Network Motif Discovery Using Subgraph Enumeration and Symmetry-Breaking

Joshua A. Grochow and Manolis Kellis

Computer Science and AI Laboratory, M.I.T.
Broad Institute of M.I.T. and Harvard
joshuag@cs.uchicago.edu (current affiliation), manoli@mit.edu

Abstract. The study of biological networks and network motifs can yield significant new insights into systems biology. Previous methods of discovering network motifs – network-centric subgraph enumeration and sampling – have been limited to motifs of 6 to 8 nodes, revealing only the smallest network components. New methods are necessary to identify larger network sub-structures and functional motifs.
Here we present a novel algorithm for discovering large network motifs that achieves these goals, based on a novel symmetry-breaking technique, which eliminates repeated isomorphism testing, leading to an exponential speed-up over previous methods. This technique is made possible by reversing the traditional network-based search at the heart of the algorithm to a motif-based search, which also eliminates the need to store all motifs of a given size and enables parallelization and scaling. Additionally, our method enables us to study the clustering properties of discovered motifs, revealing even larger network elements.
We apply this algorithm to the protein-protein interaction network and transcription regulatory network of *S. cerevisiae*, and discover several large network motifs, which were previously inaccessible to existing methods, including a 29-node cluster of 15-node motifs corresponding to the key transcription machinery of *S. cerevisiae*.

1 Introduction

1.1 Network Motifs

In the past decade, new technologies have enabled the observation and study of networks of thousands and millions of nodes, such as social networks, computer networks, and, notably, *biological networks*, including protein-protein interaction networks [4–6], genetic regulatory networks [12, 18], and metabolic networks [7]. In order to extract meaningful information from these vast and sometimes noisy datasets, it is necessary to develop methods of computational analysis that are both efficient and robust to errors in the underlying data.

Network motifs – patterns of connectivity that occur significantly more frequently than expected – were introduced by Milo *et al.* [18] and provide one such robust property of biological networks. Network motifs also provide an important tool for understanding the modularity and the large-scale structure of networks

[8, 13, 20, 25]. The importance of network motifs as information-processing modules has been modeled theoretically [12, 21] and verified experimentally [8, 13, 20, 25]. Network motifs also have numerous other applications: they have been used to classify networks into "superfamilies" [17], they have been used in combination with machine learning techniques to determine the most appropriate network model for a given real-world network [16], and they have been used to determine which properties to use in parsimony models of phylogeny [19].

Unfortunately, all of these applications are hampered by the limited size of motifs discoverable by current methods. Exact counting methods have only been reported to find motifs up to 4 nodes [18] and motif generalizations up to 6 nodes [10]. Subgraph sampling methods have found motifs up to 7 [9] and 8 nodes [1, 16]. The statistical measures developed by Ziv *et al.* [26] are an important step towards larger network structures, but unfortunately lack a one-to-one correspondence with subgraphs, making them potentially difficult to interpret. Motif generalizations [10] are another important step towards these goals, although current methods are still limited to finding motif generalizations of only 6 nodes.

This current size limitation leaves many fundamental questions unanswered, and significant additional insight could be gained by exploring larger subgraphs and finding larger motifs. [1, 10]. We should not expect *a priori* that the building blocks of complex networks are as small as 4 nodes, or that the largest significant structures and pathways contain only 8 nodes. What are the fundamental building blocks? How do they combine to form larger structures? [1, 10] Do networks which share the same building blocks also share the same combinations of these blocks? [10] How can larger structures be used to distinguish between networks of different types, or between proposed models for a given network? [1]

In this paper, we present a new approach for discovering network motifs. The heart of our algorithm exhaustively assesses the significance of **a single query subgraph** as a potential motif. This can then be applied to all subgraphs of a given size to emulate the behavior of previous exhaustive algorithms, but with an exponential speed-up due to a **novel symmetry-breaking technique** (which is not feasible with previous methods). The symmetry-breaking technique also allows us to write instances of a subgraph to disk as they are found, **further eliminating limitations due to memory usage**. We are thus able to find motifs of up to 15 nodes, to find all instances of subgraphs of 31 nodes, and potentially even larger subgraphs. Although this work is motivated by biological networks, and this paper focuses on the protein-protein interaction (PPI) network and the transcription network of *S. cerevisiae*, our methods are applicable to any network – directed or undirected – and thus to many different fields, even outside the realm of biology.

In this section, we review previous work and give an overview of our algorithm, outlining several novel techniques which apply both to our approach and to previous approaches. In Sec. 2 we present our algorithm in detail. In Sec. 3 we present benchmarks comparing our approach to previous approaches. Additionally, we present data as to the effectiveness of the resulting improvements as

applied to both the transcription and PPI networks of *S. cerevisiae*. In Sec. 4, we present some of the larger subgraphs we have discovered. Finally, in Sec. 5 we discuss the significance of these contributions for the understanding of networks in general.

