph isomorphic to those in that equivalence class and call it a *pattern*. Figure 1c shows the pattern that represents the equivalence class $\{S_1, S_2, S_3\}$.

There are alternative definitions of the frequency of a pattern in a given graph. The classical frequency definition is the number of all subgraphs of the target graph which are isomorphic to the given pattern. This definition, also known as the $F1$ measure [16], counts all the subgraphs regardless of whether they overlap with each other or not. There are two other frequency definitions which avoid overlaps between different subgraphs. $F2$ measure counts the largest subset of subgraphs in a given equivalence class which do not share any edges with the rest of the subgraphs in that subset. It however allows them to share nodes. $F3$ measure is more stringent as it requires that no two subgraphs can share a node. Consider the pattern in Fig. 1c and the target graph in Fig. 1a. The frequency of this pattern in the target graph according to the $F1$ measure is three as it has three embeddings ($\{S_1, S_2, S_3\}$). On the other hand $F2$ is two $\{S_1, S_2\}$, and $F3$ is one (S_1 or S_2 or S_3). From here on, we denote the $F1$, $F2$, and $F3$ counts of a motif M in graph G using the notations $F1_G(M)$, $F2_G(M)$, and $F3_G(M)$ respectively.

The *downward closure* property states that the frequency of a pattern should monotonically decrease as this pattern grows (by inserting new nodes or edges to it). More specifically, consider a function $f()$ that operates on a pattern and returns a real number. Let us denote two patterns with P_1 and P_2. We say that the function $f()$ has downward closure property if and only if $f(P_2) \leq f(P_1)$ for all (P_1, P_2) pairs where P_1 is a subgraph of P_2.

Under the light of these definitions, next we show that $F1$ measure is not downward closed. Consider the pattern P_1 in Fig. 1b. The frequency of P_1 is two in the target graph in Fig. 1a. Now consider the pattern P_2 in Fig. 1c which contains P_1. Although P_1 is a subgraph of P_2, the frequency of P_2 is three in the same graph (i.e., more than that of P_1). Next, consider the pattern P_3 in Fig. 1d. P_3 contains P_2, and its frequency is only one (i.e., less than that of P_2). This example demonstrates that the $F1$ measure not only fails to monotonically decrease, but it also fluctuates (i.e., its value may go up or down) as we grow the pattern (see [17, 18] for further discussions on this issue).

Unlike the $F1$ measure, $F2$ is downward closed. In the following, we formally prove this.

Theorem 1.1 *Assume that we are given a graph G. Given two patterns M and $\bar{M}$ where $M \subset \bar{M}$, we have $F2_G(M) \geq F2_G(\bar{M})$.*

Proof To prove this, we consider the placement of each embedding of $\bar{M}$ in G according to $F2$ measure (i.e. non-overlapping embeddings). Notice that each embedding of $\bar{M}$ contains M as $M \subset \bar{M}$. From each of these embeddings, we remove the edges that are in $\bar{M} - M$. This leads to one embedding of M for each embedding of $\bar{M}$. Thus, the number of non-overlapping embeddings of M in G is at least as much as that of $\bar{M}$ in G. Therefore, $F2_G(M) \geq F2_G(\bar{M})$. □

Similarly, we say that $F3$ measure which also counts non-overlapping embeddings, is also downward closed.

Failure to satisfy the downward closure property has major implications on the correctness of motif identification. Traditional motif identification algorithms often grow a motif starting from an initial motif of a small number of edges (see Section "Summary of existing methods"). Should they employ the $F1$ measure, these algorithms cannot have an early stopping criteria as they grow motifs. This is because the frequency can go up as we grow motif even when the current motif frequency is low. Next, we formally define the problem considered in this paper.

Problem definition Given an input graph $G = (V, E)$, the number of nodes in the target motif μ, and frequency threshold α, we aim to find all patterns of μ nodes which have frequency at least α in G under the frequency measure $F2$. The method we develop in this paper can however be easily extended to $F3$ as well (see Section "Finding MIS: Going from F1 to F2").

Summary of existing methods

We classify the literature on motif identification and counting, based on the underlying frequency measure. This is because the frequency measure dramatically changes the cost of counting motifs as well as how we can interpret the frequency of the underlying pattern. Most of the existing studies use $F1$ frequency measure to count the embeddings of a pattern in a given graph (e.g., [19–24]). These methods carry the drawbacks inherent in the $F1$ measure. First, $F1$ ignores the fact that different copies of the same motif can overlap due to the nodes and the edges they share. This can lead to artificially massive number of motif embeddings as the same node or edge can participate in multiple embeddings. To understand this better, consider the pattern and the graph in Figs. 1c and 1a respectively. $F1$ counts three copies of the pattern (S_1, S_2, and S_3). Different nodes and edges however contribute to this count at different numbers. The edge (a, b) appears only in S_1 while (b, e) appears in both S_1 and S_3.

Second and more importantly, the $F1$ measure is not downward closed. This is because as we grow a pattern by including new edges or nodes, its count as computed by $F1$ is not monotonic; it may decrease, stay the same, or increase. Lack of downward closure property makes it nearly impossible to decide if the motif found is the largest one in size while growing a pattern. Thus, using $F2$ is essential for the tractability of identifying frequent patterns. We use the $F2$ measure in this paper. Thus, the studies limited to the $F1$ measure are out of the scope of this paper.

Several algorithms tackle the problem of finding frequent patterns in multiple graphs. FSG [25] is one of the key methods in this class. These methods, however, do not count the number of occurrences of a pattern in each graph. They rather check if the given pattern appears at least once in each graph. Vanetik et al. [17] also addressed the same problem.

Finding frequent patterns or counting them without overlaps (i.e., using $F2$ or $F3$ measures) have received little attention in the literature. One of the existing algorithms in this category is SUBDUE [15]. Flexible Pattern Finder Algorithm (FPF) [16] detects frequent patterns using both $F2$ and $F3$. Two algorithms were proposed by Kuramochi and Karypis [26], named hSiGraM, vSiGraM. However, these algorithms are computationally expensive and do not scale to large graphs or motifs. We evaluate SUBDUE and FSG experimentally in Section "Results and discussion".

Methods

In this section we describe our method. Section "Algorithm overview" presents an overview of our algorithm. Section "Joining patterns to find larger patterns" explains the mechanism we use to grow motifs by joining smaller motifs. Section "Finding MIS: Going from F1 to F2" describes how we count disjoint motif instances. Section "Accelerating our algorithm through efficient filters" presents filtering techniques we implement to avoid costly isomorphism tests. Section "Complexity analysis" discusses the complexity analysis of our method.

Algorithm overview

In this section, we provide an overview of our method for discovering motifs. At the heart of our method lie four unique graph patterns. We call them the *basic building patterns* for we use them as guide to construct larger motifs of arbitrary sizes and topologies. Figure 2 presents these basic building patterns. We explain why we use these four specific patterns in Section "Joining patterns to find larger patterns" in detail.

Algorithm 1 presents the pseudo-code of our method. We elaborate on each key step of our method in subsequent sections. The algorithm takes a graph G, the number of nodes of the target motif μ, and the minimum acceptable motif frequency as input α. For each of the four basic building patterns, it first locates all subgraphs in G that are isomorphic to that pattern (Line 1). Let us denote the set of instances of the ith pattern ($i \in \{1, 2, 3, 4\}$) with S_i. In each set S_i, it is possible to have overlapping subgraps. It then extracts the maximum set of edge-disjoint subgraphs in each set S_i (Line 2) (see Section "Finding MIS: Going from F1 to F2" for details). Let us denote the resulting set with S'_i for the ith pattern. Notice that the cardinalities of the sets S_i and S'_i are the F_1 and F_2 measures of the ith pattern respectively. The union of all the sets S'_i constitutes the current motif instances as well as the basic building pattern instances at this point (Line 3). The algorithm then iteratively grows the current motif set. At each

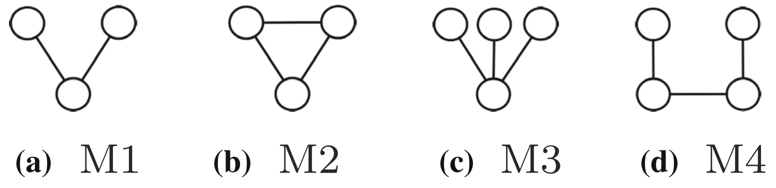


Fig. 2 The four basic patterns used by our algorithm which represent all patterns of two (**a**) or three undirected edges (**b**, **c**, and **d**)

iteration, it joins the current motif set with the basic building pattern set (Line 9). More specifically, a motif instance and a basic building pattern join if they share at least one edge. Joining two such subgraphs either creates a pattern which already exists in the current set (Line 10) or a new pattern (Line 12). At each iteration, after growing the current set, it filters the overlapping subgraphs to identify MIS for each pattern (Line 18). The algorithm removes all patterns with frequency lower than the user supplied cutoff (Line 21). It reports the frequent subgraphs that have as many edges as the target motif size (Line 23). The algorithm terminates when the current set can not be grown to have any other patterns which satisfy the target motif (i.e. each pattern in the current set is either larger than the target motif size or its frequency is lower than the user specified frequency).