1.2 Limitations of Network-Centric Approaches for Motif Discovery

Two basic techniques have been proposed for identifying network motifs: exact counting [18] and subgraph sampling [1, 9, 16]. These methods attempt to determine the significance of all or many subgraphs of a given size by comparing their frequency in a given network to their frequency in a random ensemble of networks with similar properties to the original. To determine which subgraphs are motifs, subgraph sampling [1] is an effective and efficient approach, and has been used to evaluate the significance of larger subgraphs than can be evaluated by the exact counting method.

Most methods for finding DNA *sequence motifs* scan or sample a sequence pattern from a genome. Similarly, previous techniques for finding network motifs scan or sample subgraphs from a network, and count the number of occurrences of each subgraph encountered. (This process is then repeated for each network in a random ensemble resembling the initial network, and the counts are compared.) For the discovery of DNA sequence motifs, this general methodology is very efficient, because sequence motifs can be efficiently hashed based on their content. Thus a single linear scan of the genome suffices to exhaustively count all possible substrings of a given size, *regardless of the size of the substrings*.

In contrast, for the discovery of network motifs, enumerating all subgraphs of a given size is in general *exponential in the number of nodes of the subgraphs*. Additionally, there is no known efficient algorithm that correctly identifies two graphs as isomorphic or not. (The *graph isomorphism problem* is not known to be either in P or to be NP-complete.) This intrinsic difference in complexity between discovering *sequence* motifs and discovering *network* motifs makes traditional network-scanning methodologies inefficient for network motif discovery.

1.3 Distinguishing Features of the New Algorithm

To avoid these limitations of the traditional network-centric approaches, we have taken a motif-centric approach which has several attractive features, outlined here. Features 1-3 are specific to motif-centric methods, while features 4 and 5 can also benefit traditional network-centric methods.

(1) Searching for a single query graph. To avoid the increased complexity of subgraph enumeration (in the absence of an appropriate hashing scheme) our algorithm works by exhaustively searching for the instances of a *single query graph* in a network. (To find all motifs of a given size we couple this search with subgraph enumeration, using McKay's `geng` and `directg` tools [15]). Even though the *subgraph isomorphism problem* – finding a given graph as a subgraph of a larger network – is known to be NP-complete, several algorithmic improvements

enable this search to be carried out effectively in practice, even for subgraphs up to 31 nodes (and potentially even more).

(2) Mapping instead of enumerating. Rather than enumerating all connected subgraphs of a given size and testing to see whether each is isomorphic to the query graph, our algorithm attempts to map the query graph onto the network in all possible ways. We developed this technique for subgraph isomorphism independently, but subsequently identified a prior use [23].

(3) Taking advantage of subgraph symmetries. We introduce a technique that *avoids spending time finding a subgraph more than once* due to its symmetries. This technique improves the speed of our method by a factor exponential in the size of the query subgraph (Table 1). Moreover, since each instance is discovered exactly once, our algorithm can write instances to disk as they are found, greatly improving memory usage.

(4) Improved isomorphism testing. Our isomorphism test takes into account the degree of each node, and the degrees of each node's neighbors, leading to marked improvements over current motif-finding algorithms, which use exhaustive graph isomorphism tests.

(5) Subgraph hashing. When examining all subgraphs of a given size we hash the graphs based on their degree sequences, which leads to a significant improvement in the number of isomorphism tests needed. In a *directed* network, we group the query graphs based on their *undirected* isomorphism types, find all instances, and then go back to the directed network and divide these instances into their directed isomorphism types.

2 Description of the Algorithm

For clarity, we first present the basic mapping algorithm for subgraph isomorphism, without taking into account the symmetries of the query graph. In Sec. 2.2 we incorporate our symmetry-breaking technique into the algorithm. In the pseudo-code, we identify statements used solely for symmetry-breaking by enclosing them in square brackets. Finally, In Sec. 2.3 we incorporate our technique into two new methods of finding motifs.

Throughout this section, G will denote the network being searched and H will denote the query subgraph. We say that a node g of G can *support* a node h of H if we cannot rule out a subgraph isomorphism from H into G which maps h to g based on the degrees of h and g and the degrees of their neighbors. (Other constraints could also be used here, but these two proved effective and simple to implement.) This notion of support is used to exclude inconsistent candidate maps during the backtracking search.

2.1 Finding a Given Subgraph (Subgraph Isomorphism)

We start by presenting the algorithm without symmetry-breaking. Note that symmetry-breaking is not required for correctness of the algorithm.

FindSubgraphInstances(H,G):
Finds all instances of query graph H in network G

Start with an empty set of instances.
[Find Aut(H). Let H_E be the equivalence representatives of H.]
[Find symmetry-breaking conditions C for H given H_E and Aut(H).]
Order the nodes of G by increasing degree and
 then by increasing neighbor degree sequence.
For each node g of G
 For each node h of H [H_E] such that g can support h
  Let f be the partial map associating $f(h) = g$.
  Find all isomorphic extensions of f [up to symmetry]
   i.e. call IsomorphicExtensions(f,H,G[,$C(h)$]).
  Add the images of these maps to the set of all instances.
 Remove g from G.
Return the set of all instances.

FindSubgraphInstances includes the *images* of the maps in the list of instances, thus merging all maps which differ only by a symmetry of H. (Without symmetry-breaking, the algorithm spends additional time finding several distinct maps to a single subgraph.)

IsomorphicExtensions is a backtracking search to find all isomorphisms from H into G. As is standard in backtracking searches, the algorithm uses the most constrained neighbor to eliminate maps that cannot be isomorphisms: that is, the neighbor of the already-mapped nodes which is likely to have the fewest possible nodes it can be mapped to. First we select the nodes with the most already-mapped neighbors, and amongst those we select the nodes with the highest degree and largest neighbor degree sequence.

For each call to IsomorphicExtensions, f is extended by a single node. Each time an extension is made, the algorithm ensures that the newly mapped node is appropriately connected to the already-mapped nodes. Thus when IsomorphicExtensions returns a map, it is guaranteed to be an isomorphism.

We have effectively pushed the isomorphism testing of previous exhaustive methods into IsomorphicExtensions, which allows the isomorphism test to abort early. The ability to abort early when finding instances of a particular query graph presents significant savings over previous methods.

2.2 Exploiting Subgraph Symmetries to Speed Up the Search

Due to symmetries, a given subgraph of G may be mapped to a given query graph H multiple times. For example, the subgraph in Fig. 1 can be mapped to the same 6 nodes in 8 different ways. Thus a simple mapping-based search for a query graph will find each instance of the query graph as many times as the graph has symmetries. To avoid this, we compute and enforce several symmetry-breaking conditions, which ensure that there is a *unique* map from the query graph H to each instance of H in G, so that our search only spends time finding each instance once.

IsomorphicExtensions**(f,H,G[,C(h)]:**
Finds all isomorphic extensions of partial map $f : H \to G$ [satisfying C(h)]

Start with an empty list of isomorphisms.
Let D be the domain of f.
If $D = H$, return a list consisting solely of f. (Or write to disk.)
Let m be the most constrained neighbor of any $d \in D$
(constrained by degree, neighbors mapped, etc.)
For each neighbor n of $f(D)$
If there is a neighbor $d \in D$ of m such that n is *not* neighbors with $f(d)$,
or if there is a *non*-neighbor $d \in D$ of m such that n *is* neighbors with $f(d)$
[or if assigning $f(m) = n$ would violate a symmetry-breaking condition in $C(h)$],
then continue with the next n.
Otherwise, let $f' = f$ on D, and $f'(m) = n$.
Find all isomorphic extensions of f'.
Append these maps to the list of isomorphisms.
Return the list of isomorphisms.

The symmetries of a graph H are known as automorphisms (self-isomorphisms), and the group of automorphisms of H is denoted $\mathrm{Aut}(H)$. For a set A of automorphisms, two nodes are said to be "A-equivalent" if there is some automorphism in A which maps one to the other, or simply "equivalent" if $A = \mathrm{Aut}(H)$. We denote the A-equivalence of two nodes n_1, n_2 by $n_1 \sim_A n_2$. This equivalence relation partitions the nodes of H into equivalence classes. Since starting a map from two equivalent nodes is unnecessary and wasteful, FindSubgraphInstances uses a set consisting of one representative from each equivalence class of H.

The symmetry-breaking conditions are based on labellings of the nodes of H by integers, represented as maps from $H \to \mathbf{Z}$. Let $\ell : G \to \mathbf{Z}$ be a labelling of the nodes of G by *distinct* integers. Then each map $f : H \to G$ generates a labelling $L : H \to \mathbf{Z}$, given by $L(n) = \ell(f(n))$ for nodes $n \in H$. Thus, conditions on labellings of H translate into restraints on maps from H into G.

Given a set of conditions C, we say an automorphism α *preserves the conditions* C if, given a labelling L_1 of H which satisfies C, the corresponding labelling $L_2 : H \to \mathbf{Z}$ given by $L_2(n) = L_1(\alpha(n))$ also satisfies C. We are thus searching for conditions C such that the only automorphism which preserves C is the identity. This ensures there will be exactly one map from H onto each of its instances in G which satisfies the conditions.

To find these conditions, we pick an $\mathrm{Aut}(H)$-equivalence class $\{n_0, \ldots, n_k\}$ of nodes of H, and we impose the condition $L(n_0) < \mathrm{MIN}(L(n_1), \ldots, L(n_k))$. Any automorphism must send n_0 to one of the n_i, since these are all of the nodes equivalent to n_0. But to preserve this condition, an automorphism must send n_0 to itself. Then we continue recursively, replacing $\mathrm{Aut}(H)$ with the set A of automorphisms which send n_0 to itself. For example, see Fig. 1.