Algorithm 1 Motif Discovery algorithm

Input:

- Target motif size μ
- Frequency threshold α
- Input graph $G = (V, E)$

output:

- Motif topologies, and their instance subgraphs, that each have same number of nodes as μ and its $F2 \geqslant \alpha$

```
1: $BPSf1$ = getAllSubgraphs-Isomorphic-to-BasicPatterns()
2: $BPS$ = extract-maxDisjointSubgraphs-PerPattern($BPSf1$)
3: CurrentSet ($CS$) = $BPS$
4: newSet ($NS$) = $\phi$
5: while $CS$ has new patterns and at least one of them with number of nodes $< \mu$ and its $F2 \geqslant \alpha$ do
6:     for each pattern $p1$ in $CS$ do
7:         for each pattern $p2$ in $BSP$ where $p2 \neq p1$ do
8:             for each subgraph $s1 \in p1$ and $s2 \in p2$ do
9:                 $s3$ = join($s1$, $s2$)
10:                if $s3 \in$ existing pattern $P$ then
11:                    add $s3 \in P$ in $NS$ if not duplicate
12:                else
13:                    Create $P_{new}$ with $s3$ topology, add $s3 \in P_{new}$ in $NS$
14:                end if
15:            end for
16:        end for
17:    end for
18:    $CS$= extractmaxDisjointSubgraphsPerPattern($NS$)
19:    for each pattern $p1 \in CS$ do
20:        if $F2$ of $p1 < \alpha$ then
21:            Delete $p1$ and all subgraphs $\in p1$
22:        else if number of nodes of $p1 = \mu$ then
23:            put $p1$ and all subgraphs $\in p1$ in the output
24:        end if
25:    end for
26:    $NS$ = $\phi$
27: end while
```

Joining patterns to find larger patterns

Here, we describe one join iteration of our method; the process of joining the subgraphs of current set of patterns with the subgraphs of the *basic building patterns* to construct larger patterns. At the end of the iteration, the resulting set of subgraphs becomes the current set of subgraphs for the next join iteration.

Recall that we join two subgraphs only if they share at least one edge. Joining two such subgraphs either yields a pattern that is isomorphic to one of the existing patterns or a new one. In the former case, we consider the set of subgraphs S isomorphic to that pattern. We check if the new subgraph is already in S. If it is in S, we discard it. Otherwise, we store it in S. In the latter case (i.e., the pattern is observed the first time), we save this as a new pattern and also keep the corresponding subgraph.

Notice that, although the subgraphs in S do not overlap prior to join, this may no longer hold after new subgraphs are inserted into S. At the end of each join iteration, we select the MIS for each pattern. We defer the discussion on how we do this to Section "Finding MIS: Going from F1 to F2". We then remove the patterns with $F2$ values below the user supplied frequency threshold, α. This eliminates non-promising patterns, and thus, reduces the number of candidate patterns for the next join iteration. Using the $F2$ measure ensures that patterns maintain downward closure property. Thus, non-frequent patterns will never grow to yield frequent patterns.

Why do we need different equivalence classes? If the motif frequency is measured using $F1$, it is sufficient to join the subgraphs belonging to existing patterns with only those which belong to the same equivalence class of the simple pattern with two edges (see Fig. 2a) to construct any larger pattern. This however is not true when $F2$ (or $F3$) is used to count the motif frequency. To understand the rationale behind this, recall that each equivalence class represents a set of disjoint isomorphic subgraphs. As a result, no two subgraphs from the same equivalence class join for they do not share any edges. Therefore we need more than one equivalence class to construct new and larger patterns.

Given that we need multiple patterns, next, we seek the answer to the following question: What is the smallest set of patterns which can be used to produce arbitrary large topologies by joining them? Here we outline the key steps of the proof that the four basic building patterns, presented in Fig. 2, suffice to construct any larger pattern. That said, we do not guarantee to find all copies of such patterns in the target network.

Before we discuss our induction steps, we explain our strategy on a specific motif size of four to improve the clarity of the discussion on induction. Figure 3 shows all the possible patterns which can be constructed with undirected four edges. A careful inspection shows that each

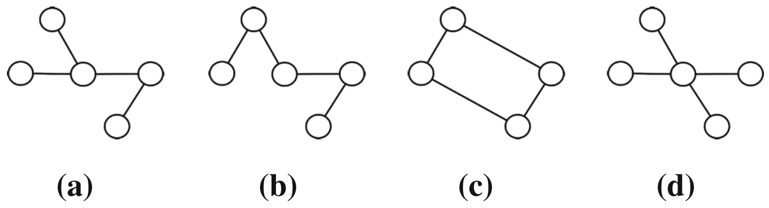


Fig. 3 All patterns which can be constructed with four undirected edges. **a**, **b**, and **d** represent patterns with 4 edges and 5 nodes while **b** represents pattern with four nodes and four edges

one is an overlapping combination of two of the basic building patterns. For instance, the pattern in Fig. 3a can result from joining the basic pattern in Fig. 2a with the basic pattern in Fig. 2c. It is worth noting that we can construct some of the patterns in Fig. 3 by joining two different pairs of basic building patterns. This redundancy ensures we can still locate a specific pattern even if one of those pairs does not exist. Therefore, our method can construct any pattern with four edges from patterns with three or two edges.

We conduct our proof for the arbitrary pattern size by induction.

Basis The four basic patterns in Fig. 2 constitute all possible graph topologies with two or three edges.

Induction step We assume that our method can construct any pattern with up to k edges ($k \geq 3$). We next show that any pattern with $k+1$ edges can be constructed by joining a pattern with k edges with one of the basic building patterns.

Recall that the downward closure property states that those smaller patterns have at least as much frequency as the larger one according to $F2$ (see Theorem 1.1). This means that if a pattern with $k+1$ edges is frequent, then so is any of the k edge patterns obtained by removing an edge from that pattern.

Consider a graph G and a copy of a pattern $P1$ of size k edges in G, S_1. Also, consider a copy of a pattern $P2$ with $k+1$ edges such that $P2$ contains $P1$ and one additional edge. Let us denote this additional edge with (a, b). We need to show that $P2$ can be obtained from $P1$ by joining it with at least one of the basic patterns.

Since both $P1$ and $P2$ are connected graphs, at least one of the two nodes a and b has an edge in $P1$. Without violating the generality of the proof, let us assume that b has an edge (b, c) in $P1$. Figure 4a illustrates the two edges (a, b) and (b, c).

First, we consider using the basic pattern $M1$ in Fig. 2a in the join operation. In this case, a copy of $M1$, $\{(a, b), (b, c)\}$ will join with S_1 having a common edge (b, c) which will result in the pattern $P2$ with $k+1$ edges. This join however occurs only if the subgraph $\{(a, b), (b, c)\}$ is included in the $F2$ counts of $M1$ (i.e. within the chosen non-overlapping copies of $M1$).

If this condition fails, we consider the degrees of the two nodes b and c in pattern $P1$. We start with node c. Let us denote the degree of a node with function $deg()$ (e.g. $deg(c)$ is the degree of node c in pattern $P1$).

If $deg(c) > 1$, then c has at least one more edge on top of (b, c). Let us denote this edge with (c, d) (see Fig. 4b). In this scenario, we join a copy of the motif $M4$ (Fig. 2d), $\{(a, b), (b, c), (c, d)\}$ (if this copy exists in the $F2$ count of $M4$) to obtain $P2$.

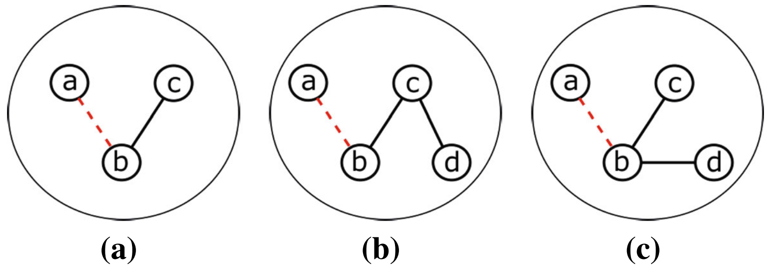


Fig. 4 a A subgraph S_2 in a hypothetical graph G. S_2 is isomorphic to a pattern $P2$ of size $k+1$ edges. If we remove the additional edge (a, b) we obtain S_1 which is isomorphic to P_1 where $P_1 \subset P_2$. Notice that S_1 could have arbitrary $k-1$ edges rather than (b, c). Here we obtain S_2 as a result of joining S_1 with the subgraph $\{(a, b), (b, c)\}$ which belongs to $M1$ equivalence class (see Fig. 2a). **b** Failure to accomplish the join in (**a**), we seek to inspect $deg(c)$ and $deg(b)$ in S_1. The first possibility is that $deg(c) > 1$. This means that the subgraph $\{(b, c), (c, d)\}$ exists. We then can join S_1 with the subgraph $\{(a, b), (b, c), (c, d)\}$ which belongs to $M4$ equivalence class (see Fig. 2d) to obtain S_2 which is isomorphic to a pattern $P2$ of size $k+1$ edges. **c** The second possibility is that $deg(b) > 1$. This means that the subgraph $\{(b, c), (b, d)\}$ exists. We then can join S_1 with the subgraph $\{(a, b), (b, c), (b, d)\}$ which belongs to $M3$ equivalence class (see Fig. 2c) to obtain S_2

Finally, if $deg(c) = 1$, it is guaranteed that $deg(b) > 1$. This is because if both nodes b and c have degree one, $S1$ cannot be a connected subgraph. Let us denote one of the additional edges of b with (b, d) (see Fig. 4c). In this case, we join the subgraph that isomorphic to the pattern $M3$, $\{(a, b), (b, c), (b, d)\}$, with S_1 to obtain $P2$. We can do this if this copy exists in the $F2$ count of $M3$.

In summary, we conclude that any pattern $P2$ with $k+1$ edges can be constructed by joining a pattern $P1$ with k edges (or $k-1$ edges) and one of the basic building patterns to obtain the additional edge (or edges) if at least one of the many possible scenarios hold. We however cannot guarantee that the joins will find all of the instances of the $k+1$ edge pattern on the target graph.

Recall that as we aim to calculate the frequency of a given motif using $F2$, there is no self join of any pattern. Thus, the basic building patterns set is the smallest set of patterns as we can not construct one of those four patterns using the three other patterns. More specifically, this means that we can not use only one of those four basic building patterns to construct larger patterns by joining pairs of subgraphs belong to that pattern's equivalence class. This is because if we join the embeddings of a single motif topology (such as the first pattern in Fig. 2a) we cannot get any larger pattern as they do not share any edge(s).