Because FindSubgraphInstances starts with a particular node, we can consider that node already fixed. (Note that the version of FindSubgraphInstances which uses symmetry-breaking only iterates over a set of equivalence

class representatives, and not over all nodes of H.) Thus for each representative used by FindSubgraphInstances, SymmetryConditions must generate a series of symmetry-breaking conditions which start by fixing that node.

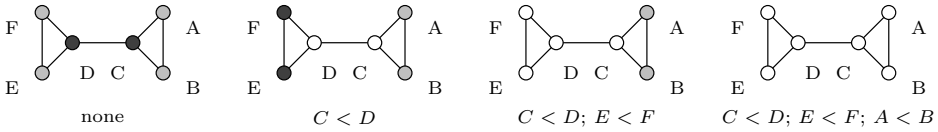

Fig. 1. Finding conditions that will break all the symmetries of a 6-node graph. White nodes are fixed by any automorphism preserving the indicated conditions, and other nodes are shaded according to their equivalence class under the automorphisms which preserve the indicated conditions.

To find the automorphisms of H, we use IsomorphicExtensions *without* symmetry-breaking, which returns an exhaustive list of all isomorphisms from H to itself. To find the automorphisms which fix a node or a set of nodes, the algorithm filters this list in a single pass.

Finding the automorphisms of a graph is thought to be computationally expensive[1], but in practice we have found this is far from the bottleneck in motif-finding algorithms. We were able to *exhaustively* find the automorphisms of all 11,117 8-node undirected graphs in under 30 seconds on a standard laptop, and McKay's tools [14] can find all the automorphisms of very large graphs very rapidly (e.g. some graphs with thousands of nodes and millions of automorphisms, in less than one second on a standard laptop).

SymmetryConditions:
Finds symmetry-breaking conditions for H given H_E, $\mathbf{Aut}(H)$

Let M be an empty map from equivalence representatives to sets of conditions.
For each $n \in H_E$
 Let C be an empty set of conditions.
 $n' \leftarrow n$, and $A \leftarrow \mathrm{Aut}(H)$.
 Do until $|A| = 1$:
  Add "$\textsc{label}(n') < \textsc{min}\{\textsc{label}(m) | m \sim_A n' \text{ and } m \neq n'\}$" to C.
  $A \leftarrow \{f \in A | f(n') = n'\}$.
  Find the largest A-equivalence class E.
  Pick $n' \in E$ arbitrarily.
 Let $M(n) = C$.
Return M.

[1] Finding graph automorphisms is at least as hard as determining if two graphs are isomorphic. Like the graph isomorphism problem, the graph automorphism problem is not known to be either in P or to be NP-complete.

2.3 Subgraph Evaluation and Network Motif Discovery

To find network motifs we enumerate candidate subgraphs H (exhaustively or by sampling), and evaluate each candidate based on its instances.

Evaluating candidate subgraphs. We find all instances of a query graph H in the network G, as well as in a random ensemble of networks with the same degree distribution and same distribution of 3-node subgraphs as G.[2] We evaluate the overrepresentation of the query graph H based on the z-score of its abundance in G against the distribution of its abundance in the random ensemble, as in [18, 21].

Exhaustive subgraph enumeration. Our method can be used to find all instances of subgraphs of a given size, similar to previous exhaustive methods. To do this, we generate all non-isomorphic graphs of a particular size using McKay's `geng` and `directg` tools [15]. Then for each graph, we evaluate its significance as above.

Subgraph sampling. Our method can also be used in combination with subgraph sampling. We sample connected subgraphs (usually relatively large, compared to previous network motifs: 10, 15, or 20 nodes) by picking a node at random, and taking a random walk until we have as many nodes as desired [16]. Then we assess the significance of this subgraph as above.

Sampling subgraphs to find anti-motifs. Some studies have also considered *anti*-motifs: subgraphs which are significantly *under*represented compared to randomized versions of the network. To use a sampling method to find anti-motifs, it might be more fruitful to sample initial subgraphs from the random ensemble rather than the network being studied. Anti-motifs will be more prevalent in the ensemble than in the target network, and thus are more likely to be discovered by sampling from the ensemble.

3 Results and Evaluation

We applied our algorithm to the PPI network (1379 nodes, 2493 edges) [4] and transcription network (685 nodes, 1052 edges) [2] of *S. cerevisiae* and compared its performance to previous methods of motif disccovery.

Comparison with previous methods: time. We compare the time requirements of our method to those of Milo *et al.* [18] (Fig. 2). We make this comparison on the undirected PPI network of *S. cerevisiae* [4], by exhaustively counting subgraphs up to 7 nodes.

We implemented both our algorithm and two versions of the Milo *et al.* algorithm [18]: both as originally presented [18], and also by additionally hashing subgraphs by their degree sequence (Sec. 1.3). Fig. 2 shows that our algorithm

[2] Although Shen-Orr *et al.*[21] use a model in which the distribution of $(n-1)$-node subgraphs is preserved when looking for n-node motifs, they only applied this to the case $n = 4$, and we have found it computationally infeasible to preserve this distribution for $n > 4$. Nonetheless, we have found it fruitful to preserve the distribution of 3-node subgraphs, regardless of n.

provides an *exponential* improvement in time, even compared to the modified version of the previous algorithm [18].

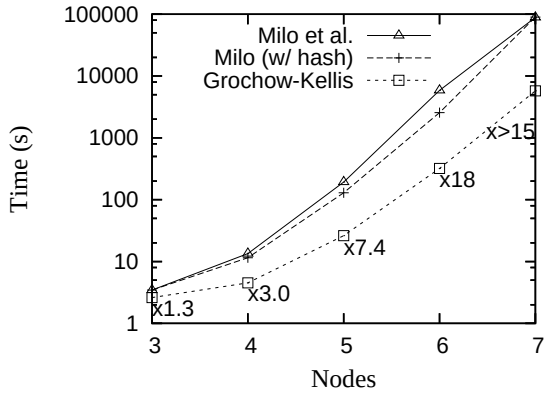

Fig. 2. The runtimes of the original algorithm of Milo *et al.* [18], an improved version of their algorithm, and our new algorithm, as applied to the undirected PPI network of *S. cerevisiae* [4]. The speed-up from the original algorithm of Milo *et al.* [18] to our algorithm is indicated. (Note: the values for 7 nodes for the two variants of Milo *et al.*'s algorithm are underestimates: the program ran out of memory before finishing.)

Comparison with previous methods: space. Our method gains an exponential memory advantage over previous exhaustive methods by not keeping a list of previously visited subgraphs. In the previous exact counting method [18], a list of the subgraphs encountered at each node is necessary in order to avoid duplication, even when the instances of the subgraphs are not desired as output. Thus the space required by the previous method is proportional to the number of subgraphs of a given size going through a given node, which can be exponential in the size of the subgraphs. Because our method does not need to keep such a list, its asymptotic memory requirements are determined by the maximum depth of recurion of IsomorphicExtensions, which is linear in the size of the query graph. Our method thus uses exponentially less space than previous exhaustive methods.

Disk usage. Furthermore, our algorithm uses no more memory to find a list of all instances than to simply count the instances. Since each instance is encountered exactly once, it can be written to disk and removed from active memory as soon as it is encountered, using effectively no additional memory.

Improvement due to symmetry-breaking. The main reason for these improvements is our novel symmetry-breaking technique. Symmetry-breaking ensures that each instance is discovered exactly once, so our method does not have to check a list of the subgraphs previously encountered at a node in order to avoid duplicate counting, while the previous method of exact counting does. Ta-

ble 1 quantifies this contribution explicitly, showing that the average number of automorphisms of graphs weighted by their occurences in the PPI network and regulatory network of yeast – i.e. the savings gained by symmetry-breaking – appears to grow exponentially.

Table 1. The number of subgraphs encountered by our algorithm with and without symmetry-breaking (including multiple encounters for the version without symmetry-breaking). The improvement factor is exactly the average number of automorphisms of subgraphs of the associated size.

	Undirected PPI Network			Directed Regulatory Network		
Nodes	Total Subgraphs Searched	With Symmetry-Breaking	Improvement	Total Subgraphs Searched	With Symmetry-Breaking	Improvement
3	3.7×10^4	1.1×10^4	$\times3.13$	2.6×10^4	1.3×10^4	$\times2.02$
4	4.0×10^5	7.0×10^4	$\times5.77$	9.7×10^5	1.8×10^5	$\times5.41$
5	4.4×10^6	4.1×10^5	$\times10.9$	4.4×10^7	2.5×10^6	$\times18.0$
6	5.1×10^7	2.3×10^6	$\times22.2$	2.3×10^9	3.2×10^7	$\times73.3$
7	5.7×10^8	1.2×10^7	$\times46.3$	1.3×10^{11}	4.0×10^8	$\times334$
8	6.4×10^9	6.6×10^7	$\times96.2$	—	—	—

4 Discovered Motifs and Their Biological Significance

Discovered motifs. We exhaustively evaluated all candidate motifs and anti-motifs up to 7 nodes in the PPI network of *S. cerevisiae*[4] (1379 nodes, 2493 edges). We used a random ensemble of networks with the same degree distribution and the same distribution of 3-node subgraphs as the PPI network.[3] The most significant subgraphs tend to be motifs rather than anti-motifs: using a z-score cutoff of 4.0, only 3 of the 54 significant subgraphs of size at most 7 were anti-motifs. Two of the motifs were trees, and the most dense motif had 18 edges. Most of the significant graphs were of moderate density: the mean number of edges for 7-node motifs and anti-motifs is 11.49 ± 2.89.