Finding MIS: Going from F1 to F2

Here, we explain how we compute the $F2$ frequency for a given pattern. We use two algorithms for this purpose. We explain why we have two separate algorithms later in this section after describing the two algorithms. The first one is a heuristic used in the literature [16]. This algorithm constructs a new graph, called the *overlap graph* for each pattern as follows. Each node in the overlap graph of a pattern denotes an embedding of that pattern in the target graph. We add an edge between two nodes of the overlap graph if the corresponding embeddings represented by those nodes overlap in the original graph. Once the overlap graph is constructed, the algorithm starts by selecting the node with the minimum degree (i.e. overlaps with the minimum number of embeddings) in the overlap graph. We include the subgraph represented by this node in the edge-disjoint set. We then delete that node along with all of its neighboring nodes in the overlap graph. We update the degree of the neighbors of the deleted nodes. We repeat this process of picking the smallest degree node and shrinking the overlap graph until the overlap graph is empty.

The algorithm described above works well for patterns with small number of embeddings. It however becomes computationally impractical as the number of embeddings of the underlying pattern gets large. This is because both constructing the overlap graph (particularly identifying its edges) and updating it are computationally expensive tasks. Therefore, we use this algorithm for all patterns except for the basic building patterns (where number of embeddings are often too large).

The second algorithm addresses the scalability issue of the the first one. This scalability issue is imposed by the expensive task of calculating the degree of each node in the overlap graph (i.e. the number of overlaps of each embedding). Recall from the previous algorithm that this number is considered as a loss value when selecting the node (i.e. embedding) with minimum degree (i.e. number of overlaps) to include in the final MIS of the pattern under consideration. Briefly, the second algorithm we introduce here avoids the expensive task of calculating number of overlaps for each embedding. The algorithm performs this by *algebraically* computing such numbers instead of performing actual overlapping tests. Once we compute node degrees of the overlap graph, this algorithm selects the disjoint embeddings the same way as the former algorithm described before. More specifically, the algorithm selects the node with the minimum degree and includes its corresponding embedding in the final MIS. It then removes neighboring nodes to that node from the overlap graph. It repeats this process until the overlap graph is empty. Next, we explain how we compute the degree of a node in the overlap graph for the pattern $M1$ in Fig. 2a. Our computation is similar for the other three basic building patterns, yet tailored towards their specific topologies (derivation is shown in Additional file 1: Appendix). Figure 5 shows a hypothetical subgraph $S_1 = \{(a, c), (b, c)\}$ in the input graph G which is isomorphic to $M1$. This subgraph is represented by a node in the overlap graph of $M1$'s embeddings. Let us denote the degree of a node in the original graph G with function $d()$ (e.g. $d(v_i)$ is the degree of node v_i). Another embedding of $M1$ in G overlaps with S_1 only if it contains the edge (a, c), or (b, c). Any edge in G connected to the middle

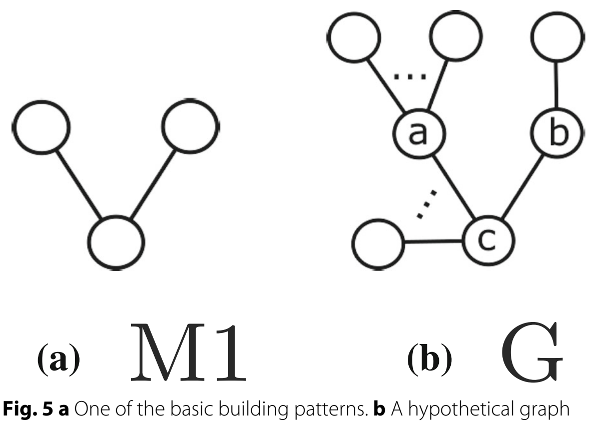


Fig. 5 a One of the basic building patterns. **b** A hypothetical graph that contains subgraphs isomorphic to the pattern M1 in (**a**) node c forms two overlapping embeddings, one with the subgraph that has edge the (a, c) and the other with the subgraph that has the edge (b, c). We exclude the edges belong to S_1 (i.e. the embedding we want to calculate its number of overlaps) itself from the potential edges of G that considered in the overlapping embeddings with S_1. Thus, by excluding the two edges (a, c) and (b, c) from c's degree, node c yields $2 \times (d(c)$ - 2) overlaps. In addition, any edge that belongs to node a forms an embedding when combined with the edge (a, c). Excluding the edge (a, c), node a yields $d(a)$ - 1 overlaps. Similarly, node b produces $d(b)$ - 1 overlaps. Thus, the total number of overlaps for the embedding $S_1 = \{(a, c), (b, c)\}$ combined from edges of its three nodes $\{(a, b, c)\}$ is

$$2(d(c)-2)+d(a)-1+d(b)-1 = 2d(c)+d(a)+d(b)-6$$

Notice that unlike the first algorithm, the second one requires a unique derivation for each pattern. Thus, we apply it only to the basic building patterns, for their topologies do not depend on the input graph. Also, it is worth noting that typically the basic building blocks have much larger number of embeddings as compared to the patterns derived by joining them. Thus, the efficiency of the second algorithm is needed for them more than the patterns obtained in subsequent iterations (see experimental results).

To adapt our method to count non-overlapping embeddings of each pattern according to $F3$ instead of $F2$, we only need to change how we calculate the MIS of this pattern. More specifically, we change the criteria which states that "two subgraphs overlap if they share at least one edge" to "two subgraphs overlap if they share at least one node" (see Section "Definitions and notation"). This will result in changing the overlap graph constructed using the first method we explain in this section. In addition, it will also have slight change in calculating the total number of overlap of each embedding using the second method we discuss in this section. Practically, we expect the overlap graph to be denser when we use the $F3$ measure as compared to that for the $F2$ measure. To illustrate this, consider the graph G in Fig. 1a and the pattern in Fig. 1c. This patter have 3 embeddings in G which are S_1, S_2, and S_3 defined by the set of edges {(a,b), (a,c), (b,c), (b,e)}, {(e,f), (f,g), (e,g), (e,d)}, {(e,f), (f,g), (e,g), (b,e)} respectively. Figure 6a and Fig. 6b represent the overlap graph of this pattern based on $F2$ and $F3$ measures respectively.

Accelerating our algorithm through efficient filters

Recall that at each iteration, our algorithm generates new subgraphs. For each of these subgraphs, it checks if this subgraph is isomorphic to one of the patterns constructed till that iteration. Isomorphism test is a computationally expensive task. Next, we describe how we avoid a large fraction of these tests.

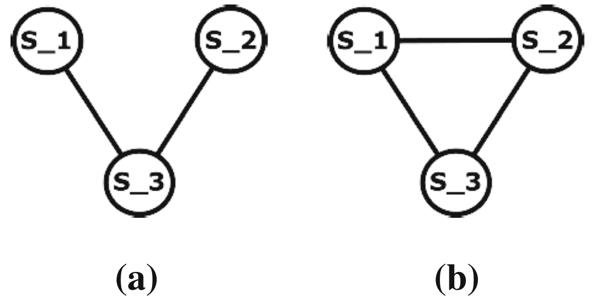


Fig. 6 a The overlap graph of the pattern in Fig. 1c based on F_2 measure of this pattern in the graph in Fig. 1a. **b** The overlap graph of the same pattern based on F_3 measure

We develop two canonical labeling strategies for patterns. Canonical labeling assigns unique labels to the nodes of a given pattern [27]. If two patterns are isomorphic, then they have the same canonical labeling. The inverse is however not true. Unlike isomorphism test, comparing the canonical labeling is a trivial task. Following from this observation, when we construct a new subgraph, we first compare its canonical labeling to those of existing patterns. We then limit the costly isomorphism test to only those patterns which have the same canonical labeling as the new subgraph.

The first canonical labeling counts the degree (i.e. number of incident edges) of each node in the given pattern. It then sorts those degrees and keeps them as a vector we call the *degree vector*. If two patterns have different degree vectors, then they are guaranteed to have different topologies. Despite its simplicity, this labeling filters out a large fraction of patterns. To test its efficiency, we have tested it on random graphs generated using Barabási—Albert model [28]. We generate 1000 pairs of graphs where each pair is non-isomorphic and have the same number of nodes and edges. The degree vector successfully filters 85 % of the 1000 experiments.

The second canonical labeling extends on the first one. It was first introduced by [29]. Consider a pattern $P = (V, E)$. Let us define the distance between two nodes v_i, $v_j \in V$ as the number of edges on the shortest path that connects v_i and v_j and denote it with x_{ij}. Let us define the diameter of P as the maximum distance between any two nodes, and denote it with x. Using this notation, we assign label to node v_i as: $\sum_{j}^{j \in V} 2^{x-x_{ij}-d(v_j)}$. Once we compute the labels of all the nodes in the given pattern, we sort them. We call the resulting vector the *nodes vector*. Similar to the first labeling above, two isomorphic graphs are guaranteed to yield the same labeling. We compute and compare the nodes vector with only the patterns which cannot be eliminated using the first canonical labeling. We then consider the patterns with identical canonical labels for graph isomorphism.

Complexity analysis

Here we analyze the complexity of our method. We refer to Algorithm 1 as we discuss the steps of our method. For each steep, we explain its complexity. We then summarize the complexity of all steps to denote the overall complexity of our method. These steps are

- **Find all subgraphs isomorphic to each of the four basic patterns (Line 1)**: In this step, we analyze each of the four basic patterns separately since they have different topologies. For the pattern $M1$ in Fig. 2a, to get all subgraphs isomorphic to this pattern, we consider all edges connected to each node in the underlying network. We select any two edges combination connected to every node. Here, we denote the degree of a node with function $d()$ (e.g. $d(v_i)$ is the degree of node v_i). Thus, the complexity of collecting subgraphs that are isomorphic to $M1$ is $\sum_{v_i \in V} \binom{d(v_i)}{2}$. Similarly, for the pattern $M3$ in Fig. 2c, we select any three edges combination connected to each node in G. Thus, the complexity of constructing subgraphs which are isomorphic to $M3$ is $\sum_{v_i \in V} \binom{d(v_i)}{3}$. For the pattern $M2$ in Fig. 2b, we consider each edge e_{ij} in G with two nodes v_i and v_j. We collect edges of both nodes. We then select one edge connected to v_i and one edge connected to v_j (on the condition that these two edges are connected from the other end) along with e_{ij} to form a subgraph isomorphic with $M2$. Thus, the complexity of constructing subgraphs that are isomorphic to $M3$ is $\sum_{e_{ij} \in E} d(v_i)d(v_j)$. Similarly to $M2$, we perform the same operation to get isomorphic subgraphs to the pattern $M4$ in Fig. 2d. Only this time we make sure that the two edges belong to v_i and v_j are not connected with each other from the other end. Thus, the complexity of constructing subgraphs that are isomorphic to $M4$ is $\sum_{e_{ij} \in E} d(v_i)d(v_j)$. Collectively, the complexity of performing this step is $\mathcal{O}(\sum_{v_i \in V} d(v_i)^3 + \sum_{e_{ij} \in E} d(v_i)d(v_j))$. Notice that, theoretically, the worst case scenario happens when $d(v_i) = \mathcal{O}(n)$. In this scenario, the complexity of this step becomes $\mathcal{O}(n^4)$.
- **Extract maximum disjoint set for basic patterns (Line 2)**: In this step, we use the algebraic algorithm described in Section "Finding MIS: Going from F1 to F2" (second one) to calculate the number of overlaps of each subgraph belonging to each pattern equivalence class. This process takes constant time. We calculate this algebraic equations as we construct subgraphs in the previous step. We then sort those subgraphs within each equivalence class in decreasing order of their number of overlaps. This process has complexity equal to $\mathcal{O}(mlog(m))$ where m is the number of subgraphs in each equivalence class. Recall from previous step that this number is $\mathcal{O}\left(\sum_{v_i \in V} d(v_i)^3 + \sum_{e_{ij} \in E} d(v_i)d(v_j)\right)$. Thus, the complexity of this step is $\mathcal{O}\left(\left(\sum_{v_i \in V} d(v_i)^3\right) log\left(\sum_{v_i \in V} d(v_i)^3\right) + \left(\sum_{e_{ij} \in E} d(v_i)d(v_j)\right) log\left(\sum_{e_{ij} \in E} d(v_i)d(v_j)\right)\right)$.
- **Join Iterations (Lines 5–27)**: In this step, we analyze the complexity of one join iteration. We then summarize the complexity of all join iterations. Let us denote the number of current patterns in iteration i with x_i. Notice that, for the first iteration $x_i = 4$. Recall that in each join iteration, we increase the size of each of the current patterns with one or two edges. In addition, the patterns of the first join iteration are at least of size 2. Thus, the size (i.e. number of edges) of each of the current patterns in iteration i is at least $i + 2$. The number of subgraphs isomorphic to each of the current patterns is at most $\frac{|E|}{i+2}$ since they are non-overlapping subgraphs. Recall that the subgraphs of the basic patterns are non-overlapping within each pattern. Thus, the number of subgraphs of the patterns $M1$, $M2$, $M3$, and $M4$ are $\frac{|E|}{2}$, $\frac{|E|}{3}$, $\frac{|E|}{3}$, and $\frac{|E|}{3}$ respectively. Collectively, the number of subgraphs of the basic patterns is $\mathcal{O}(|E|)$.
  In the join iteration, we start by joining subgraphs of current patterns with the subgraphs of the basic patterns (Lines 6–9). Thus, the total number of joins we perform at iteration i is $\mathcal{O}\left(|E|\frac{|E|}{i+2}x_i\right)$. For each join, we compare the resulting subgraph against all patterns (Line 10). Recall that, we use filters to avoid this costly isomorphism check (see Section "Accelerating our algorithm through efficient filters"). Thus, the complexity of this operation is $\mathcal{O}(x_i)$. If this subgraph is isomorphic to one on the current patterns, we check whether this subgraph is a duplicate of one of the subgraphs which already exists in this equivalence class (Line 11). We search an indexed list of those subgraphs in $\mathcal{O}\left(log\left(\frac{|E|}{i+2}\right)\right)$. Collectively, we obtain the complexity of performing all joins at iteration i by multiplying the three complexities above and get $\left(|E|\frac{|E|}{i+2}x_i x_i log\left(\frac{|E|}{i+2}\right)\right)$, which equals$\mathcal{O}\left(x_i^2 \frac{|E|^2}{i+2} log\left(\frac{|E|}{i+2}\right)\right)$.
  Upon completing all join operations, our algorithm extracts the MIS for each pattern (Line 18) using the overlap graph algorithm described in Section "Finding MIS: Going from F1 to F2" (first one). Notice that we perform this operation for the new set of patterns, x_{i+1} (current patterns of next iteration) for which the number of patterns is at most $\frac{|E|}{i+3}$ (This is because each pattern is of size $i + 3$ and no two patterns overlap). For each pattern, we collect

the overlapped subgraphs of each subgraph in $\mathcal{O}\left(\left(\frac{|E|}{i+3}\right)^2\right)$. We then sort the subgraphs in decreasing order of their number of overlaps in $\mathcal{O}\left(\frac{|E|}{i+3} log\left(\frac{|E|}{i+3}\right)\right)$ time. Thus we extract the MIS for all patterns in $\mathcal{O}\left(x_{i+1}\left(\frac{|E|}{i+3}\right)^3 log\left(\frac{|E|}{i+3}\right)\right)$.
Finally, we check each resulting pattern (Line 19–25) and delete it if its frequency is less than the threshold α. We perform this step in $\mathcal{O}(x_{i+1})$ time.
Recall that in each join iteration, we increase the size of each of the current patterns with one or two edges. Also recall that we start the with patterns of at least of size 2. Thus, total number of join iterations we perform until we reach to all patterns are at least of the target motif size is $\mu - 2$. Thus, the complexity of all join iterations is $\mathcal{O}\left(\sum_{i=1}^{\mu-2}\left(x_i^2 \frac{|E|^2}{i+2} log\left(\frac{|E|}{i+2}\right) + x_{i+1}\left(\frac{|E|}{i+3}\right)^3 log\left(\frac{|E|}{i+3}\right) + x_{i+1}\right)\right)$ or simply $\mathcal{O}\left(\sum_{i=1}^{\mu-2}\left[x_i \frac{|E|^2}{i} log\left(\frac{|E|}{i+2}\right)\right]\left[x_i + \frac{|E|}{i^2}\right] + x_{i+1}\right)$

In summary, the complexity of our method considering all the previous steps is

$$\mathcal{O}\Bigg(\left(\sum_{v_i\in V} d(v_i)^3\right)\left(1 + log\left(\sum_{v_i\in V} d(v_i)^3\right)\right) + \left(\sum_{e_{ij}\in E} d(v_i)d(v_j)\right)\left(1 + log\left(\sum_{e_{ij}\in E} d(v_i)d(v_j)\right)\right) + \sum_{i=1}^{\mu-2}\left(\left[x_i\frac{|E|^2}{i} log\left(\frac{|E|}{i+2}\right)\right]\left[x_i + \frac{|E|}{i^2}\right] + x_{i+1}\right)\Bigg)$$

Notice that x_i here depends significantly on the topology and the density of the given network G. To the best of our knowledge, there is no closed formula that calculates x_i (i.e. the number of unique topologies of certain size in a given graph G).

Results and discussion

In this section, we experimentally evaluate the performance of our motif discovery algorithm on synthetic and real graphs (Section "Evaluation of running time"). We measure the running time and accuracy of our algorithm. We compare our algorithm to two state of the art algorithms, FSG [25] and SUBDUE [15] (Section "Comparison with existing methods"). We evaluate the statistical significance of the most abundant motif in each of the real graph (Section "Evaluation of statistical significance"). We present a case study of the motifs identified by our method on Human herpesvirus PPI network (Section "Case study on Human herpesvirus"). In all of our experiments, we report the motif frequency using the $F2$ measure.

Data set We use real and synthetic datasets in our experiments. The real graphs are the PPI networks of seven organisms taken from the MINT database [30] (see Table 1 for details). We first remove the nodes and edges of these graphs which are guaranteed to not be a part of the motif to be found. To do that, we filter a subset of the nodes of each network as follows. We first identify connected subgraphs of each graph. Let us denote the size of the motif we aim to find with μ. We remove the connected subgraphs with less than μ nodes. Table 1 lists these networks and their sizes after filtering them for $\mu = 5$ (which is the smallest motif size in all of our experiments).

In addition to the real dataset, we construct synthetic graphs. The purpose of having synthetic dataset is to systematically evaluate our method by varying network characteristics (network size and density) in a controlled environment. We build this dataset using the Barabási—Albert model [28] for it captures the connectivity patterns of real networks [31–33]. Moreover, this model has been frequently used in the literature to simulate real networks.

Implementation and environment We implement our algorithm in C++ and perform experiments on a computer equipped with AMD Opteron(tm) Processor 1.4 GHz CPU, 500 GBs of main memory running Linux operating system.

Evaluation of running time

In this experiment, we evaluate the running time of our motif discovery algorithm. Our goal here is to observe the effect of varying parameters; graph size, graph density, and motif size on the running time of our algorithm.

Effect of graph and motif size

We evaluate the running time of our method under varying graph and motif sizes using both synthetic and real datasets.