Large discovered motifs. We also discovered larger motifs by first sampling connected subgraphs from the PPI network of *S. cerevisiae*, and then assessing their significance using our method. We sampled approximately 100 connected subgraphs of 15 and 20 nodes, and found several motifs. One such 15-node motif (Fig. 3) represents a common connectivity pattern found within the transcriptional machinery of *S. cerevisiae* (see discussion below).

Clustering of discovered motifs and larger network structures. We noted that almost all of the larger subgraphs we evaluated have large numbers of overlapping instances, which become apparent since our method reports all network instances of a discovered motif. To quantify this property, we developed

[3] See footnote 2.

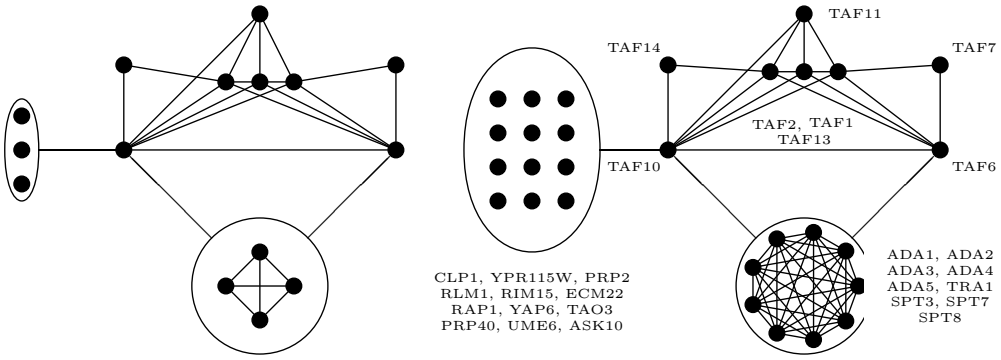

Fig. 3. A motif of 15 nodes and 34 edges (left). An edge from a group of nodes to a node n indicates that each node in the group is connected to n. This motif appears 27,720 times in the PPI network of *S. cerevisiae*[4], and does not appear at all in random ensembles which preserve the degree distribution and the distribution of 3-node subgraphs. All 27,720 instances are clustered into a total of 29 nodes (right), corresponding to the cellular transcription machinery.

a subgraph clustering score, based on the number of subgraph instances overlapping a given node, averaged over all nodes in any subgraph instance. We applied this score to evaluate the clustering properties of all discovered motifs, and we found that indeed some of the most abundant motifs show striking clustering properties.

The clustered instances frequently reveal important larger network structures. For example, the 15-node motif of Fig. 3 occurs 27,720 times in a single sub-network of 29 nodes, part of the core transcription machinery of *S. cerevisiae*. This includes a complete 11-node graph (including the two central hubs) corresponding to the SAGA complex, and consisting almost entirely of chromatin modification and histone acetylation factors an 8-node core (shared by all instances of the 15-node motif) corresponding to the TFIID complex, and 12 attachments, which are known activators and suppressors of these two complexes [11]. Similarly, the subgraph of 20 nodes shown in Fig. 4 occurs 5,020 times in a total of 31 nodes, enriched in cell-cycle regulation.

The role of combinatorial effects. The extreme clustering properties of the most abundant motifs appear to result from combinatorial connectivity patterns prevalent in larger network structures. For example, all 27,720 instances of the 15-node motif in Fig. 3 result by choosing 3 attachments from the left and 4 attachments from the the bottom of Fig. 3 ($\binom{12}{3}\binom{9}{4} = 27,720$), and similarly for the 5,020 instances of the 20-node subgraph in Fig. 4. Additionally, in the random ensemble, these combinatorially appearing structures occur either thousands of times, or not at all – they almost never occur just a few or a few hundred times.

Although motif clustering has previously been observed [3] and demonstrated analytically [24], previous motifs studied do not have enough nodes to exhibit the extreme combinatorial clustering we observed for large subgraphs (at least 15 nodes). The magnitude of this combinatorial clustering effect brings into question

the current definition of network motif, when applied to larger structures. We propose that additional statistics, either alone or in combination, might be well-suited to identify larger meaningful network structures: our subgraph clustering score, the total number of nodes covered by all instances of the query graph, the total number of edges, and the weighting of the number of nodes/edges based on the number of overlapping instances. All of these statistics can be easily calculated using our algorithm, since it finds and stores all motif instances, and these will be the subject of future studies.

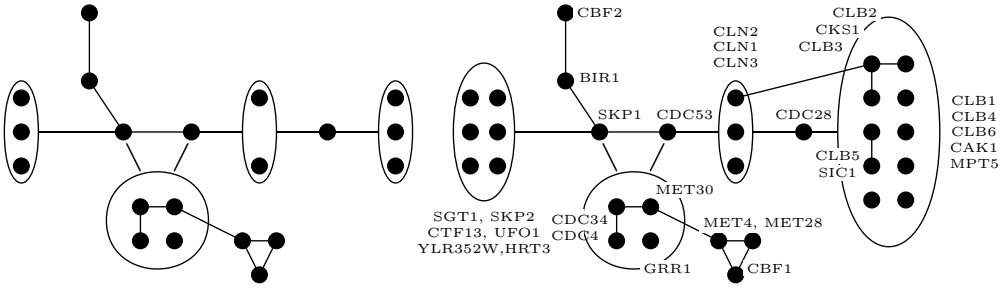

Fig. 4. A subgraph of 20 nodes and 27 edges (left). An edge from a group of nodes to a node n indicates that each node in the group is connected to n. This subgraph appears 5,020 times in the PPI network of *S. cerevisiae* [4]. All 5,020 instances are clustered into a total of 31 nodes (right), enriched in cell-cycle regulation.

5 Discussion

We presented a novel approach to the discovery of network motifs, based on a solution to the subgraph isomorphism problem that uses a new symmetry-breaking technique, an improved isomorphism test, and hashing based on degree sequences. Several of the techniques presented in our algorithm can also be used in previous algorithms, and lead to significant improvements.

We implemented our algorithm and used it to find significant structures of 15 and 20 nodes in the PPI network and the regulatory network of *S. cerevisiae*, where previous methods had been limited to motifs of 6 and 8 nodes. Using our approach to motif-finding, we re-discovered the cellular transcription machinery – as a 29-node cluster of 15-node motifs – based solely on the structure of the protein interaction network.

Previous methods of motif discovery were network-centric, and could therefore not take advantage of subgraph symmetries. By using a motif-centric algorithm instead, we are able to use symmetry-breaking to get an exponential improvement.

5.1 Applications and Advantages of the New Method

(1) Finding larger motifs. Our improvements have enabled the exhaustive discovery of motifs up to 7 nodes. To find even larger motifs, we sample a connected subgraph as in [16], and then find *all* its instances and assess its significance using our method. This technique has enabled us to find motifs up to 15 nodes and examine subgraphs up to 31 nodes.

(2) Querying a particular subgraph. Our method can be used to query whether a particular subgraph is significant, whereas previous methods can only do this by examining all subgraphs of the same size, which quickly becomes prohibitive for even moderate sizes. This technique could be used to explore *in silico* the prevalence of a subgraph of interest, identified experimentally (e.g. known pathways), computationally (e.g. motif generalizations [10]), or genetically.

(3) Exploring motif clustering. Because our algorithm finds all instances of a given subgraph, it can be used to explore how these instances cluster together to form larger structures. For example, after finding a 15-node motif, we were able to determine that all of its 27,720 instances clustered in 29 nodes (Fig. 3).

(4) Time and space. Our method, applied to all subgraphs of a given size, takes exponentially less time than previous methods, even when we implement the previous method with our hashing scheme (Sec. 3). Additionally, there are essentially no space limitations on our method: since each instance is found exactly once due to our symmetry-breaking technique, it can be written to disk and removed from active memory as soon as it is found.

(5) Parallelization. Our method is more easily parallelizable than previous motif-finding methods, since each subgraph can be counted on a separate processor. We have found this attribute to be very useful, and we believe other researchers will as well, as cluster computing becomes commonplace in the computational biology community.

5.2 Clustering Properties of Large Subgraphs and Motifs

We revealed that larger subgraphs tend to cluster together *combinatorially* – that is, all instances share a significant core of nodes, and each instance represents a choice of attachments to these core nodes. This combinatorial clustering brings into question the relevance of the standard definition of network motif for large subgraphs of 15 nodes or more. We proposed several statistics which may be more appropriate in this domain.

Finally, we mention that the statistics of Ziv *et al.* [26] may not suffer from these combinatorial effects. The main drawback of these statistics is their lack of one-to-one correspondence with subgraphs. In combination with our algorithm, however, the large subgraphs encompassed by these statistics could be further explored, allowing for a clearer interpretation of the most significant statistics.

Moving forward, we expect the network motifs and methodology presented here will open a window into the large structures and global organization of biological and other networks.

Acknowledgements. The authors would like to thank Pouya Kheradpour, Mike Lin, Matt Rasmussen, Alex Stark, and Radek Szklarczyk (all at the M.I.T. Computer Science and AI Laboratory) for many useful and interesting discussions.