Results on synthetic graphs We generate synthetic graphs of varying size (i.e. number of nodes) from 100 to

Table 1 The size (number of Proteins and interactions) of the PPI networks selected from the MINT database

Network name	Network code	Number of proteins	Number of interactions
Human herpesvirus8	hhv-8	48	82
Campylobacter jejuni	cje	109	117
Treponema pallidum	tpa	108	173
Rattus norvegicus	rno	535	643
Helicobacter pylori	hpy	717	1472
Escherichia coli	eco	616	1561
Plasmodium falciparum	pfa	1221	2577

1000 at increments of 100. We fix the graph density to two edges per node on the average (i.e., mean node degree is set to four). We set the minimum desired motif frequency, $\alpha = 10$. We run experiments for motif sizes $\mu = 5$, 10, and 15 and report the running time. Figure 7 presents the results.

The results demonstrate that our method scales well with growing graph and motif sizes. The running time grows with increasing graph and motif sizes, yet it remains practical for very large graphs. For motif sizes of 5 and 10, it runs in only several minutes even for the largest input graph. As the motif size grows, the cost increases. However, our method can identify very large motifs in a little over a day for massive networks. We observe that the motif size has more influence on the performance of our method than the input graph size. This is because the number of alternative motif topologies grow exponentially with the motif size. This is an inherent characteristic of the underlying computational problem. However, even when the motif size is 15 our method remains to have a practical running time.

Results on real graphs Next, we test our method on real dataset. We set the minimum desired motif frequency, $\alpha = 5$. We run experiments for motif sizes $\mu = 5$, 10, and 15 and report the running time. Figure 8 presents the results. Similar to the synthetic dataset results, our method scales to large graph and motif sizes on the real dataset. Note that the number of alternative motif topologies grows exponentially with the motif size. Furthermore, the cost of subgraph isomorphiosm also grows exponentially with the motif size. Despite these two major complicating factors, the running time of our method increases only by about an order of magnitude when we increase the motif size by five. Finally, the parallel between these results and those in Fig. 7 suggests that synthetic graphs generated by Barabási—Albert model have similar structural properties as the real PPI graphs.

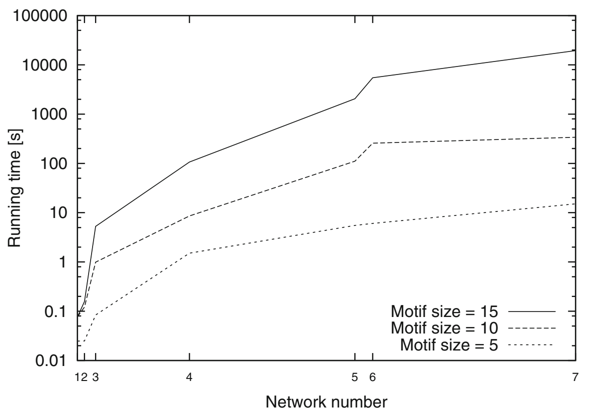


Fig. 8 The total running time of our method for the real PPI networks. Network numbers 1 to 7 on the x-axis correspond to hhv-8, cje, tpa, rno, hpy, eco, and pfa PPI networks respectively. The positions of the PPI networks on the x-axis indicate the sizes of the input graphs (see Table 1). The y-axis shows the running time in seconds

Effect of graph size and density

Here, we evaluate the effect of varying input graph size and density on the running time of our algorithm. We use synthetic dataset in order to control the graph density in this experiment. We generate synthetic graphs varying network size from 100 to 1000 at increments of 100. We set the desired motif frequency $\alpha = 5$ and the motif size $\mu = 10$. We vary graph density from one to four which covers broad range of biological networks [34]. For each input graph and density value, we report the total running time. Figure 9 presents the results.

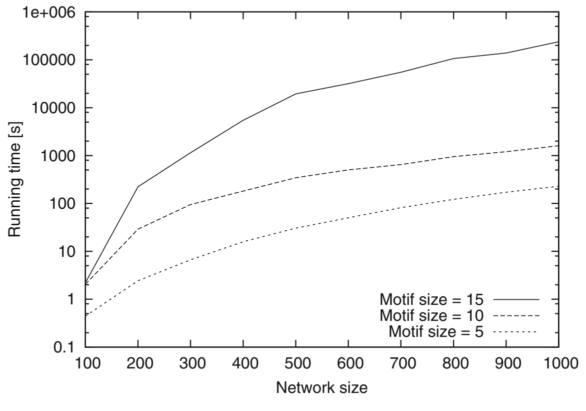


Fig. 7 The total running time of our method for varying graph size and motif sizes (number of nodes). Motif size varies from 5 to 15. The x-axis shows the input graph sizes varying from 100 to 1000. The y-axis shows the total running time in seconds

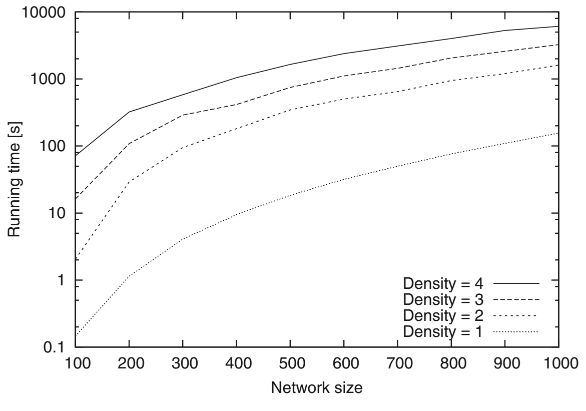


Fig. 9 The total running time of our method for the synthetic graphs with different graph sizes (number of nodes) and varying graph densities from 1 to 4. The x-axis shows the input graph sizes. The y-axis shows the total running time in seconds

We observe that the running time increases with growing graph density. As the graph density increases, the number of alternative embeddings of a given motif grows as well. This also increases the number of overlapping subgraph pairs, which in turn increases the cost of finding MIS for each pattern to calculate its $F2$ frequency (see Section "Finding MIS: Going from F1 to F2"). Despite these major complications inherent in the nature of the motif counting problem, our method remains scalable with respect to growing density. These results suggest that our method is reliable and computationally feasible for a broad range of networks with different sizes and densities.

Comparison with existing methods

Here, we compare our method against two methods in the literature which are tailored towards a problem similar to the one considered in this paper, namely SUBDUE and FSG. We measure the running time and accuracy. We compute accuracy in terms of three parameters, the number of unique motifs found, the average frequency per motif in the target graph, and the frequency of the most abundant motif.

Of these two methods, for SUBDUE, we only report the accuracy of the result as we observe that for most datasets and motif sizes, SUBDUE fails to identify motifs (results shown later in this section). For FSG, we only report the running time. This is because FSG finds motifs in multiple graphs, limited to at most one embedding per graph. In other words, it cannot find multiple embeddings of the same motif in a single graph. Therefore, FSG would yield very low accuracy when applied to a single graph. In the rest of the paper, we will refer to our method as MD (Motif Discovery) for simplicity.

Comparison with SUBDUE

In this experiment, we analyze the effect of varying input graph and motif sizes on the accuracy of our method as compared to that of SUBDUE. We use real dataset in this experiment (see Table 1). SUBDUE does not allow the user to set a minimum allowable motif frequency parameter. It finds all subgraph topologies of a given size even for those subgraphs that appear only once. Due to this limitation of SUBDUE, to have a fair comparison, we set $\alpha = 1$ for our method as well. We follow our earlier definition (see "Definitions and notation"), and use motif size μ to denote the number of nodes in the given motif topology. We run both methods on each input graph using motif sizes $\mu =$ 5, 10, and 15. We report the accuracy of our method as well as SUBDUE. Figures 10, 11, and 12 present the results of $\mu = 5$, 10, and 15 respectively.

Our results for $\mu = 5$ (Fig. 10) demonstrate that both methods identify similar number of unique motifs, yet our method outperforms SUBDUE significantly in terms of the average frequency per motif in all cases (see Fig. 10b).

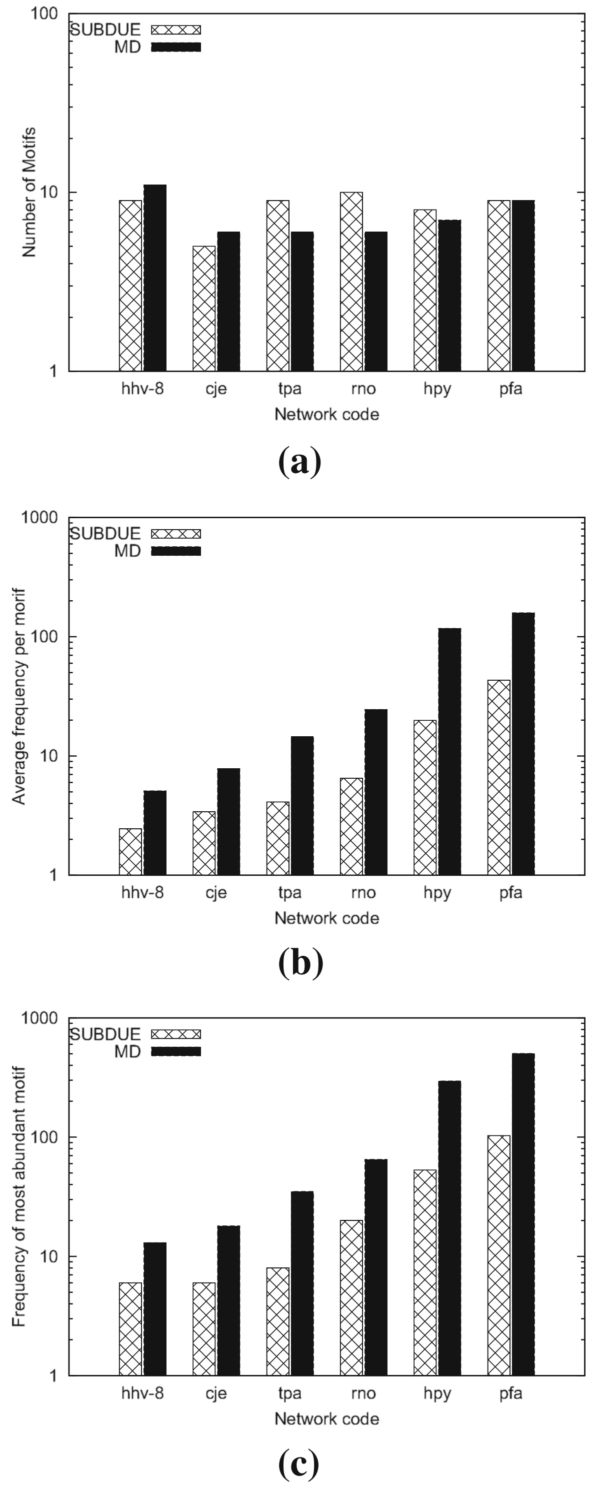


Fig. 10 The accuracy of our method (MD) and SUBDUE in terms of three measures **a** the number of unique motif topologies found, **b** the average frequency per motif in the target graph, and **c** the frequency of the most abundant motif. Results are for the motif size $\mu = 5$ on the real dataset (see Table 1)