All algorithms were implemented in Java using the Java Universal Networks and Graphs (JUNG) framework [22]. Our software is available on request. McKay's `geng` and `directg` tools [15] were used to enumerate all graphs of a given size. Much of the computational work was carried out on the compute cluster at the Broad Institute of M.I.T. and Harvard.

This work was supported in part by startup funds from Professor Kellis.

References

1. K. Baskerville and M. Paczuski. Subgraph ensembles and motif discovery using a new heuristic for graph isomorphism, 2006. arxiv.org:q-bio/0606023.
2. M. C. Costanzo, M. E. Crawford, J. E. Hirschman, J. E. Kranz, P. Olsen, L. S. Robertson, M. S. Skrzypek, B. R. Braun, K. L. Hopkins, P. Kondu, C. Lengieza, J. E. Lew-Smith, M. Tillberg, and J. I. Garrels. Ypd(tm), pombepd(tm), and wormpd(tm): model organism volumes of the bioknowledge(tm) library, an integrated resource for protein information. *Nucleic Acids Res.*, 29:75–79, 2001.
3. R. Dobrin, Q. K. Beg, A.-L. Barabási, and Z. N. Oltvai. Aggregation of topological motifs in the *Escherichia coli* transcriptional regulatory network. *BMC Bioinformatics*, 5:10, Jan 2004.
4. J.-D. J. Han, N. Bertin, T. Hao, D. S. Goldberg, G. F. Berriz, L. V. Zhang, D. Dupuy, A. J. M. Walhout, M. E. Cusick, F. P. Roth, and M. Vidal. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, 430(6995):88–93, Jul 2004.
5. A. Jaimovich, G. Elidan, H. Margalit, and N. Friedman. Towards an integrated protein-protein interaction network: a relational markov network approach. *J. Comp. Bio.*, 13:145–164, 2006.
6. H. Jeong, S. Mason, A.-L. Barabási, and Z. N. Oltvai. Centrality and lethality of protein networks. *Nature*, 411, 2001.
7. H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407, 2000.
8. S. Kalir, J. McClure, K. Pabbaraju, C. Southward, M. Ronen, S. Leibler, M. G. Surette, and U. Alon. Ordering genes in a flagella pathway by analysis of expression kinetics from living bacteria. *Science*, 292(5524):2080–2083, Jun 2001.
9. N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, Jul 2004. Evaluation Studies.
10. N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Topological generalizations of network motifs. *Phys. Rev. E*, 70:031909, 2004.
11. T. I. Lee and R. A. Young. Transcription of eukaryotic protein-coding genes. *Annu. Rev. Genet.*, 34:77–137, 2000.
12. S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *PNAS*, 100(21):11980–11985, Oct 2003.
13. S. Mangan, A. Zaslaver, and U. Alon. The coherent feedforward loop serves as a sign-sensitive delay element in transcription networks. *J. Mol. Biol.*, 334(2):197–204, Nov 2003.

14. B. D. McKay. Practical graph isomorphism. In *Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, Vol. I (Winnipeg, Man., 1980)*, volume 30, pages 45–87, 1981. http://cs.anu.edu.au/∼bdm/nauty/.
15. B. D. McKay. Isomorph-free exhaustive generation. *J. Algorithms*, 26:306–324, 1998.
16. M. Middendorf, E. Ziv, and Chris H. Wiggins. Inferring network mechanisms: the Drosophila melanogaster protein interaction network. *PNAS*, 102(9):3192–3197, Mar 2005.
17. R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, Mar 2004.
18. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, Oct 2002.
19. T. M. Przytycka. An important connection between network motifs and parsimony models. In *RECOMB 2006*, pages 321–335, 2006.
20. M. Ronen, R. Rosenberg, B. I. Shraiman, and U. Alon. Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics. *Proc Natl Acad Sci U S A*, 99(16):10555–10560, Aug 2002.
21. S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature Genetics*, 31(1):64–68, May 2002.
22. JUNG Framework Development Team. Jung: The java universal network/graph framework, 2005.
23. J. R. Ullman. An algorithm for subgraph isomorphism. *J. Assoc. Comp. Mach.*, 23(1):31–42, Jan 1976.
24. A. Vazquez, R. Dobrin, D. Sergi, J.-P. Eckmann, Z. N. Oltvai, and A.-L. Barabasi. The topological relationship between the large-scale attributes and local interaction patterns of complex networks. *PNAS*, 101(52):17940–17945, Dec 2004.
25. A. Zaslaver, A. E. Mayo, R. Rosenberg, P. Bashkin, H. Sberro, M. Tsalyuk, M. G. Surette, and U. Alon. Just-in-time transcription program in metabolic pathways. *Nature Genetics*, 36(5):486–491, May 2004.
26. E. Ziv, R. Koytcheff, M. Middendorf, and C. Wiggins. Systematic identification of statistically significant network measures. *Phys. Rev. E*, 71:016110, 2005.