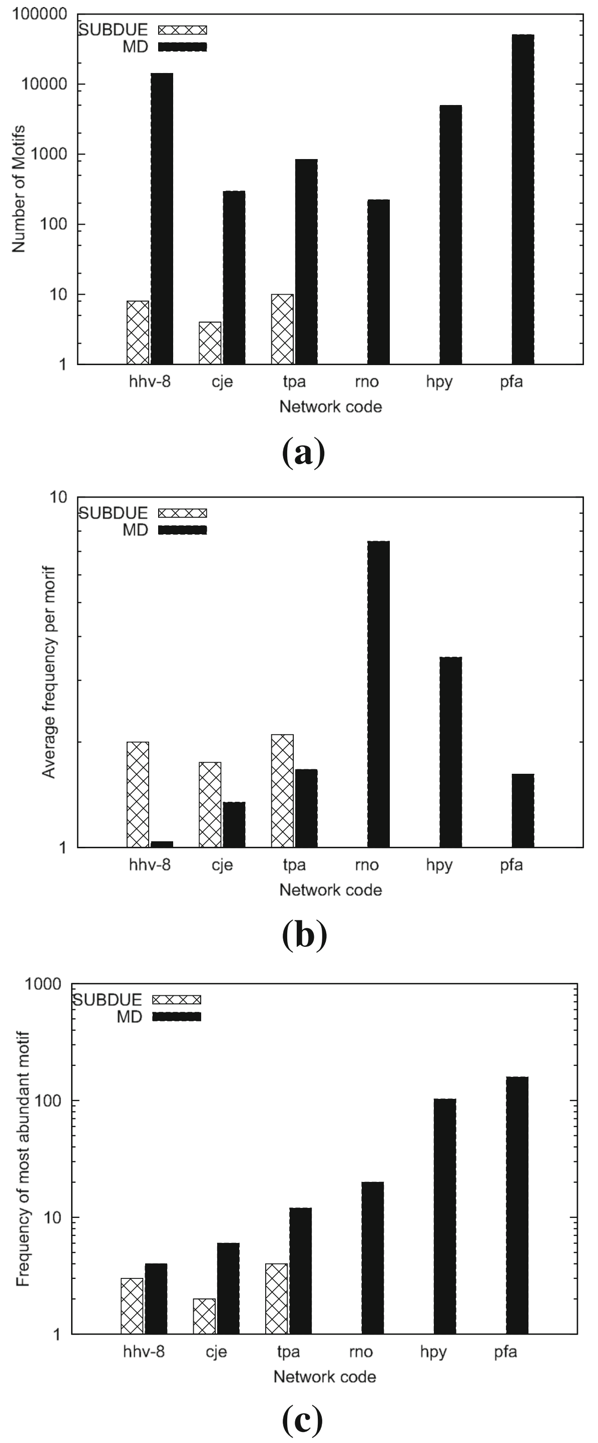


Fig. 11 The accuracy of our method (MD) and SUBDUE in terms of three measures **a** the number of unique motif topologies found, **b** the average frequency per motif in the target graph, and **c** the frequency of the most abundant motif. Results are for the motif size $\mu = 10$ on the real dataset (see Table 1)

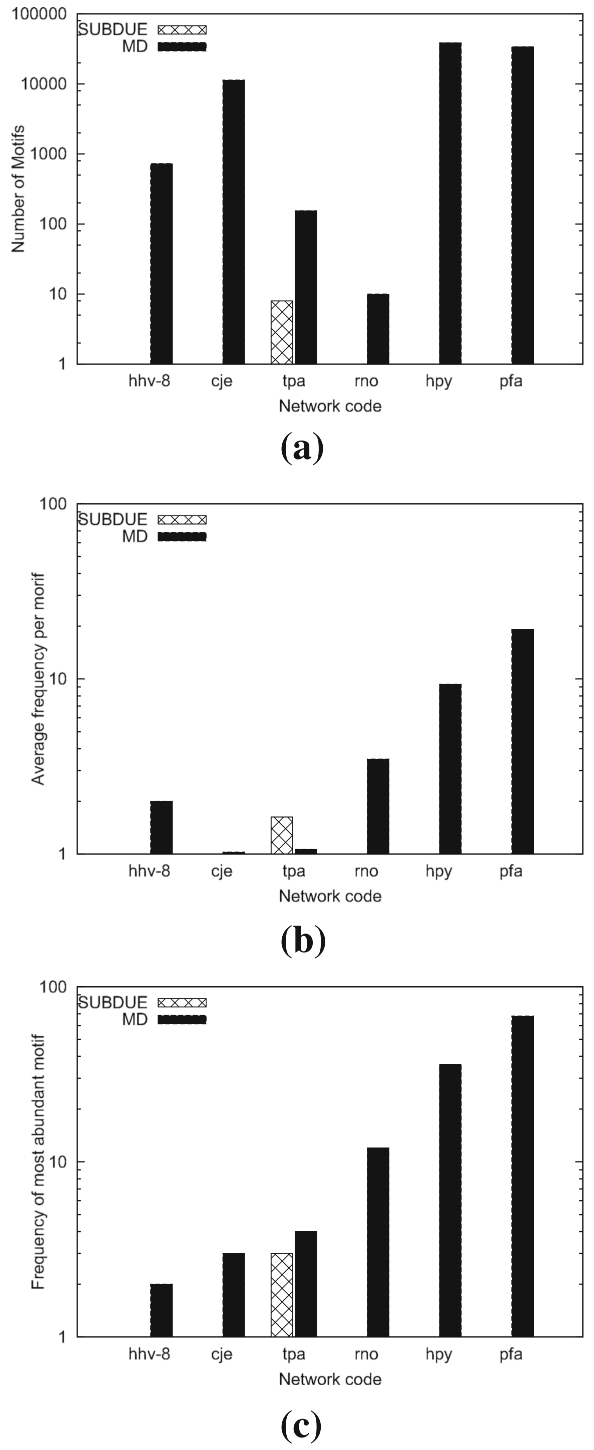


Fig. 12 The accuracy of our method (MD) and SUBDUE in terms of three measures **a** the number of unique motif topologies found, **b** the average frequency per motif in the target graph, and **c** the frequency of the most abundant motif. Results are for the motif size $\mu = 15$ on the real dataset (see Table 1)

When we focus on the most abundant topology of each method, we observe a similar pattern; our method always finds patterns with much higher frequency than SUBDUE in all the experiments (see Fig. 10c). It is worth nothing that motif discovery problem gets exponentially harder with growing motif size. As a result, we expect most algorithms tailored for motif identification to perform well for small motif sizes such as $\mu = 5$. Next, we observe how our method and SUBDUE perform for large values of μ.

As we grow the motif size to $\mu = 10$ (Fig. 11), the results suggest that the gap between our method and SUBDUE grows rapidly in terms all three accuracy measures. More importantly, the results also show that in half of the cases, particularity where the input graph size is large, SUBDUE could not find any motifs while our method continue to locate patterns with high frequency. For example, our method was capable of finding motif topologies with frequency over 100 while SUBDUE could not locate any motif (see Fig. 11c).

For few cases (see Fig. 11b), (hhv-8, cje, and tpa), the average frequency per motif of SUBDUE is slightly higher than that of our method. This is because, we set the minimum frequency $\alpha = 1$. Our method locates many topologies which exist only once while SUBDUE fails to locate them. For example, our algorithm finds thousands of unique motif topologies while subdue outputs only 8 motif topologies for the hhv-8 organism (see Fig. 11a). As a result, these unique topologies pull the average frequency down. That said, Fig. 11c confirms that our method can identify motifs which are more frequent than those found by SUBDUE even for those organisms.

As we further increase the motif size to $\mu = 15$ (Fig. 12), the significance of our method becomes more prevalent. We observe that SUBDUE could not find any motifs in any of the graphs accept for tpa's PPI network. On the other hand, our algorithm not only identifies a massive number of patterns (see Fig. 12a), but also some of these patterns have very large frequencies (see Fig. 12c).

In summary, the results demonstrate that our method scales to large input graph and motif sizes and continue to locate patterns with high frequency for a broad range of motif and input graph sizes while SUBDUE fails to do so.

Comparison with FSG

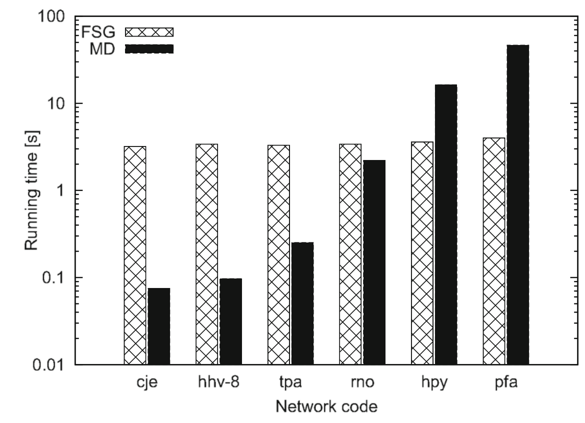
In this experiment, we compare the effect of different input graph and motif sizes to the running time of our algorithm and that of FSG. We use real dataset in this experiment (see Table 1). FSG method requires multiple graphs as input. It defines the frequency of the motif topology as number of different graphs that this motif appears within. Since our method operate on one input graph , we set the desired motif frequency $\alpha = 1$ to be consistent with FSG. FSG defines motif size as the number of edges in the given motif. To be consistent with FSG, we use μ to denote the number of edges in the motif in this experiment. We run both methods on each input graph using motif sizes $\mu = 7$, 8, and 9. We report the running time of our method (MD) as well as FSG. We do not run experiments for $\mu > 9$ as FSG fails to scale to large motif sizes unlike our method. Figure 13 presents the results.

We observe that our method (MD) is orders of magnitude faster than FSG, particularly in large motif sizes. The running time of our method increases slowly with both motif size and the graph size. On the other hand, the running time of FSG increases slowly with the input graph size, but very rapidly with the motif size. Only for a few cases of small motif sizes (i.e ≤ 7 edges) FSG performs better than our method. This is due the overhead of calculating F2 for the basic building patterns where number of overlapped embeddings is huge. That said, the running time difference in those cases are negligible. These results suggest that our method outperforms FSG in terms of running time for a broad range of input real biological networks with different sizes. This performance advantage is further magnified by the fact that our method can find multiple embeddings of each motif while FSG finds only one. The two main reasons behind the fact that our method is significantly faster than FSG is that our method (i) does not calculate the frequency of the each new pattern by locating the copies of this pattern in the network using subgraph isomorphism as FSG does, and (ii) it ensures that every generated pattern exists at least once in the underlying graph.

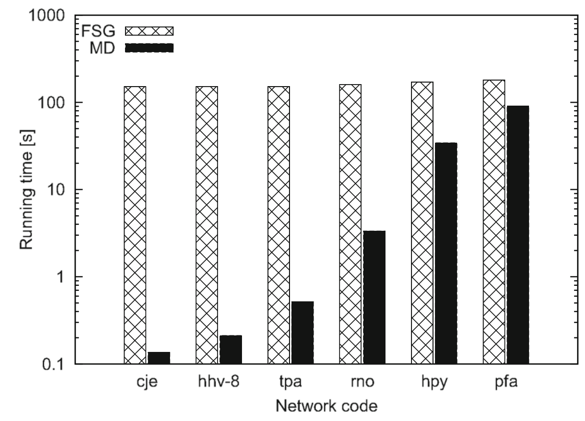
Evaluation of statistical significance

In this experiment, we evaluate the statistical significance of the most abundant motif identified by our method in each of the six PPI networks (see Table 1). We compute the statistical significance of the abundance of the most frequent motif of a given size in two alternative approaches. Each of these two approaches measures a different aspect of the significance.

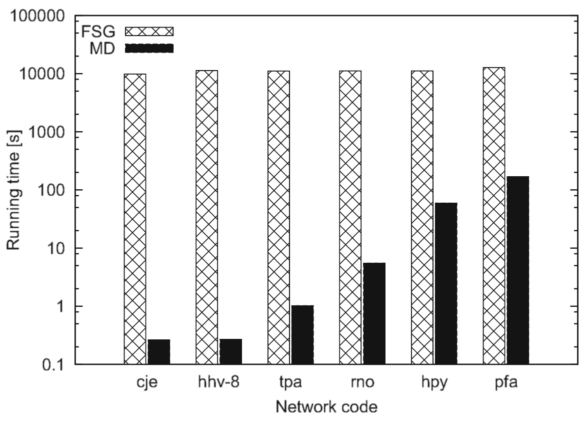
- The first approach measures the statistical significance of the frequency of most abundant motif with respect to the abundances of all motifs with the same size in the same graph. More specifically, given a target graph $G = (V, E)$ and motif size μ, we first find all motifs of size μ in G. Assume that there are totally m such motifs. Let us denote the frequency of these motifs with $x_1, x_2, \ldots, x_m$, with x_1 being the largest among all. Let us denote the mean and standard deviation of these m frequency values with $\bar{x}$ and σ. We report the z-score of the frequency of the most abundant motif as $\frac{x_1 - \bar{x}}{\sigma}$.
- The second approach measures the statistical significance of the frequency of the most abundant motif in the original graph with respect to those in



(a) Motif size = 7



(b) Motif size = 8



(c) Motif size = 9

Fig. 13 The total running time of our method (MD) and FSG for the real PPI networks (see Table 1). The y-axis shows the running time in seconds for three motif sizes; **a**. motif size = 7, **b**. motif size =8, and **c**. motif size =9

the random ensemble of graphs of the same size and degree distributions. More specifically, given a target graph $G = (V, E)$ and motif size μ, let us denote the frequency of the most abundant motif of this size in G with x. We construct a set of n random networks from G through *degree preserved edge shuffling* [35, 36]. Note that degree preserved edge shuffling is an iterative technique, which is often used in the literature to construct random network topologies with same size and degrees as a given target graph $G = (V, E)$. At each iteration of this technique, we randomly pick two edges from E. Let us denote these edges with (v_1, v_2) and (u_1, u_2), where v_1, v_2, u_1, $u_2 \in V$. We remove these two edges from E and insert two new edges (v_1, u_2) and (u_1, v_2). This way as the network topology evolves randomly, we ensure that the degrees of all the nodes remain unchanged. We repeat these iterations large number of times (exactly $10 \times |E|$ times) to randomize the entire network. Using the strategy above, we generate 100 random graphs, denoted with $G_1, G_2, \ldots, G_{100}$. For each random graph G_i, we measure the frequency of the most abundant motif of size μ. Let us denote this number as x_i. Let us denote the mean and standard deviation of these 100 frequency values with $\bar{x}$ and σ. We report the z-score of the frequency of the most abundant motif as $\frac{x-\bar{x}}{\sigma}$.

For both of the approaches above, we assume that a z-score above 2 or below -2 implies high statistical significance (i.e., two standard deviations away from the mean). The larger the magnitude of z-score is, the more significant the result is. Tables 2 and 3 present the z-score for each of the six PPI network and three motif size ($\mu = 5, 10, 15$) combinations using the first and the second approach described above respectively.

Table 2 suggests that, for small motif size (i.e. $\mu = 5$), the most abundant motif is not significantly more frequent than other motifs of the same size. However, as motifs get large in size (i.e. $\mu = 10$ and 15), the gap between the frequency of the most abundant motif and the rest of the motifs becomes highly significant. This implies that larger motifs characterize topological properties of PPI networks better than small motifs. This is because when motif size

Table 2 The *z*-scores that represent signifncance of the most abundant motif aginast other motifs in in the same network in each PPI network usig three motif size

Network code	Motif size = 5	Motif size = 10	Motif size = 15
hhv-8	1.52	14.00	4.67
cje	1.41	5.53	12.12
tpa	1.45	7.19	3.36
rno	1.58	4.31	9.74
hpy	1.54	13.70	9.003
pfa	1.87	35.32	7.43

Table 3 The *z*-scores that represent signifncance of the most abundant motif aginast most abundant motifs in 100 random networks in each PPI network usig three motif sizes

Network code	Motif size = 5	Motif size = 10	Motif size = 15
hhv-8	2.79	-0.54	-2.83
cje	2.32	0.99	-0.82
tpa	3.21	5.27	2.83
rno	-0.49	-4.02	-4.83
hpy	22.42	8.61	6.15
pfa	10.53	5.16	4.80

is small different motifs have similar frequency values, and this cannot be statistically different in abundance than each other. On the other hand, for large motif size, although the number of unique motif topologies is large, they vary a lot in their abundances; the most frequent one gets significantly more abundant than the rest.

Table 3 shows that, for most of the PPI network and motif size combinations, the most abundant motif is highly over-represented in the original network compared to random networks. In three cases (*Rattus norvegicus*, $\mu = 10$ and 15, and *Human herpesvirus8*, $\mu = 15$), we observe that the most abundant is significantly under-represented. These results demonstrate that the motif abundance in PPI networks is not random for nearly all combinations we tested. Thus, studying these structures has great potential to help understand how these networks function. Among the six PPI networks, *Rattus norvegicus* stands out to be the one with consistently under-represented or random motif abundance. The PPI of *Helicobacter pylori* consistently has the most significant motif abundance for all motif sizes. This indicates that the interactions in this network follow a regular pattern repeating themselves at different locations of the network. Finally, notice that the two z-score values reported in Tables 2 and 3 do not follow the same pattern (that is a high z-score according to one measure does not imply a high value for the other). This implies that the frequencies of different motifs (i.e., including the ones which are not most abundant) in these PPIs differ from those in random networks. In other words, the PPI networks topologically deviate from random networks.

Case study on Human herpesvirus

Here we briefly analyze the motifs identified by our method on the hhv-8 PPI network which causes Kaposi's sarcoma disease. We choose this organism in our case study as it has the smallest PPI network among the organisms in our database (see Table 1). Notice from Fig. 11c that despite its small size (48 nodes and 82 edges), hhv-8 has four disjoint embeddings of a very large motif with 10 nodes, covering a significant fraction of its PPI network. This begs the question whether there is a fundamental recurring function that hhv-8 serves and is covered through evolutionary process with high redundancy. Figure 14 presents the structure of those four embeddings. Each row of Table 4 lists the Uniprot ids of the ten proteins that contribute to each of those embeddings. Analysis of these proteins in the Gene Ontology database [37] reveals that three of those four embeddings, each contains two proteins one responsible for viral DNA packaging (O40944 and P88919) and one responsible for virion assembly (P88954). Without either process, no infectious progeny virus could be formed [38]. Several studies use these two processes as targets to identify effective inhibitors. The existence of these two process in each of the three instances reflects the functional importance of the motif topology found. These results suggest that our algorithm can find significant and valuable motifs which can be use to detect key functions governed by the network processes.

Conclusions

In this paper, we developed a scalable method to solve the motif identification problem given an input graph, desired motif size μ, and minimum frequency of desired motif α. We proposed a set of small patterns, we call *basic building patterns* each containing two or three edges. We proved that any motif with four or more edges can be constructed as a join of these patterns. Our method first locates instances of the basic building patterns. It then iteratively grows known motifs at that iteration by joining them with the instances of these patterns. We developed efficient mechanisms to avoid a significant fraction of the

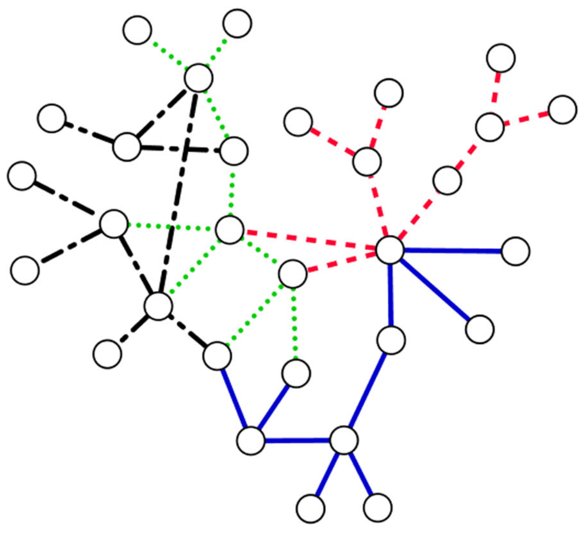


Fig. 14 The organization of the four isomorphic subgraphs of 10 nodes in the hhv-8 PPI network. Each supgraph has different color and pattern

Table 4 Each row lists the Uniprot IDs of the proteins in an embedding of the most abundant motif of size 10 found by our method in hhv-8 PPI network

O40944	P88947	P88935	P88951	P88960	P88940	P90489	P88918	P90495	P88902
O40910	O40944	P88947	P88929	P88920	P88925	P88927	P90486	P88918	P88954
P88918	P88919	P88929	P88948	P88920	P88950	O36551	P88942	Q98141	P88954
O40944	Q98141	P88920	P88951	P88954	P88947	P88948	P88958	P88939	P88944

costly isomorphism tests. We also introduced a new and efficient strategy for solve the MIS extraction problem. We analyzed the time complexity of our method based on the number of nodes and edges in the target network and the number of frequent motifs at each iteration. Our experiments on PPI networks from MINT comprehensively demonstrated that our method is significantly faster and more accurate than the existing methods. Furthermore, we observed using synthetic networks that the running time of our algorithm is reasonable with growing the size of the target network and network density. We also showed using PPI networks that the increase in the running time of our algorithm is dramatically less than that of the competing methods as the motif size grows. We evaluated the statistical significant of the most abundant motif of PPI networks resulting from our algorithm.

Additional file

Additional file 1: Appendix 1. This appendix shows the algebraic derivation of number of embeddings for each three of the four basic building blocks (see Section 1). In addition, the appendix lists further experimental analysis. Appendix file is attached as PDF file. (ZIP 151 kb)

Abbreviations
FBF: Flexible pattern finder; MIS: Maximum independent set; PPI: Protein-protein interaction

Acknowledgments
Not applicable.

Funding
This work has been supported partially by the US National Science Foundation under grant DBI-1262451.

Availability of data and materials
The real dataset supporting the conclusions of this article is publicly available at http://bioinformatics.cise.ufl.edu/code/nm-data.zip.

Authors' contributions
Both authors participated in the design, and evaluation of the method described in this paper. RE implemented the methods and gathered experimental results. Both authors wrote and approved the final manuscript.

Authors' information
We have no additional information about the authors.

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

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Received: 6 April 2016 Accepted: 21 September 2016
Published online: 06 October 2016

References
1. Zhu X, Gerstein M, Snyder M. Getting connected: analysis and principles of biological networks. Genes Dev. 2007;21(9):1010–1024.
2. Charlebois DA, Balázsi G, Kærn M. Coherent feedforward transcriptional regulatory motifs enhance drug resistance. Phys Rev E. 2014;89(5):052708.
3. Ay F, Kellis M, Kahveci T. SubMAP: aligning metabolic pathways with subnetwork mappings. J Comput Biol. 2011;18(3):219–35.
4. Wuchty S, Stadler PF. Centers of complex networks. J Theor Biol. 2003;223(1):45–53.
5. 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.
6. Shen-Orr SS, Milo R, Mangan S, Alon U. Network motifs in the transcriptional regulation network of escherichia coli. Nat Genet. 2002;31(1):64–8.
7. Wang P, Lü J, Yu X. Identification of important nodes in directed biological networks: A network motif approach. PLOS ONE. 2014;9(8): e106132.
8. Wuchty S, Oltvai ZN, Barabási AL. Evolutionary conservation of motif constituents in the yeast protein interaction network. Nat Genet. 2003;35(2):176–9.
9. Masoudi-Nejad A, Schreiber F, Kashani Z. Building blocks of biological networks: a review on major network motif discovery algorithms. IET Syst Biol. 2012;6(5):164–74.
10. Milenković T, Lai J, Pržulj N. Graphcrunch: a tool for large network analyses. BMC Bioinformatics. 2008;9(1):70.
11. Deshpande M, Kuramochi M, Wale N, Karypis G. Frequent substructure-based approaches for classifying chemical compounds. IEEE Trans Knowl Data Eng. 2005;17(8):1036–50.
12. Yanover C, Singh M, Zaslavsky E. M are better than one: an ensemble-based motif finder and its application to regulatory element prediction. Bioinformatics. 2009;25(7):868–74.
13. Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness: WH Freeman New York; 1979.
14. Cook SA. The complexity of theorem-proving procedures. In: ACM Symposium on Theory of Computing. ACM; 1971. p. 151–8.
15. Holder LB, Cook DJ, Djoko S, et al. Substucture discovery in the subdue system. In: KDD Workshop. Workshop on Knowledge Discovery in Databases; 1994. p. 169–80.
16. Schreiber F, Schwöbbermeyer H. Frequency concepts and pattern detection for the analysis of motifs in networks. In: Transactions on Computational Systems Biology. Springer; 2005. p. 89–104.
17. Vanetik N, Gudes E, Shimony SE. Computing frequent graph patterns from semistructured data. In: ICDM. IEEE; 2002. p. 458–65.
18. Yan X, Zhou X, Han J. Mining closed relational graphs with connectivity constraints. In: ACM SIGKDD. ACM; 2005. p. 324–33.
19. Grochow JA, Kellis M. Network motif discovery using subgraph enumeration and symmetry-breaking. In: Research in Computational Molecular Biology. Springer; 2007. p. 92–106.
20. Kashtan N, Itzkovitz S, Milo R, Alon U. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. Bioinformatics. 2004;20(11):1746–58.

21. Omidi S, Schreiber F, Masoudi-Nejad A. Moda: an efficient algorithm for network motif discovery in biological networks. Genes Genet Syst. 2009;84(5):385–95.
22. Wernicke S. Efficient detection of network motifs. IEEE/ACM Trans Comput Biol Bioinformatics (TCBB). 2006;3(4):347–59.
23. Chen J, Hsu W, Lee ML, Ng SK. Nemofinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs. In: ACM SIGKDD. ACM; 2006. p. 106–15.
24. Kashani ZR, Ahrabian H, Elahi E, Nowzari-Dalini A, Ansari ES, Asadi S, Mohammadi S, Schreiber F, Masoudi-Nejad A. Kavosh: a new algorithm for finding network motifs. BMC Bioinformatics. 2009;10(1):318.
25. Kuramochi M, Karypis G. An efficient algorithm for discovering frequent subgraphs. IEEE Trans Knowl Data Eng. 2004;16(9):1038–1051.
26. Kuramochi M, Karypis G. Finding frequent patterns in a large sparse graph. Data Mining Knowl Discov. 2005;11(3):243–71.
27. Babai L, Luks EM. Canonical labeling of graphs. In: ACM Symposium on Theory of Computing. ACM; 1983. p. 171–83.
28. Barabási AL, Albert R. Emergence of scaling in random networks. Science. 1999;286(5439):509–12.
29. Baskerville K, Paczuski M. Subgraph ensembles and motif discovery using a new heuristic for graph isomorphism. Phys Rev E. 2006;74:051903.
30. Chatr-Aryamontri A, Ceol A, Palazzi LM, Nardelli G, Schneider MV, Castagnoli L, Cesareni G. MINT: the Molecular INTeraction database. Nucleic Acids Res. 2007;35(suppl 1):572–4.
31. Dorogovtsev SN, Mendes JFF, Samukhin AN. Structure of growing networks with preferential linking. Phys Rev Lett. 2000;85(21):4633.
32. Jeong H, Tombor B, Albert R, Oltvai ZN, Barabási AL. The large-scale organization of metabolic networks. Nature. 2000;407(6804):651–4.
33. Redner S. How popular is your paper? an empirical study of the citation distribution. Eur Phys J B-Condensed Matter Complex Syst. 1998;4(2): 131–4.
34. Leclerc RD. Survival of the sparsest: robust gene networks are parsimonious. Mol Syst Biol. 2008;4(1):213.
35. Milo R, Kashtan N, Itzkovitz S, Newman ME, Alon U. On the uniform generation of random graphs with prescribed degree sequences. 2003. arXiv preprint cond-mat/0312028.
36. Gale D, et al. A theorem on flows in networks. Pacific J Math. 1957;7(2): 1073–82.
37. Ashburner M, Ball CA, et al. Gene ontology: tool for the unification of biology. Nat Genet. 2000;25(1):25–9.
38. Homa FL, Brown JC. Capsid assembly and dna packaging in herpes simplex virus. Rev Med Virol. 1997;7(2):107.